Amazon SageMaker

开发人员指南

Amazon SageMaker: 开发人员指南

Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其它商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅中国的 Amazon Web Services 服务入门。

Table of Contents

什么是 Amazon SageMaker？ 1
Amazon SageMaker 功能 1
Amazon SageMaker 定价 3
您是首次接触 Amazon SageMaker 的用户吗？ 3
工作方式 3
借助 Amazon SageMaker 进行 Machine Learning 3
探索、分析和处理数据 5
公平性和模型可解释性 5
在机器学习生命周期中评估公平性和可解性的最佳实践 6
示例笔记本 6
SageMaker 澄清文档指南 7
模型训练 7
模型部署 9
验证模型 10
模型监控 10
ML 框架和工具包 11
Apache MXNet 11
Apache Spark 12
Chainer 20
拥抱面部联接 21
PyTorch 22
R 23
Scikit-learn 25
SparkML Serving 26
TensorFlow 26
Triton 推理服务器 27
支持的区域和配额 28
请求提升服务配额以下项目： SageMaker 资源 29
入门 30
设置 Amazon SageMaker 先决条件 30
创建 Amazon 账户 30
创建 IAM 管理员用户和组 31
注册到域 32
使用快速安装注册 32
使用 IAM 进行注册 33
选择 VPC。 35
删除域 36
SageMaker JumpStart 38
打开 JumpStart 38
使用 JumpStart 39
方案模板 40
模型 42
部署模型 44
模型部署配置 44
微调模型 52
微调数据源 52
微调部署配置 53
超参数 53
训练输出 54
SageMaker JumpStart 行业 54
开始使用笔记本实例 58
使用的 Machine Learning SageMaker Python 开发工具包 58
教程概述 58
第 1 步：创建 Amazon SageMaker 笔记本实例 59

第 2 步：创建 Jupyter 笔记本 60
第 3 步：下载、浏览和转换数据 60
第 4 步：训练模型 64
第 5 步：部署模型 68
第 6 步：评估模型 70
步骤 7：清除 73
Machine Learning 环境 75
SageMaker 域 75
SageMaker Studio 77
UI 概述 77
使用 Studio 启动程序 80
使用 Studio 笔记本 82
自定义 Studio 101
执行常见任务 125
Studio 定价 130
问题排查 130
Amazon SageMaker 上的 rStudio 131
rStudio 组件 132
与 rStudio 工作台的区别 132
在 SageMaker 上管理 rStudio 132
在 Amazon SageMaker 上使用 RStudio 143
笔记本实例 147
AL2 与 AL1 实例 147
AL1 维护阶段计划 147
可用内核 148
迁移到 Amazon Linux 2 149
创建笔记本实例 149
访问笔记本实例 151
更新笔记本实例 152
自定义笔记本实例 152
生命周期配置最佳实践 153
在笔记本实例中安装外部库和内核 154
笔记本实例软件更新 156
使用笔记本控制 Amazon EMR Spark 实例 156
示例笔记本 158
在 Jupyter 经典视图中使用或查看示例笔记本 158
在 Jupyterlab 视图中使用或查看示例笔记本 159
设置笔记本内核 160
Git 回购 161
将 Git 存储库添加到您的亚马逊 SageMaker 账户 161
创建具有关联 Git 存储库的笔记本实例 164
关联 CodeCommit 不同的存储库Amazon笔记本实例的账户 165
在笔记本实例中使用 Git 存储库 166
笔记本实例元数 168
监控 Amazon Jupyter 日志 CloudWatch 日志 168
Autopilot: 自动 ML 170
开始使用 171
示例 171
视频 171
教程 172
创建 Autopilot 实验 172
数据集和问题类型 177
数据集 177
训练和验证数据集 177
问题类型 177
模型支持、指标和验证 178
支持 Autopilot 算法 178

Autopilot 候选项指标 179
自动驾驶仪交叉验证 180
部署模型 180
可以理解 182
生成的模型 182
生成笔记本 186
数据探索报告 186
候选项的笔记本 192
模型洞察 192
配置推理输出 197
回归和分类问题类型的推理容器定义 197
为分类模型选择推理响应 198
配额 199
你可以增加的配额 199
资源配额 200
API 参考 201
SageMaker API 参考 201
亚马逊 SageMaker Python 开发工具包 202
Amazon 命令行界面（CLI） 202
AmazonSDK for Python (Boto) 202
适用于 .NET 的Amazon开发工具包 203
适用于 C++ 的 Amazon 软件开发工具包 203
适用于 Go 的 Amazon 开发工具包 203
Amazon SDK for Java 203
Amazon适用于 JavaScript 的开发工具包 203
Amazon适用于 PHP V3 的开发工具包 204
Amazon适用于 Ruby V3 的开发工具包 204
标记数据 205
标签数据（Ground Truth） 205
您是 Ground Truth 的新用户吗？ 206
开始使用 206
标签图片 210
Text Label 224
为视频和视频帧添加标签 232
标记 3D 点云 260
验证和调整标签 318
创建自定义标记工作流 325
创建标记作业 350
使用输入和输出数据 377
增强的数据标记 436
安全和权限 446
监控标记作业状态 460
创建和管理人力 463
使用 Amazon Mechanical Turk 人力 463
管理供应商人力 466
使用私有人力 467
Crowd HTML 元素参考 485
Crowd SageMaker rowd HTML 元素 486
Augmented AI Crowd HTML 元素 537
准备和分析数据集 542
检测训练前数据偏差 542
亚马逊 SageMaker 澄清偏见和公平性条款 542
示例笔记本 543
测量训练前偏置 544
为训练前数据中的偏差生成报告 SageMaker 工作室 551
使用数据 Wrangler 准备数据 556
首先使用数据 Wrangler 557

导入 566
创建和使用数据 Wrangler 流 601
获取有关数据和数据质量的见解 611
转换数据 623
可视化 658
Export 664
安全和权限 676
发行说明 684
问题排查 685
更新数据 Wrangler 560
关闭数据 Wrangler 690
使用 Studio 笔记本大规模准备数据 691
自带图片 691
从 Studio 发现亚马逊 EMR 集群 692
从 Studio Connect 到 Amazon EMR 集群 695
在 Amazon EMR 中排除和监控工作负载 709
从 Studio 管理亚马逊 EMR 集群 710
所需权限 725
处理数据 728
示例笔记本 728
CloudWatch Logs 和指标 728
使用 Apache Spark 的数据处理 729
运行 Spark 处理 Job 729
使用 scikit-learn 处理数据 729
使用框架处理器处理数据 730
拥抱 Face 框架处理器 731
MXNet 框架处理器 731
PyTorch 框架处理器 732
TensorFlow 框架处理器 733
XGBoost 框架处理器 734
使用您自己的处理代码 735
使用处理容器运行脚本 735
构建您自己的处理容器 736
创建、存储和共享功能 741
如何运作功能商店 741
创建功能组 742
查找、发现和共享功能 742
在线商店中存储的功能的实时推断 742
模型训练和 Batch 推断的离线商店 742
功能数据摄取 742
..... 743
开始使用 Amazon SageMaker Feature Store 743
Feature Store 概念 743
创建要素组 744
向 IAM 角色添加所需的策略 754
将功能商店与 Studio 一起使用 759
数据源和摄取 767
直播摄取 767
使用功能商店的数据 Wrangler 767
Batch 摄取 Spark 连接器设置 769
使用 Athena 和查询功能商店Amazon Glue 772
示例 Athena 查询 772
跨账户离线商店访问 773
第 1 步：在账户 A 中设置离线商店访问角色 773
第 2 步：在账户 B 中设置离线存储 S3 存储桶 774
第 3 步：在账户 A 中设置离线商店 KMS 加密密钥 775
第 4 步：在账号 A 中创建功能组 776

安全性和访问控制 777
使用Amazon KMSAmazon SageMaker Feature Store 的权限 777
授权为您的在线商店使用客户管理密钥 777
使用授权来授权特色商店 779
监控功能商店与之交互Amazon KMS 779
访问在线商店中的数据 779
授权为离线商店使用客户管理密钥 779
配额、命名规则和数据类型 779
限制和配额 779
命名规则 780
数据类型 780
Amazon SageMaker 功能商店离线商店数据格式 780
Amazon SageMaker Feature Store 笔记 781
Feature Store 样本 781
培训 783
选择算法 783
选择算法 784
基本机器学习范例的问题类型 786
使用 内置算法 787
使用强化学习 1757
实验 1758
SageMaker 实验功能 1758
创建实验 1759
查看和比较实验、试验和试验组件 1763
跟踪和比较教程 1766
使用 Studio 搜索实验 1772
清除实验资源 1776
使用控制台和 API 进行搜索 1778
Debugger（调试程序） 1783
调试程序功能 1783
支持的框架和算法 1785
调试程序架构 1786
教程 1788
使用配置调试程序SageMakerPython 开发工具包 1796
使用 SageMaker API 配置调试器 1816
内置规则列表 1826
创建自定义规则 1866
将调试器用于自定义训练容器 1868
对调试器规则执行操作 1872
Studio 中的调试程序 1881
调试程序交互式报告 1896
使用 smDebug 客户端库分析数据 1920
在 TensorBoard 中可视化调试器输出张量 1927
调试器的最佳实践 1930
高级主题和参考 1932
自动模型优化 1939
超参数优化的工作方式 1940
定义指标 1941
定义超参数范围 1942
优化多个算法 1943
例如：超参数优化 Job 1947
提前停止训练作业 1957
运行热启动超参数优化作业 1959
自动模型优化的资源限制 1962
超参数优化的最佳实践 1963
分布式训练 1964
开始使用分布式培训 1965

基本分布式训练概念 1965
高级概念 1966
策略 1967
优化分布式训练 1968
方案 1969
SageMaker 内置分布式培训功能 1971
并行数据训练 1971
模型并行训练 1998
SageMaker 分布式培训笔记本例 2043
澄清偏差检测和模型可解释性 2045
示例笔记本 2045
先决条件 2046
澄清处理任务 2046
入门：澄清容器 2047
配置澄清处理 Job 2048
配置分析 2049
运行 SageMaker 澄清处理任务 2055
检测训练后数据和模型偏差 2058
模型可以解释 2072
排查作业 2079
增量训练 2081
执行增量训练（控制台） 2081
执行增量训练 (API) 2083
访问训练数据 2085
托管的 Spot 训练 2085
使用托管的 Spot 训练 2086
托管的 Spot 训练生命周期 2086
使用检查点 2087
框架和算法 2087
启用检查点 2088
浏览检查点文件 2089
从检查点恢复训练 2089
检查点的注意事项 2090
使用增强清单文件 2090
增强清单文件格式 2090
流式处理增强清单文件数据 2091
使用增强清单文件（控制台） 2091
使用增强清单文件 (API) 2093
使用 CloudWatch 指标监控和分析 2093
定义训练指标 2094
监视训练 Job 指标（CloudWatch 控制台） 2096
监视训练 Job 指标（SageMaker 控制台） 2097
例如：查看训练和验证曲线 2099
部署推理模型 2101
先决条件 2101
您需要做什么？ 2101
管理模型部署 2101
部署自己的推理代码 2102
实时推理 2102
托管选项 2102
自动扩展模型 2144
主机实例存储卷 2158
在生产中安全更新模型 2158
最佳实践 2165
监控模型的数据和模型质量、偏见和可解释性 2167
调用实时端节点 2213
异步推理 2215

工作方式 2215
怎样入门？ 2216
创建、调用和更新异步终端节点 2216
监控异步端节点 2223
检查预测结果 2226
自动缩放异步端点 2228
批量转换 2229
使用批量转换获取大型数据集的推理 2229
加快批量转换作业 2231
使用批量转换测试生产变体 2231
批量转换错误 2231
示例笔记本 2231
将预测结果与输入关联 2231
Batch 存储转换 2236
部署防护机制 2236
如何开始 2237
自动回滚配置和监控 2237
蓝/绿部署 2239
Exclusions 2251
推理成本优化最佳实践 2251
最佳实践 2251
使用模型注册表编录模型 2252
创建模型组 2254
注册版本 2257
模型组和版本 2264
查看模型版本详情 2265
更新模型批准状态 2268
部署模型 2271
部署记录 2275
模型注册表 2278
使用在边缘部署模型 SageMaker Edge 管理器 2280
为什么使用边缘管理器？ 2280
如何工作？ 2280
如何使用 SageMaker Edge Manager？ 2281
入门 2281
设置设备和队列 2294
Package 模型 2300
Edge Manager Agent 2306
管理模型 2317
使用 Neo 优化模型性能 2325
什么是 SageMaker Neo？ 2325
工作方式 2326
示例笔记本 2326
编译模型 2327
云实例 2340
Edge 设备 2365
错误排查 2382
Elastic Inference 2387
EI 的工作原理 2388
选择 EI 加速器类型 2388
在 SageMaker 笔记本实例中使用 EI 2389
在托管终端节点上使用 EI 2389
支持 EI 的框架 2389
将 EI 与 SageMaker 内置算法结合使用 2389
EI 示例笔记本 2389
设置以使用 EI。 2390
将 EI 附加到笔记本实例 2393

具有 Elastic Inference 的终端节点 2395
在 GPU 驱动程序升级过程中尽量减少中断的最佳 2398
当前版本和支持的实例系列 2398
使用 GPU 功能排除模型容器故障 2398
使用不匹配驱动程序版本的最佳实践 2399
配备 SageMaker 的 Docker 容器 2400
场景和指导 2400
Docker 容器基础知识 2400
使用预构建 SageMaker Docker 映像 2401
预置了的深度学习映像 2401
预构建的 Scikit-learn 和 Spark ML 映像 2402
深度图表网络 2403
扩展预构建的容器 2405
调整你自己的 Docker 容器以与 SageMaker 配合使用 2414
单个框架库 2414
SageMaker 训练和推理工具包 2414
调整您自己的训练容器 2416
调整自己的推理容器 2425
使用自己的算法和模型创建容器 2428
使用您自己的训练算法 2429
使用您自己的推理代码 2435
示例笔记本 2442
设置 2443
在 Scikit-Learn 中培训的主机模型 2443
封装 TensorFlow 和 Scikit-Learn 模型供 SageMaker 中使用 2443
在 SageMaker 上训练和部署神经网络 2443
使用管道模式训练 2443
自带 R 模型 2443
扩展预构建的 PyTorch 容器映像 2444
在自定义容器上训练和调试培训作业 2444
故障排除 2444
工作流程 2445
亚马逊 SageMaker 模型构建管道 2445
管道概述 2446
创建和管理管道 2478
项目 2507
为什么选择 MPlops？ 2507
SageMaker 项目 2509
使用项目所需的 SageMaker Studio 权限 2511
创建 mLOPS 项目 2512
模板 2512
查看资源 2519
更新 mLOPS 项目 2520
删除 mLOPS 项目 2521
项目演练 2522
ML 世系跟踪 2527
跟踪实体 2528
SageMaker 创建的实体 2530
手动创建实体 2531
查询谱系实体 2534
跨账户跟踪 2540
Kubernetes 编排 2543
Kubernetes 的 SageMaker 运营商 2544
用于 Kubeflow 管道的 SageMaker 组件 2580
常见问题 2593
Augmented AI 2598
开始使用亚马逊 Augmented AI 2599

Amazon A2I 的核心组件 2600
使用 Augmented AI 的先决条件 2603
教程：从 Amazon A2I 控制台开始使用 2603
教程：开始使用 Amazon A2I API 2608
使用案例和示例 2616
将 SageMaker 笔记本实例与 Amazon A2I Jupyter 笔记本电脑一起使用 2617
配合使用与 Amazon Textract 2617
使用 Amazon Rekognition 2619
结合使用自定义任务类型 2621
创建人工审核工作流程 2622
创建人工审核工作流程（控制台） 2623
创建人工审核工作流程 (API) 2624
亚马逊中用于人 Augmented AI 循环激活条件的 JSON 架构 2628
删除人工审核工作流程 2639
使用控制台或 SageMaker API 删除流定义 2639
创建和启动人工循环 2640
为内置任务类型创建和启动人工循环 2641
为自定义任务类型创建和启动人工循环 2643
后续步骤: 2644
删除人工循环 2644
人类循环数据保留和删除 2645
使用控制台或 Amazon A2I API 停止和删除流定义 2645
创建和管理工作人员任务模板 2647
创建和删除工作人员任务模板 2647
创建自定义 Worker 任务模板 2648
创建良好的工作人员说明 2652
监控和管理您的人工循环 2653
输出数据 2654
从内置任务类型输出数据 2654
输出自定义任务类型的数据 2660
跟踪员工活动 2661
权限和安全性 2662
CORS 权限要求 2663
向用于创建流定义的 IAM 角色添加权限 2663
创建可以调用 Amazon A2I API 操作的 IAM 用户 2664
创建具有调用 Amazon A2I、Amazon Textract 和 Amazon Rekognition API 操作权限的 IAM 用户 2665
启用工作人员任务模板预览 2666
将 Amazon A2I 与Amazon KMS加密存储桶 2666
其他权限和安全资源 2667
CloudWatch Events 2667
将事件从人工循环发送到 CloudWatch Events 2668
设置目标以处理事件 2669
使用人工审核输出 2669
更多信息 2669
API 参考 2669
编程教程 2669
Marketplace 2671
将自己的算法和模型与AmazonMarketplace 2671
创建算法和模型包资源 2671
使用算法和模型包资源 2676
安全性 2684
访问控制 2684
访问控制和 Studio 笔记本电脑 2684
控制对笔记本实例的根访问权限 2685
数据保护 2686
使用加密保护静态数据 2686
使用加密保护传输中的数据 2688

密钥管理 2690
互联网络流量保密性 2690
Identity and Access Management 2690
Audience 2691
使用身份进行身份验证 2691
使用策略管理访问 2692
Amazon 的方式 SageMaker 与 IAM 协同工作 2694
基于身份的策略示例 2696
跨服务困惑副预防 2717
SageMaker 角色 2722
亚马逊 SageMaker API 权限参考 2739
AmazonSageMaker 的托管策略 2754
问题排查 2778
日志记录和监控 2779
合规性验证 2780
故障恢复能力 2780
基础设施安全性 2780
SageMaker 扫描Amazon Web Services Marketplace针对安全漏洞的训练和推理容器 2781
从 VPC 内 Connect 到资源 2781
在互联网免费模式下运行训练和推理容器 2784
连接到 SageMaker 通过 VPC 接口终端节点节点 2785
授予权 SageMaker 访问您的亚马逊 VPC 中的资源 2793
监控 2810
使用 CloudWatch 进行监控 2810
终端节点调用指标 2811
多模型终端节点指标 2812
作业和终端节点指标 2813
Ground Truth 指标 2816
功能商店指标 2818
管道指标 2818
使用 CloudWatch 记录 2820
使用 CloudTrail 记录 SageMaker API 调用 2821
CloudTrail 中的 SageMaker 信息 2821
由自动模型优化执行的操作 2822
了解 SageMaker 日志文件条目 2822
使用 EventBridge 自动执行 2823
训练作业状态更改 2824
超级参数优化作业状态更改 2825
转换作业状态更改 2826
端点状态更改 2827
功能组状态更改 2828
模型包状态更改 2829
管道执行状态更改 2830
管道步骤状态更改 2830
SageMaker 映像状态更改 2831
SageMaker 映像版本状态更改 2831
端节点部署状态更改 2832
API 和开发工具包参考 2835
概览 2835
Amazon SageMaker 的编程模型 2835
Amazon SageMaker 文档历史记录 2837
Amazon词汇表 2839
..... mmdcccxl

什么是 Amazon SageMaker？

Amazon SageMaker 是一项完全托管的机器学习服务。借助 SageMaker，数据科学家和开发人员可以快速、轻松地构建和训练机器学习模型，然后直接将模型部署到生产就绪托管环境中。它提供了一个集成的 Jupyter 编写 Notebook 实例，供您轻松访问数据源以便进行探索和分析，因此您无需管理服务器。此外，它还可以提供常见的机器学习算法，这些算法经过了优化，可以在分布式环境中高效处理非常大的数据。借助对自带算法和框架的原生支持，SageMaker 可以提供灵活并且适合具体工作流程的分布式训练选项。通过在 SageMaker Studio 或 SageMaker 控制台中单击几下鼠标按钮来启动模型，以将模型部署到安全的、可扩展的环境中。训练和托管按使用分钟数计费，没有最低费用，也不需要前期承诺。

本指南包括有关 SageMaker 功能的信息和教程。有关更多信息，请参阅Amazon SageMaker 开发人员资源.

主题

- Amazon SageMaker 功能 (p. 1)
- Amazon SageMaker 定价 (p. 3)
- 您是首次接触 Amazon SageMaker 的用户吗？ (p. 3)

Amazon SageMaker 功能

Amazon SageMaker 包括以下功能：

SageMaker Studio (p. 77)

一体式机器学习环境，您可以在同一应用程序中构建、训练、部署和分析模型。

SageMaker 模型注册表 (p. 2252)

用于部署机器学习模型的版本控制、工件和世系跟踪、审批工作流程以及跨账户支持。

SageMaker 项目 (p. 2507)

使用 SageMaker 项目使用 CI/CD 创建端到端机器学习解决方案。

SageMaker 模型构建管道 (p. 2445)

创建和管理与 SageMaker 作业直接集成的机器学习管道。

SageMaker ML 世系追踪 (p. 2527)

跟踪机器学习工作流的血统。

SageMaker Data Wrangler (p. 556)

在 SageMaker Studio 中导入、分析、准备和特征数据。您可以将 Data Wrangler 集成到机器学习工作流程中，以简化和简化数据预处理和功能工程，只需几乎没有编码即可。您还可以添加您自己的 Python 脚本和转换以自定义数据准备工作流程。

SageMaker Feature Store (p. 741)

集中存储要素和关联元数据，因此可以轻松发现和重复使用要素。您可以创建两种类型的商店，一种是在线或离线商店。在线商店可用于低延迟、实时推理用例，离线商店可用于训练和批量推断。

SageMaker JumpStart (p. 38)

通过精心策划的一键式解决方案、例如笔记本电脑和可部署的预训练模型了解 SageMaker 的特性和功能。您还可以微调模型并部署这些模型。

SageMaker 澄清 (p. 5)

通过检测潜在的偏见来改进机器学习模型，并帮助解释模型所做的预测。

SageMaker Edge 管理器 (p. 2280)

优化边缘设备的自定义模型，创建和管理队列，并在有效的运行时间运行模型。

SageMaker Ground Truth (p. 205)

高质量的训练数据集，它们通过使用工作人员和机器学习来创建标记数据集。

Amazon Augmented AI (p. 2598)

构建人工审查机器学习预测所需的工作流程。Amazon A2I 为所有开发人员带来人工审核，消除了与构建人工审核系统或管理大量人工审核人员相关的千篇一律的繁重工作。

SageMaker Studio 笔记本 (p. 82)

下一代 SageMaker 笔记本电脑，其中包括Amazon IAM Identity Center (successor to Amazon Single Sign-On)(IAM Identity Center) 集成、快速的启动时间和单击共享。

下一代 SageMaker 笔记本，包括较短的启动时间和单击共享。

SageMaker 实验 (p. 1758)

实验管理和跟踪。您可以使用跟踪的数据重新构建实验，在对等方进行的实验基础上逐步构建，并跟踪模型谱系以进行合规性和审核验证。

SageMaker 调试器 (p. 1783)

在整个训练过程中检查训练参数和数据。自动检测常见错误并向用户发出警报，例如参数值变得太大或太小。

SageMaker Autopilot (p. 170)

不了解机器学习的用户可以快速构建分类和回归模型。

SageMaker 模型监控器 (p. 2167)

监控和分析生产中的模型（终端节点），以检测数据偏差和模型质量偏差。

SageMaker Neo (p. 2325)

训练机器学习模型一次，然后在云端和边缘的任何位置运行。

SageMaker Elastic Inference (p. 2387)

提高吞吐量并减少获取实时推理的延迟。

强化学习 (p. 1757)

代理通过其操作获得最大化的长期奖励。

预处理 (p. 728)

分析和预处理数据，处理功能工程问题，并评估模型。

批量转换 (p. 2229)

预处理数据集，在不需要持久终端节点时运行推理，并将输入记录与推理相关联，以帮助解释结果。

Amazon SageMaker 定价

与其他一样Amazon使用 Amazon SageMaker 时，您无需签订合同或承诺最低使用量。有关使用 SageMaker 的成本的更多信息，请参阅SageMaker 定价.

您是首次接触 Amazon SageMaker 的用户吗？

如果您是首次接触 SageMaker 的用户，我们建议您执行以下操作：

1. ReadAmazon SageMaker 的工作原理 (p. 3)— 此部分概述了 SageMaker，说明了主要概念，并介绍了使用 SageMaker 构建 AI 解决方案时涉及的核心组件。我们建议您按所呈现的顺序阅读本主题。
2. 设置 Amazon SageMaker 先决条件 (p. 30)— 此部分介绍如何设置Amazonaccount.
3. Amazon SageMaker Autopilot 通过自动执行机器学习任务以简化机器学习体验。如果您不熟悉 SageMaker，它提供了最简单的学习路径。它还用作一种极佳的 ML 学习工具，它对为每个自动 ML 任务生成的笔记本提供代码可见性。有关其功能的简介，请参阅使用亚马逊自动开发模型 SageMaker Autopilot (p. 170)。为了开始构建、训练和部署机器学习模型，Autopilot 提供了：
   - 示例：使用 Amazon 探索建模 SageMaker Autopilot (p. 171)
   - 视频: 使用 Autopilot 自动执行和探索机器学习过程 (p. 171)
   - 教程: 开始使用 Amazon SageMaker Autopilot (p. 172)
4. 开始使用 Amazon SageMaker (p. 30)— 此部分将指导您使用 SageMaker Studio 或 SageMaker 控制台和 SageMaker API 训练您的第一个模型。您可以使用 SageMaker 提供的训练算法。
5. 探索其他主题 – 根据您的需求，执行以下操作：
   - 提交 Python 代码以使用深度学习框架进行训练— 在 SageMaker 中，您可以使用您自己的训练脚本来训练模型。有关信息，请参阅将 Machine Learning 框架、Python 和 R 与 Amazon SageMaker 结合使用 (p. 11)。
   - 直接从 Apache Spark 中使用 SageMaker— 有关信息，请参阅将 Apache Spark 与 Amazon SageMaker 结合使用 (p. 12).
   - 使用 SageMaker 训练和部署自己的自定义算法— 使用 Docker 包装您的自定义算法，以便在 SageMaker 中训练和/或部署这些算法。要了解 SageMaker 如何与 Docker 容器交互以及 SageMaker 对 Docker 镜像的要求，请参阅将 Docker 容器用于 SageMaker (p. 2400).
6. 查看API 参考— 此部分描述 SageMaker API 操作。

Amazon SageMaker 的工作原理

SageMaker 是一项完全托管的服务，可让您快速、轻松地将基于机器学习的模型集成到您的应用程序中。本部分概述了机器学习并解释 SageMaker 的工作原理。如果您是首次接触 SageMaker 的用户，我们建议您按顺序阅读以下部分：

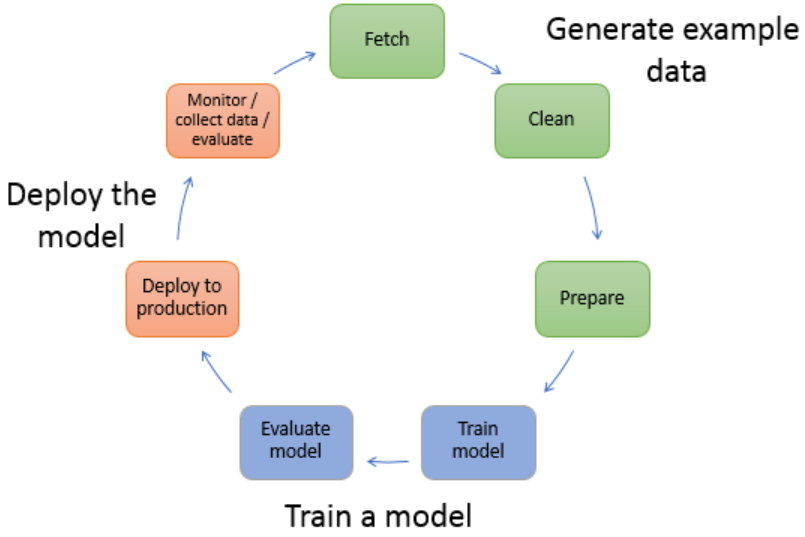
1. 借助 Amazon SageMaker 进行 Machine Learning (p. 3)
2. 探索、分析和处理数据 (p. 5)
3. 使用 Amazon SageMaker 训练模型 (p. 7)
4. 在 Amazon SageMaker 中部署模型 (p. 9)
5. 将 Machine Learning 框架、Python 和 R 与 Amazon SageMaker 结合使用 (p. 11)
6. 开始使用 Amazon SageMaker (p. 30)

借助 Amazon SageMaker 进行 Machine Learning

本节介绍典型的机器学习工作流程并总结了如何使用 Amazon SageMaker 完成这些任务。

在机器学习中，您“指导”计算机进行预测或推理。首先，您使用一种算法和示例数据来训练模型。然后，您将模型集成到应用程序中，以实时且大规模地生成推理。在生产环境中，模型通常从数以百万计的示例数据项目学习，并在数百毫秒至不到 20 毫秒的时间里生成推理。

下图说明了创建机器学习模型的典型工作流程：



如图所示，您通常会执行以下活动：

1. 生成示例数据训练模型需要示例数据。所需数据的类型取决于您希望模型解决的业务问题 (您希望模型生成的推理)。例如，假设您要创建一个模型以预测提供给手写体数字输入图像的编号。要训练此类模型，您需要手写体数字的示例图像。

   数据科学家经常花费大量时间来探索和预处理或“争辩”示例数据，然后再将其用于模型训练。要对数据进行预处理，您通常执行以下操作：

   a. 获取数据— 您可能拥有内部示例数据存储库，或者您可能使用公开可用的数据集。通常，您将一个或多个数据集提取到单个存储库中。

   b. 清理数据要改进模型训练，请检查数据并根据需要进行清理。例如，如果您的数据具有值为 `country name` 和 `United States` 的 `US` 属性，您可能想要编辑数据以保持一致。

   c. 准备或转换数据要提高性能，您可以执行额外的数据转换。例如，您可以选择合并属性。如果您的模型预测到需要为飞机除冰的条件，您可以将温度和湿度属性合并为一种新的属性以获得更好的模型，而不是单独使用这些属性。

   在 SageMaker 中，您对笔记本实例上的 Jupyter 笔记本中的示例数据进行预处理。您使用笔记本获取数据集，探索并准备用于模型训练。有关更多信息，请参阅 探索、分析和处理数据 (p. 5)。有关在中准备数据的更多信息，AmazonMarketplace，请参阅数据准备.

2. 训练模型模型训练包括训练和评估模型，如下所示：

   - 训练模型— 要训练模型，您需要一种算法。您选择的算法取决于许多因素。要想获得快速、现成的解决方案，您可以使用 SageMaker 提供的其中一种算法。关于 SageMaker 提供的算法列表以及相关注意事项，请参阅。使用 Amazon SageMaker 内置算法或预训练模型 (p. 787).

     您还需要适用于训练的计算资源。根据训练数据集的大小以及需要获得结果的速度，您可以使用从单个通用实例到 GPU 实例的分布式集群等各种资源。有关更多信息，请参阅 使用 Amazon SageMaker 训练模型 (p. 7)。

- 评估模型训练模型之后，您对其进行评估，以确定推理的准确性是否可接受。在 SageMaker 中，您可以使用Amazon SDK for Python (Boto)或 SageMaker 提供的高级 Python 库将请求发送到模型以获得推理。

  您使用 SageMaker 笔记本实例中的 Jupyter 笔记本来训练和评估模型。

3. 部署模型— 传统上，您对模型进行重新设计，以将其与应用程序集成并部署。借助 SageMaker 托管服务，您可以独立部署模型，将其与您的应用程序代码解耦。有关更多信息，请参阅 部署推理模型 (p. 2101)。

机器学习是连续的周期。部署模型后，您监控推理，收集"基本实际情况"并评估模型以识别偏差。然后，您可以更新训练数据以包含新收集的基本实际情况，从而提高推理准确性。通过使用新数据集重新训练模型，即可实现这一点。随着越来越多的示例数据变得可用，您继续重新训练模型以提高准确性。

探索、分析和处理数据

使用数据集训练模型之前，数据科学家通常会对其进行探索、分析和预处理。

借助，借助，允许在上轻松、大规模地运行作业，以便对数据进行预处理和后期处理，执行功能设计以及评估模型。当与 SageMaker 提供的其他关键机器学习任务（如训练和托管）结合使用时，具备完全托管的机器学习环境的诸多优势，其中包括所有内置的安全性和合规性支持。借助，您可以灵活地使用内置的数据处理容器，或者使用您自己的容器并提交自定义作业以便在托管基础结构上运行。在提交作业之后，将启 SageMaker 计算实例、处理和分析输入数据，并在完成后释放资源。有关更多信息，请参阅 处理数据 (p. 728)。

- 有关如何运行您自己的数据处理脚本的信息，请参阅。使用 scikit-learn 处理数据 (p. 729).
- 有关如何构建您自己的处理容器以运行脚本的信息，请参阅构建您自己的处理容器（高级方案） (p. 736)。
- 有关如何使用可视化无代码界面执行探索性数据分析 (EDA) 的信息，请参阅使用亚马逊准备 ML 数据 SageMaker DATA Wrangler (p. 556).

什么是 Machine Learning 预测的公平性和模型可解释性？

Amazon SageMaker Clarify 检测潜在偏见并帮助解释模型所做的预测，以此帮助改善您的机器学习（ML）模型。它可以帮助您识别在模型训练期间或模型生产中可能出现的培训前数据和训练后可能出现的各种类型的偏见。SageMaker Clarify 帮助解释这些模型如何使用功能归因方法进行预测。它还监控模型在生产过程中针对偏见或功能归因偏移所做的推断。SageMaker Clary 提供的公平性和可解释性功能提供了帮助的组件Amazon客户构建的机器学习模型不那么偏见且更容易理解。它还提供了一些工具来帮助您生成模型监管报告，供您用来告知风险和合规团队以及外部监管机构。

越来越多地使用机器学习模型和数据驱动系统来帮助跨领域做出决策，例如金融服务、医疗保健、教育和人力资源。机器学习应用程序提供了诸如提高准确性、提高生产力和成本节约等优势，以帮助满足法规要求、改善业务决策，并提供更好的数据科学程序洞察。

- 监管— 在许多情况下，了解机器学习模型为什么进行特定预测，以及它所做的预测是否受到任何偏见的影响，无论是在训练期间还是推断时都是很重要的。最近，决策者、监管者和倡导者提高了人们对 ML 和数据驱动系统构成的道德和政策挑战的认识。特别是，他们对此类系统可能产生的歧视性影响表示关切（例如，无意中将偏见引入自动决策）。
- 业务— 在受监管领域采用 AI 系统需要信任，可以通过对训练模型的行为以及部署的模型如何作出预测提供可靠的解释来建立信任。对于某些具有可靠性、安全性和合规性要求的行业，例如金融服务、人力资

源、医疗保健和自动化运输，模型可解释性可能尤为重要。举一个常见的财务例子，纳入了 ML 模型使用的贷款应用程序可能需要解释这些模型如何向内部贷款官员、客户服务代表和预测员以及最终用户/客户组成的团队作出某些预测。

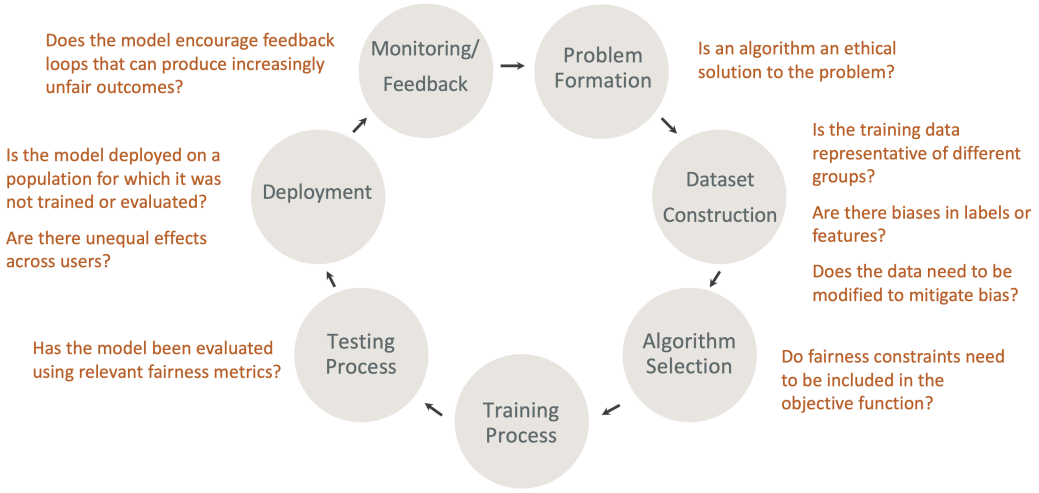
- 数据科学— 数据科学家和机器学习工程师需要工具来生成通过更好的特征工程调试和改进机器学习模型所需的见解，以确定模型是基于嘈杂或不相关的特征进行推断，并了解其模型和故障模式的局限性模特可能会遇到。

有关展示如何构建和构建涉及欺诈性汽车索赔的完整机器学习用例的博客，该用例将 SageMaker Clary 集成到 SageMaker 管道中，请参阅使用构建和构建完整的机器学习生命周期Amazon：端到端到端 Amazon SageMaker演示。本博客讨论了如何评估培训前后的偏见、如何减少偏见以及数据功能如何影响预测。机器学习生命周期中每个任务的相关代码都有链接，包括创建自动化工作流程，该工作流程将 SageMaker Clary 的公平性和可解释性功能集成到 SageMaker Pipeline 中。

在机器学习生命周期中评估公平性和可解性的最佳实践

公平即过程— 偏见和公平的概念在很大程度上取决于申请。此外，要衡量偏见的属性的选择以及偏见指标的选择可能需要以社会、法律和其他非技术考虑为指导。在关键利益相关方（例如产品、政策、法律、工程和 AI/ML 团队以及最终用户和社区）之间建立共识并实现协作，是在实践中成功采用公平意识到的机器学习方法的先决条件。

机器学习生命周期中的设计公平性和可解释性— 您应该考虑机器学习生命周期的每个阶段的公平性和可解释性：问题形成、数据集构建、算法选择、模型训练过程、测试流程、部署和监控/反馈。拥有正确的工具来进行此分析非常重要。为了鼓励参与这些考虑因素，以下是我们建议您在每个阶段提出的几个示例问题。



示例笔记本

Amazon SageMaker Clarify 提供以下示例笔记本：

- 借助 Amazon SageMaker Clary 进行可解释性和偏见检测— 使用 SageMaker Clarine 创建处理作业，以检测偏差并使用功能属性解释模型预测。
- 监控偏差漂移和功能归因漂移 Amazon SageMaker 澄清— 使用 Amazon SageMaker 模型监视器监控一段时间内的偏差漂移和功能归因偏差。
- SageMaker 澄清（自带容器）的公平性和可解释性— 本示例笔记本介绍了解 SageMaker Clarine 所需的关键术语和概念，并引导您完成端到端的数据科学工作流程，演示如何构建自己的模型和容器，这些模型和容器可以与澄清作业无缝协作、使用模型和 SageMaker Clarine 来衡量偏差，解释各种输入功能对模型决策的重要性，然后通过 SageMaker Studio 访问报告（如果您已设置实例）。

- SageMaker Clary 的公平性和可解释性-Spark 分布式处理— 此示例笔记本将引导您了解理解 SageMaker Clarine 所需的关键术语和概念，衡量数据集的训练前偏差和模型的训练后偏差，解释了各种输入要素对模型决策的重要性，并通过 SageMaker Studio 访问报告，如果你已经设置了实例。
- 缓解偏见，训练另一个公正的模型并放入模型注册表— 本笔记本介绍了如何使用 SageMaker Clarine 检测偏差，并通过合成少数过采样技术（SMOTE），训练另一个模型，然后将其连同过程中创建的所有工件的世系一起放入模型注册表中：数据、代码和模型元数据。本笔记本是一个系列的一部分，该系列展示了如何将 SageMaker Clary 集成到 SageMaker 管道中，如使用构建和构建完整的机器学习生命周期Amazon博客。

这些笔记本已验证仅在 Amazon SageMaker Studio 中运行。如果您需要有关如何在 Amazon SageMaker Studio 中打开笔记本的说明，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86). 如果系统会提示您选择内核，请选择Python 3（数据科学）.

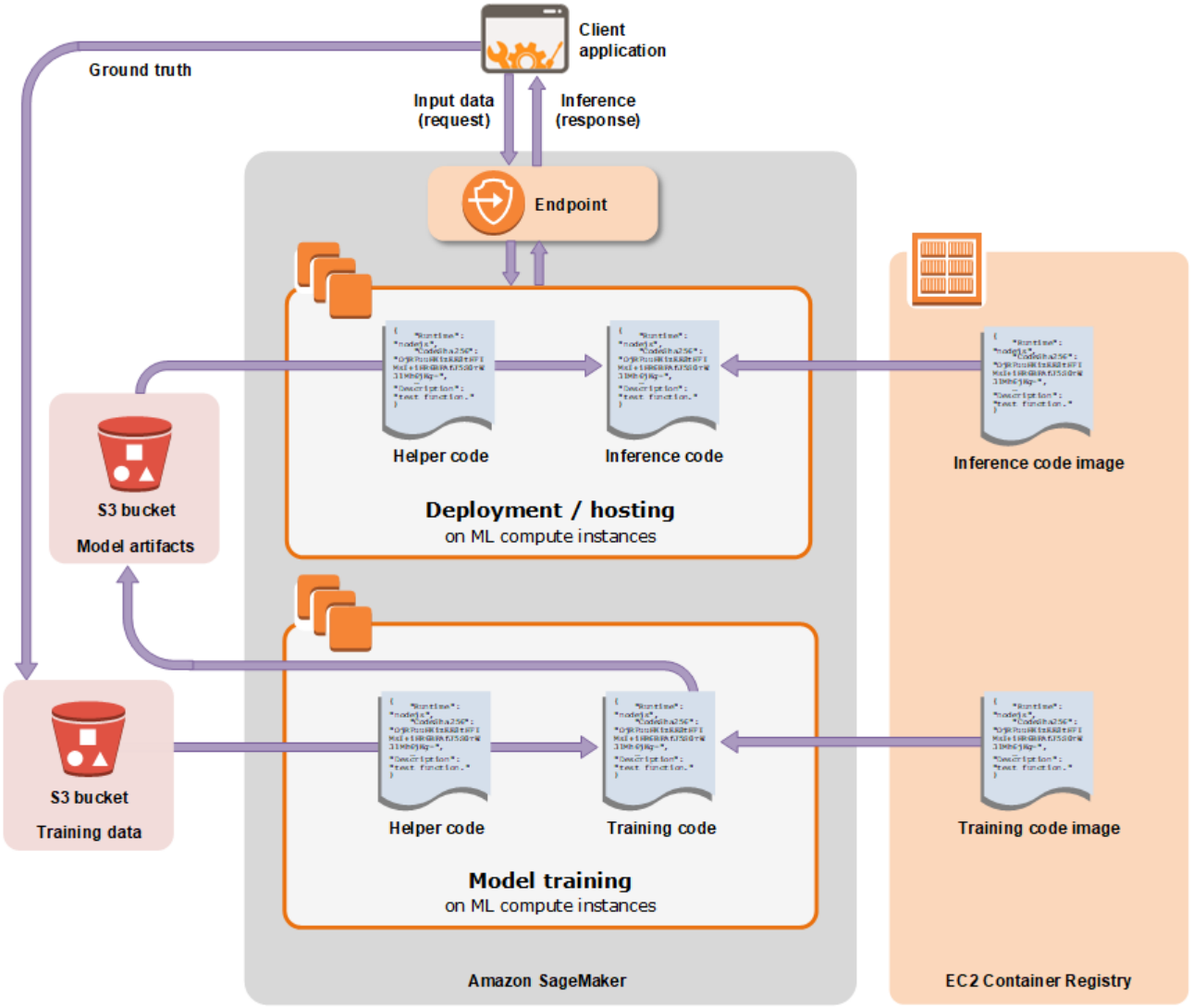
SageMaker 澄清文档指南

在机器学习生命周期的每个阶段：训练模型之前和模型训练之后，都可能会发生偏差并在数据中进行衡量。SageMaker Clarity 可以为训练模型和部署到生产的模型提供模型预测的功能归因说明，在这些模型中，可以监控模型是否有偏离其基线解释性属性的任何偏移。澄清在需要时计算基线。SageMaker Clary 的文档嵌入在相关机器学习阶段的较大的 SageMaker 文档集中，如下所示：

- 有关在用于训练模型之前检测预处理数据中的偏差的更多信息，请参阅检测训练前数据偏差 (p. 542).
- 有关检测训练后数据和模型偏差的更多信息，请参阅使用亚马逊检测训练后数据和模型偏差 SageMaker 澄清 (p. 2058).
- 有关在训练后解释模型预测的与模型无关的功能归因方法的更多信息，请参阅亚马逊 SageMaker 澄清模型的可解释性 (p. 2072).
- 有关监控由于数据偏离用于训练模型的基线而导致的生产模型推断中的偏差的更多信息，请参阅监控生产中模型的偏差漂移 (p. 2179).
- 有关监控要素贡献偏离模型训练期间建立的基线的更多信息，请参阅监控生产中模型的功能归因漂移 (p. 2182).

使用 Amazon SageMaker 训练模型

下图介绍了您如何训练和部署模型。使用 Amazon SageMaker：



标记为 SageMaker 的区域突出显示 SageMaker 两个组件：模型训练和模型部署。

要在 SageMaker 中训练模型，请创建一个训练任务。该训练作业包含以下信息：

- 存储训练数据的 Amazon Simple Storage Service (Amazon S3) 存储桶的 URL。
- 您希望 SageMaker 用于模型训练的计算资源。计算资源是 SageMaker 托管的 ML 计算实例。
- 您要存储作业输出的 S3 存储桶的 URL。
- 存储训练代码的 Amazon Elastic 容器注册表路径。有关更多信息，请参阅 Docker 注册表路径和示例代码 (p. 793)。

对于训练算法，您可以使用以下选项：

- 使用 SageMaker 提供的算法—SageMaker 提供训练算法。如果其中一种算法满足您的需求，它将是实现快速模型训练的不错的现成解决方案。关于 SageMaker 提供的算法列表，请参阅使用 Amazon SageMaker 内置算法或预训练模型 (p. 787). 要尝试使用 SageMaker 提供的算法进行练习，请参阅开始使用 Amazon SageMaker (p. 30).
- 使用 SageMaker 调试程序使用 TensorFlow、PyTorch 和 Apache MXNet 学习框架或 XGBoost 算法时检查整个训练过程中的训练参数和数据。调试程序会自动检测常见错误并向用户发出警报，例如参数值变得

太大或太小。有关使用调试程序的更多信息，请参阅亚马逊SageMakerDebugger (调试程序) (p. 1783)。调试程序示例笔记本可在 Amazon SageMaker 调试程序示例中获取。

- 将 Apache Spark 与 SageMaker 结合使用—SageMaker 提供一个可在 Apache Spark 中使用的库，以便使用 SageMaker 训练模型。使用 SageMaker 提供的库类似于使用 Apache Spark MLLib。有关更多信息，请参阅 将 Apache Spark 与 Amazon SageMaker 结合使用 (p. 12)。
- 提交自定义代码以使用深度学习框架进行您可以提交使用 TensorFlow、PyTorch 或 Apache MXNet 的自定义 Python 代码进行模型训练。有关更多信息，请参阅将 TensorFlow 与 Amazon SageMaker 结合使用 (p. 26)、将 PyTorch 与 Amazon SageMaker 结合使 (p. 22)和将 Apache MXNet 与 Amazon SageMaker 结合使用 (p. 11)。
- 使用您自己的自定义算法将您的代码放在一起作为 Docker 镜像，并在 SageMaker 中指定该镜像的注册表路径。`CreateTrainingJob`API 调用。有关更多信息，请参阅 将 Docker 容器用于 SageMaker (p. 2400)。
-

创建训练任务后，SageMaker 将启动 ML 计算实例，并使用训练代码和训练数据集来训练模型。它将生成的模型构件和其他输出保存在您为该目的指定的 S3 存储桶中。

您可使用 SageMaker 控制台或 API 创建训练任务。有关使用 API 创建训练作业的信息，请参阅 `CreateTrainingJob` API。

当您使用 API 创建训练任务时，默认情况下，SageMaker 会在 ML 计算实例上复制整个数据集。要使 SageMaker 复制各个 ML 计算实例上的数据子集，您必须将`S3DataDistributionType`字段中返回`ShardedByS3Key`. 您可以使用低级开发工具包设置此字段。有关更多信息，请参阅 `S3DataSource` 中的 `S3DataDistributionType`。

> **Important**
>
> 为了防止您的算法容器争用内存，我们为 ML 计算实例上的 SageMaker 关键系统进程预留内存，因此您不能期望看到您的实例类型的所有内存。

在 Amazon SageMaker 中部署模型

在训练机器学习模型之后，您可以使用 Amazon SageMaker 对其进行部署，以便通过以下任何方式获得预测，具体取决于您的使用案例：

- 要获取一次进行一个预测的持久实时终端节点，请使用 SageMaker 实时托管服务。请参阅 实时推理 (p. 2102)。
- 使用无服务器推理，这些工作负载在流量高峰之间存在空闲时间并可以容忍冷启动。请参阅 ???。
- 使用 Amazon SageMaker 异步推理，其有效负载大小高达 1GB、处理时间长以及接近实时延迟要求的请求。请参阅 异步推理 (p. 2215)。
- 要获取整个数据集的预测，请使用 SageMaker 批量转换。请参阅 使用批量转换 (p. 2229)。

SageMaker 还提供了用于在部署机器学习模型时管理资源和优化推理性能的功能：

- 要管理边缘设备上的模型，以便在智能相机、机器人、个人计算机和移动设备等边缘设备队列上优化、保护、监控和维护机器学习模型，请参阅使用在边缘部署模型 SageMaker Edge 管理器 (p. 2280).
- 要优化 Gluon、Keras、MXNet、PyTorch、TensorFlow、TensorFlow Lite 和 ONNX 模型，以在 Ambarella、ARM、Intel、Nvidia、NXP、Qualcomm、TensorFlow 和 Xilinx 基于 Ambarella、ARM、Intel、Nvidia、NXP、Qualcomm、TensorFlow 和 Xilinx 模型。使用 Neo 优化模型性能 (p. 2325).

有关所有部署选项的详细信息，请参阅。部署推理模型 (p. 2101).

验证机器学习模型

训练模型后，对其进行评估，以确定其性能和准确性是否能让您实现业务目标。您可以使用不同的方法生成多个模型并评估每个模型。例如，您可以对每个模型应用不同的业务规则，然后应用各种衡量指标来确定每个模型的适用性。您可以考虑您的模型是否需要比规定的更灵敏（或相反）。

您可以使用历史数据 (离线) 或实时数据评估您的模型：

- 离线测试使用历史（而不是实时）数据向模型发送请求以获取推理。

  将经过训练的模型部署到 alpha 终端节点，并使用历史数据向其发送推理请求。要发送请求，请使用 Amazon SageMaker 笔记本实例中的 Jupyter 笔记本，以及Amazon SDK for Python (Boto)或 SageMaker 提供的高级 Python 库。
- 使用实时数据在线测试—SageMaker 支持使用生产变体对生产中的模型进行 A/B 测试。生产变体是使用相同推理代码并部署在同一个 SageMaker 终端节点上的模型。您配置生产变体，使一小部分实时流量流向要验证的模型。例如，您可能会选择将 10% 的流量发送到模型变体以进行评估。对模型的性能感到满意后，您可以将 100% 的流量传送到更新后的模型。有关在生产中测试模型的示例，请参阅在生产中安全更新模型 (p. 2158)。

有关更多信息，请参阅关于如何评估模型的文章和书籍，例如评估机器学习模型。

离线模型评估的选项包括：

- 使用暂挂集进行验证— 机器学习从业人员通常会留出一部分数据作为 “保留集”。他们不将此数据用于模型训练。

  使用此方法，您可以评估模型在保留集上进行推理的效果如何。然后，您可以评估模型在初次训练中，相对于使用模型记忆，对所学内容进行概念化的效率如何。此验证方法可让您了解模型推断出正确答案的频率。

  此方法在某种程度上类似于给小学学生讲课。首先，您为他们提供一组学习示例，然后测试他们归纳所学知识的能力。借助家庭作业和测试，您提出未包含在初始学习中的问题，并确定他们是否能够有效地归纳。记忆力强的学生可能会记住这些问题，而不是学习规则。

  通常，holdout 数据集占训练数据的 20-30%。

- k 重验证— 在此验证方法中，您将示例数据集拆分为k部分。您将每个部分视为 k 次训练运行的 holdout 设置，而使用其他 k-1 个部分作为该次运行的训练设置。您可以使用类似过程生成 k 个模型，并聚合这些模型以生成最终模型。k 值通常在 5-10 范围内。

监控生产中的模型

将模型部署到生产环境后，请使用 Amazon SageMaker 模型监控器持续监控您的机器学习模型的质量。Amazon SageMaker 模型监控器使您能够设置在模型质量出现偏差（如数据漂移和异常情况）时触发的自动警报触发系统。Amazon CloudWatch Logs 收集监控模型状态的日志文件，并在模型的质量达到您预设的特定阈值时通知。CloudWatch 将日志文件存储到您指定的 Amazon S3 存储桶。通过 Amazon 模型监控器产品及早主动检测模型偏差，您可以迅速采取措施以保持和提高部署的模型质量。

有关 SageMaker 模型监控产品的更多信息，请参阅监控模型的数据和模型质量、偏见和可解释性 (p. 2167).

要使用 SageMaker 开始您的机器学习之旅，请注册Amazon账户在SageMaker.

将 Machine Learning 框架、Python 和 R 与 Amazon SageMaker 结合使用

您可以在 Amazon SageMaker 笔记本内核中直接使用 Python 和 R。还具有一些支持特定框架的内核。一种非常流行的 SageMaker 入门方法是使用Amazon SageMaker Python 开发工具包. 它提供了开源 Python API 和容器，使您可在 SageMaker 中轻松训练和部署模型，并提供了用于多种不同的机器学习和深度学习框架的示例。

有关使用特定框架或如何在 SageMaker 中使用 R 的信息，请参阅以下主题。

语言开发工具包和用户指南：

- Amazon SageMaker Python 开发工具包
- R (p. 23)
- Amazon SageMaker 的 API 参考指南 (p. 2835)

机器学习和深度学习框架指南：

- Apache MXNet (p. 11)
- Apache Spark (p. 12)
- Chainer (p. 20)
- 拥抱面部联接 (p. 21)
- PyTorch (p. 22)
- Scikit-learn (p. 25)
- SparkML Serving (p. 26)
- TensorFlow (p. 26)
- Triton 推理服务器 (p. 27)

将 Apache MXNet 与 Amazon SageMaker 结合使用

您可以通过，使用 SageMaker 使用自定义 MXNet 代码训练和部署模型。这些区域有：Amazon SageMaker Python 开发工具包借助 MXNet 评估程序和模型以及 SageMaker 开源 MXNet 容器，在 SageMaker 中编写和运行 MXNet 脚本变得更加轻松。

您需要做什么？

我想要在 SageMaker 中训练自定义 MXNet 模型。

有关示例 Jupyter 笔记本，请参阅MXNet 示例笔记本在 Amazon SageMaker 示例 GitHub 存储库中。

有关文档，请参阅使用 MXNet 训练模型。

我有一个在 SageMaker 中训练过的 MXNet 模型，我想将它部署到托管终端节点。

有关更多信息，请参阅 。部署 MXNet 模型.

我有一个在 SageMaker 之外训练过的 MXNet 模型，我想将它部署到 SageMaker 终端节点。

有关更多信息，请参阅 。从模型数据部署终端节点.

我想查看 API 文档Amazon SageMaker Python 开发工具包MXNet 类。

有关更多信息，请参阅 。MXNet 类.

我想查找 SageMaker MXNet 容器存储库。

有关更多信息，请参阅 。SageMaker MxNet 容器 GitHub 仓库.

我想查找有关支持的 MXNet 版本的信息AmazonDeep Learning Containers。

有关更多信息，请参阅 。可用的深度学习容器映像.

有关编写 MXNet 脚本模式训练脚本以及将 MXNet 脚本模式评估程序和模型与 SageMaker 结合使用的一般信息，请参阅将 MXNet 与 SageMaker Python 开发工具包结合使用.

将 Apache Spark 与 Amazon SageMaker 结合使用

本节为希望使用 Apache Spark 进行数据预处理和亚马逊的开发人员提供信息。 SageMaker ，用于模型训练和托管。有关 Apache Spark 的受支持版本的信息，请参阅获取 SageMaker Spark页面中的 SageMaker Spark GitHub 存储库。

SageMaker 在 Python 和 Scala 中都提供一个 Apache Spark 库，该库能让您通过使用它在中训练模型。SageMaker 使用`org.apache.spark.sql.DataFrame`Spark 集群中的数据帧。在模型训练后，您还可以使用来托管模型。 SageMaker 托管服务。

这些区域有： SageMaker Spark 库`com.amazonaws.services.sagemaker.sparksdk`，提供以下类等：

- `SageMakerEstimator`— 扩展`org.apache.spark.ml.Estimator`接口。您可以在 SageMaker 中使用此估计程序进行模型训练。
- `KMeansSageMakerEstimator`、`PCASageMakerEstimator`, 和`XGBoostSageMakerEstimator`— 扩展`SageMakerEstimator`类。
- `SageMakerModel`— 扩展`org.apache.spark.ml.Model`类。您可以使用此`SageMakerModel`用于在 SageMaker 中托管模型并获取推理。

与 SageMaker Studio，您可以轻松地连接到 Amazon EMR 集群。有关更多信息，请参阅 。使用 Studio 笔记本大规模准备数据.

下载 SageMaker Spark 库

您可以使用以下选项来下载 Spark Spark Spark Spark Spark SageMaker 提供的库：

- 您可以下载两者的源代码 PySpark 和 Scala 图书馆来自SageMaker Spark GitHub 存储库。
- 对于 Python Spark 库，您有以下附加选项：
  - 使用 pip install：

```
pip install sagemaker_pyspark
```

  - 在笔记本实例中，创建使用`Sparkmagic (PySpark)`或者`Sparkmagic (PySpark3)`内核并连接到远程 Amazon EMR 群集。

    Note

    EMR 集群必须配置有一个 IAM 角色，该角色具有`AmazonSageMakerFullAccess`附加策略。有关为 EMR 群集配置角色的信息，请参阅将针对 Amazon EMR 权限的 IAM 角色配置为Amazon服务中的Amazon EMR 管理指南.

- 您可以从 Maven 获取 Scala 库。通过向 pom.xml 文件添加以下依赖项，将 Spark 库添加到项目中：

```
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>sagemaker-spark_2.11</artifactId>
    <version>spark_2.2.0-1.0</version>
</dependency>
```

将您的 Apache Spark 应用程序与 SageMaker 集成

以下是将 Apache Spark 应用程序与 SageMaker 集成的步骤的高度概括。

1. 继续使用您熟悉的 Apache Spark 进行数据预处理。您的数据集仍然是 Spark 集群中的 DataFrame。将您的数据加载到 DataFrame 中并对其进行预处理，从而获得一个 features 为 org.apache.spark.ml.linalg.Vector 的 Doubles 列，以及一个可选的值为 label 类型的 Double 列。
2. 使用中的估计器 SageMaker Spark 库来训练您的模型。例如，如果您选择提供的 k-means 算法。SageMaker 对于模型训练，请调用KMeansSageMakerEstimator.fit方法。

   提供您的 DataFrame 作为输入。评估程序返回一个 SageMakerModel 对象。

   > Note
   >
   > SageMakerModel 扩展 org.apache.spark.ml.Model。

   fit 方法执行以下操作：

   a. 转换输入DataFrame通过选择以下方式转换为 protobuf 格式features和label来自输入的列DataFrame并将 protobuf 数据上载到 Amazon S3 存储桶。protobuf 格式对于 SageMaker 中的模型训练是有效的。

   b. 于中开始模型训练 SageMaker 发送 SageMaker CreateTrainingJob请求. 模特训练完成后，SageMaker 将模型构件保存到 S3 存储桶。

      SageMaker 采用您为模型训练指定的 IAM 角色，以代表您执行任务。例如，它使用角色从 S3 存储桶读取训练数据并将模型构件写入存储桶。

   c. 创建并返回一个 SageMakerModel 对象。构造函数执行以下任务，这些任务与将模型部署到 SageMaker 中相关。

      i. 发送CreateModel向 SageMaker 发出请求。
      ii. 发送CreateEndpointConfig向 SageMaker 发出请求。
      iii. 发送CreateEndpoint向 SageMaker 发出请求，该请求随后会启动指定的资源，并在其上托管模型。
3. 您可以从中托管的模型获取推理 SageMaker 使用SageMakerModel.transform.

   提供具有特征的输入 DataFrame 作为输入。transform 方法将其转换为一个包含推理的 DataFrame。在内部，transform方法将请求发送到InvokeEndpoint SageMaker API 来获取推论。transform 方法将推理附加到输入 DataFrame 中。

示例 1：使用 Amazon SageMaker 用于训练和推理的 Apache Spark

主题

- 在 Amazon 上使用自定义算法进行模型训练和托管 SageMaker 与 Apache Spark 结合使用 (p. 18)
- 在 Spark 管道中使用 SageMakerEstimator (p. 18)

亚马逊 SageMaker 提供一个 Apache Spark 库 (在 Python 和 Scala 中)，您可以使用它将您的 Apache Spark 应用程序与 SageMaker 集成。例如，您可以使用 Apache Spark 进行数据预处理，并且 SageMaker ，用于模型训练和托管。有关更多信息，请参阅 将 Apache Spark 与 Amazon SageMaker 结合使用 (p. 12)。本节提供的示例代码使用 SageMaker 提供的 Apache Spark Scala 库，用以在中训练模型。 SageMaker 使用`DataFrame`Spark 集群中。此示例还使用来托管生成的模型构件。 SageMaker 托管服务。具体来说，此示例执行以下操作：

- 对数据使用 `KMeansSageMakerEstimator` 调整 (或训练) 模型

  因为此示例使用提供的 k-means 算法。 SageMaker 为训练模型，请使用`KMeansSageMakerEstimator`. 您使用手写体单个位数数字的图像 (来自 MNIST 数据集) 来训练模型。您提供图像作为输入 `DataFrame`。为方便起见， SageMaker 在 S3 存储桶中提供此数据集。

  作为响应，评估程序返回一个 `SageMakerModel` 对象。

- 使用经过训练的 `SageMakerModel` 获取推理

  要从在 SageMaker 中托管的模型获取推理，您可以调用`SageMakerModel.transform`方法。您传递一个 `DataFrame` 作为输入。该方法将输入 `DataFrame` 转换为另一个包含从模型中获得的推理的 `DataFrame`。

  对于手写体单个位数数字的给定输入图像，该推理标识图像所属的聚类。有关更多信息，请参阅 K-Means 算法 (p. 1637)。

这是示例代码：

```
import org.apache.spark.sql.SparkSession
import com.amazonaws.services.sagemaker.sparksdk.IAMRole
import com.amazonaws.services.sagemaker.sparksdk.algorithms
import com.amazonaws.services.sagemaker.sparksdk.algorithms.KMeansSageMakerEstimator

val spark = SparkSession.builder.getOrCreate

// load mnist data as a dataframe from libsvm
val region = "us-east-1"
val trainingData = spark.read.format("libsvm")
  .option("numFeatures", "784")
  .load(s"s3://sagemaker-sample-data-$region/spark/mnist/train/")
val testData = spark.read.format("libsvm")
  .option("numFeatures", "784")
  .load(s"s3://sagemaker-sample-data-$region/spark/mnist/test/")

val roleArn = "arn:aws:iam::account-id:role/rolename"

val estimator = new KMeansSageMakerEstimator(
  sagemakerRole = IAMRole(roleArn),
  trainingInstanceType = "ml.p2.xlarge",
  trainingInstanceCount = 1,
  endpointInstanceType = "ml.c4.xlarge",
  endpointInitialInstanceCount = 1)
  .setK(10).setFeatureDim(784)
```

```
// train
val model = estimator.fit(trainingData)

val transformedData = model.transform(testData)
transformedData.show
```

该代码执行以下操作：

- 将 MNIST 数据集从 SageMaker 提供的 S3 存储桶 (awsai-sparksdk-dataset) 变成 SparkDataFrame(mnistTrainingDataFrame):

```
// Get a Spark session.

val spark = SparkSession.builder.getOrCreate

// load mnist data as a dataframe from libsvm
val region = "us-east-1"
val trainingData = spark.read.format("libsvm")
  .option("numFeatures", "784")
  .load(s"s3://sagemaker-sample-data-$region/spark/mnist/train/")
val testData = spark.read.format("libsvm")
  .option("numFeatures", "784")
  .load(s"s3://sagemaker-sample-data-$region/spark/mnist/test/")

val roleArn = "arn:aws:iam::account-id:role/rolename"
trainingData.show()
```

show 方法显示数据帧中的前 20 行：

```
+-----+--------------------+
|label|            features|
+-----+--------------------+
|  5.0|(784,[152,153,154...|
|  0.0|(784,[127,128,129...|
|  4.0|(784,[160,161,162...|
|  1.0|(784,[158,159,160...|
|  9.0|(784,[208,209,210...|
|  2.0|(784,[155,156,157...|
|  1.0|(784,[124,125,126...|
|  3.0|(784,[151,152,153...|
|  1.0|(784,[152,153,154...|
|  4.0|(784,[134,135,161...|
|  3.0|(784,[123,124,125...|
|  5.0|(784,[216,217,218...|
|  3.0|(784,[143,144,145...|
|  6.0|(784,[72,73,74,99...|
|  1.0|(784,[151,152,153...|
|  7.0|(784,[211,212,213...|
|  2.0|(784,[151,152,153...|
|  8.0|(784,[159,160,161...|
|  6.0|(784,[100,101,102...|
|  9.0|(784,[209,210,211...|
+-----+--------------------+
only showing top 20 rows
```

在每行中：

- label 列标识图像的标签。例如，如果手写数字的图像是数字 5，则标签值为 5。
- features 列存储 org.apache.spark.ml.linalg.Vector 值的向量 (Double)。这是手写数字的 784 个特征。(每个手写数字都是一个 28x28 像素的图像，从而形成 784 个特征。)

- 创建 SageMaker 估算器 (`KMeansSageMakerEstimator`)

  这些区域有：`fit`此估计程序的方法使用提供的 k-means 算法。 SageMaker 使用输入训练模型`DataFrame`. 作为响应，它会返回一个 `SageMakerModel` 对象，此对象可用于获取推理。

  Note

  这些区域有：`KMeansSageMakerEstimator`扩展 SageMaker`SageMakerEstimator`，而后者会扩展 Apache Spark`Estimator`.

```
val estimator = new KMeansSageMakerEstimator(
  sagemakerRole = IAMRole(roleArn),
  trainingInstanceType = "ml.p2.xlarge",
  trainingInstanceCount = 1,
  endpointInstanceType = "ml.c4.xlarge",
  endpointInitialInstanceCount = 1)
  .setK(10).setFeatureDim(784)
```

  构造函数参数提供用于训练模型并将其部署到 SageMaker 的信息：

  - `trainingInstanceType`和`trainingInstanceCount`— 标识要用于模型训练的 ML 计算实例的类型和数量。

  - `endpointInstanceType`标识在 SageMaker 中托管模型时要使用的 ML 计算实例类型。默认情况下，采用一个 ML 计算实例。

  - `endpointInitialInstanceCount`— 标识最初支持在 SageMaker 中托管模型的终端节点的 ML 计算实例数量。

  - `sagemakerRole`—SageMaker 采用此 IAM 角色代表您执行任务。例如，对于模型训练，它从 S3 读取数据并将训练结果（模型构件）写入到 S3 中。

    Note

    此示例隐式创建 SageMaker 客户端。要创建此客户端，您必须提供凭证。API 使用这些凭证来验证对 SageMaker 的请求。例如，它使用凭证来验证对请求进行身份验证，用于创建训练作业和 API 调用，以使用 SageMaker 托管服务。

  - 创建 `KMeansSageMakerEstimator` 对象后，您可以设置以下参数，以便用在模型训练中：
    - k-means 算法在模型训练过程中应创建的集群数量。您可以指定 10 个聚类，每个数字（0 到 9）一个。
    - 标识每个输入图像都有 784 个特征 (每个手写数字都是 28x28 像素的图像，从而形成 784 个特征)。

- 调用评估程序 `fit` 方法

```
// train
val model = estimator.fit(trainingData)
```

  您将输入 `DataFrame` 作为一个参数传递。模型进行了训练模型的所有工作，并将其部署到 SageMaker。有关更多信息，请参阅将您的 Apache Spark 应用程序与 SageMaker 集成 (p. 13)。作为响应，您将获得`SageMakerModel`对象，您可以使用它从在 SageMaker 中部署的模型获取推理。

您只提供输入 DataFrame。您不需要指定用于模型训练的 k-means 算法的注册表路径，因为 KMeansSageMakerEstimator 知道它。

- 调用SageMakerModel.transform方法以从在 SageMaker 中部署的模型获取推理。

  transform 方法采用 DataFrame 作为输入，传输它，并返回另一个包含从模型获得的推理的 DataFrame。

```
val transformedData = model.transform(testData)
transformedData.show
```

  为简单起见，我们使用在本例中用于模型训练的相同 DataFrame 作为 transform 方法的输入。transform 方法执行以下操作：

  - 序列化features输入中的列DataFrame转到 protobuf 然后将其发送到 SageMaker 用于推理的终点。
  - 将 protobuf 响应反序列化成转换的 distance_to_cluster 中的两个附加列 (closest_cluster 和 DataFrame)。

  show 方法获取对输入 DataFrame 中前 20 行的推理：

```
+-----+--------------------+------------------+---------------+
|label|            features|distance_to_cluster|closest_cluster|
+-----+--------------------+------------------+---------------+
|  5.0|(784,[152,153,154...|  1767.897705078125|            4.0|
|  0.0|(784,[127,128,129...|  1392.157470703125|            5.0|
|  4.0|(784,[160,161,162...| 1671.5711669921875|            9.0|
|  1.0|(784,[158,159,160...| 1182.6082763671875|            6.0|
|  9.0|(784,[208,209,210...| 1390.4002685546875|            0.0|
|  2.0|(784,[155,156,157...|  1713.988037109375|            1.0|
|  1.0|(784,[124,125,126...| 1246.3016357421875|            2.0|
|  3.0|(784,[151,152,153...|  1753.229248046875|            4.0|
|  1.0|(784,[152,153,154...|  978.8394165039062|            2.0|
|  4.0|(784,[134,135,161...|  1623.176513671875|            3.0|
|  3.0|(784,[123,124,125...|  1533.863525390625|            4.0|
|  5.0|(784,[216,217,218...|  1469.357177734375|            6.0|
|  3.0|(784,[143,144,145...|  1736.765869140625|            4.0|
|  6.0|(784,[72,73,74,99...|   1473.69384765625|            8.0|
|  1.0|(784,[151,152,153...|    944.88720703125|            2.0|
|  7.0|(784,[211,212,213...| 1285.9071044921875|            3.0|
|  2.0|(784,[151,152,153...| 1635.0125732421875|            1.0|
|  8.0|(784,[159,160,161...| 1436.3162841796875|            6.0|
|  6.0|(784,[100,101,102...| 1499.7366943359375|            7.0|
|  9.0|(784,[209,210,211...| 1364.6319580078125|            6.0|
+-----+--------------------+------------------+---------------+
```

  您可以如下所示解释数据：

  - 具有 label 5 的手写数字属于集群 4 (closest_cluster)。
  - 具有 label 0 的手写数字属于集群 5。
  - 具有 label 4 的手写数字属于集群 9。
  - 具有 label 1 的手写数字属于集群 6。

有关如何运行这些示例的更多信息，请参阅 GitHub 上的 https://github.com/aws/sagemaker-spark/blob/master/README.md。

在 Amazon 上使用自定义算法进行模型训练和托管 SageMaker 与 Apache Spark 结合使用

In示例 1：使用 Amazon SageMaker 用于训练和推理的 Apache Spark (p. 13)，您使用kMeansSageMakerEstimator因为此示例使用亚马逊提供的 k-means 算法。 SageMaker ，用于模型训练。您可以选择使用自己的自定义算法进行模型训练。假设您已经创建了 Docker 图像，您可以创建自己的图像。SageMakerEstimator然后为自定义映像指定 Amazon Elastic 容器注册表路径。

以下示例显示如何从 SageMakerEstimator 中创建 KMeansSageMakerEstimator。在新的评估程序中，您可以显式指定训练和推理代码图像的 Docker 注册表路径。

```
import com.amazonaws.services.sagemaker.sparksdk.IAMRole
import com.amazonaws.services.sagemaker.sparksdk.SageMakerEstimator
import
 com.amazonaws.services.sagemaker.sparksdk.transformation.serializers.ProtobufRequestRowSerializer
import
 com.amazonaws.services.sagemaker.sparksdk.transformation.deserializers.KMeansProtobufResponseRowDeseri

val estimator = new SageMakerEstimator(
  trainingImage =
    "811284229777.dkr.ecr.us-east-1.amazonaws.com/kmeans:1",
  modelImage =
    "811284229777.dkr.ecr.us-east-1.amazonaws.com/kmeans:1",
  requestRowSerializer = new ProtobufRequestRowSerializer(),
  responseRowDeserializer = new KMeansProtobufResponseRowDeserializer(),
  hyperParameters = Map("k" -> "10", "feature_dim" -> "784"),
  sagemakerRole = IAMRole(roleArn),
  trainingInstanceType = "ml.p2.xlarge",
  trainingInstanceCount = 1,
  endpointInstanceType = "ml.c4.xlarge",
  endpointInitialInstanceCount = 1,
  trainingSparkDataFormat = "sagemaker")
```

在该代码中，SageMakerEstimator 构造函数中的参数包括：

- trainingImage— 标识包含自定义代码的训练图像的 Docker 注册表路径。
- modelImage— 标识包含推理代码的图像的 Docker 注册表路径。
- requestRowSerializer— 实现com.amazonaws.services.sagemaker.sparksdk.transformation.RequestRowSerializer.

  此参数序列化输入中的行DataFrame将其发送到中托管的模型 SageMaker 用于推理。
- responseRowDeserializer— 实现

  com.amazonaws.services.sagemaker.sparksdk.transformation.ResponseRowDeserializer.

  此参数将在 SageMaker 中托管的响应反序列化回DataFrame.
- trainingSparkDataFormat— 指定 Spark 在从上载训练数据时使用的数据格式。DataFrame到 S3。例如，"sagemaker" 用于 protobuf 格式，"csv" 用于逗号分隔值，"libsvm" 用于 LibSVM 格式。

您可以实施自己的 RequestRowSerializer 和 ResponseRowDeserializer 以序列化或反序列化您的推理代码支持的数据格式中的行，例如 .libsvm 或 .csv。

在 Spark 管道中使用 SageMakerEstimator

您可以在 org.apache.spark.ml.Estimator 管道中使用 org.apache.spark.ml.Model 评估程序和 SageMakerEstimator 模型，以及 SageMakerModel 评估程序和 org.apache.spark.ml.Pipeline 模型，如以下示例所示：

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.feature.PCA
import org.apache.spark.sql.SparkSession
import com.amazonaws.services.sagemaker.sparksdk.IAMRole
import com.amazonaws.services.sagemaker.sparksdk.algorithms
import com.amazonaws.services.sagemaker.sparksdk.algorithms.KMeansSageMakerEstimator

val spark = SparkSession.builder.getOrCreate

// load mnist data as a dataframe from libsvm
val region = "us-east-1"
val trainingData = spark.read.format("libsvm")
  .option("numFeatures", "784")
  .load(s"s3://sagemaker-sample-data-$region/spark/mnist/train/")
val testData = spark.read.format("libsvm")
  .option("numFeatures", "784")
  .load(s"s3://sagemaker-sample-data-$region/spark/mnist/test/")

// substitute your SageMaker IAM role here
val roleArn = "arn:aws:iam::account-id:role/rolename"

val pcaEstimator = new PCA()
  .setInputCol("features")
  .setOutputCol("projectedFeatures")
  .setK(50)

val kMeansSageMakerEstimator = new KMeansSageMakerEstimator(
  sagemakerRole = IAMRole(integTestingRole),
  requestRowSerializer =
    new ProtobufRequestRowSerializer(featuresColumnName = "projectedFeatures"),
  trainingSparkDataFormatOptions = Map("featuresColumnName" -> "projectedFeatures"),
  trainingInstanceType = "ml.p2.xlarge",
  trainingInstanceCount = 1,
  endpointInstanceType = "ml.c4.xlarge",
  endpointInitialInstanceCount = 1)
  .setK(10).setFeatureDim(50)

val pipeline = new Pipeline().setStages(Array(pcaEstimator, kMeansSageMakerEstimator))

// train
val pipelineModel = pipeline.fit(trainingData)

val transformedData = pipelineModel.transform(testData)
transformedData.show()
```

参数 `trainingSparkDataFormatOptions` 配置 Spark 以将“projectedFeatures”列序列化成 protobuf 以用于模型训练。此外，Spark 在默认情况下将“label”列序列化成 protobuf。

由于我们想要使用“projectedFeatures”列进行推理，因此我们将列名传递到 `ProtobufRequestRowSerializer`。

以下示例显示了一个转换的 `DataFrame`：

```
+-----+--------------------+--------------------+-------------------+---------------+
|label|            features|   projectedFeatures|distance_to_cluster|closest_cluster|
+-----+--------------------+--------------------+-------------------+---------------+
|  5.0|(784,[152,153,154...|[880.731433034386...|     1500.470703125|            0.0|
|  0.0|(784,[127,128,129...|[1768.51722024166...|      1142.18359375|            4.0|
|  4.0|(784,[160,161,162...|[704.949236329314...|  1386.246826171875|            9.0|
|  1.0|(784,[158,159,160...|[-42.328192193771...| 1277.0736083984375|            5.0|
|  9.0|(784,[208,209,210...|[374.043902028333...|   1211.00927734375|            3.0|
|  2.0|(784,[155,156,157...|[941.267714528850...|  1496.157958984375|            8.0|
|  1.0|(784,[124,125,126...|[30.2848596410594...| 1327.6766357421875|            5.0|
|  3.0|(784,[151,152,153...|[1270.14374062052...| 1570.7674560546875|            0.0|
```

```
|  1.0|(784,[152,153,154...|[-112.10792566485...|     1037.568359375|            5.0|
|  4.0|(784,[134,135,161...|[452.068280676606...| 1165.1236572265625|            3.0|
|  3.0|(784,[123,124,125...|[610.596447285397...|  1325.953369140625|            7.0|
|  5.0|(784,[216,217,218...|[142.959601818422...| 1353.4930419921875|            5.0|
|  3.0|(784,[143,144,145...|[1036.71862533658...| 1460.4315185546875|            7.0|
|  6.0|(784,[72,73,74,99...|[996.740157435754...| 1159.8631591796875|            2.0|
|  1.0|(784,[151,152,153...|[-107.26076167417...|   960.963623046875|            5.0|
|  7.0|(784,[211,212,213...|[619.771820430940...|   1245.13623046875|            6.0|
|  2.0|(784,[151,152,153...|[850.152101817161...|  1304.437744140625|            8.0|
|  8.0|(784,[159,160,161...|[370.041887230547...| 1192.4781494140625|            0.0|
|  6.0|(784,[100,101,102...|[546.674328209335...|    1277.0908203125|            2.0|
|  9.0|(784,[209,210,211...|[-29.259112927426...| 1245.8182373046875|            6.0|
+-----+--------------------+--------------------+-------------------+---------------+
```

开发工具包示例 使用 Amazon SageMaker 与 Apache Spark 结合使用

以下列表是可用示例的子集。访问网站示例查看更多信息。

- `sagemaker-spark`：SageMaker 的 Spark 库
- SageMaker PySpark K-均值聚类 MNIST 示例
- 使用 Apache Spark 和 SageMaker 处理的分布式数据处理

> **Note**
>
> 要在笔记本实例上运行笔记本，请参阅示例笔记本 (p. 158). 要在 Studio 上运行笔记本电脑，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86).

将 Chainer 与 Amazon SageMaker 结合使用

您可以使用 SageMaker 通过自定义 Chainer 代码训练和部署模型。SageMaker Python 开发工具包 Chainer 评估程序和模型以及 SageMaker 开源 Chainer 容器使得在 SageMaker 中编写和运行 Chainer 脚本更加轻松。

您需要做什么？

我想要在 SageMaker 中训练自定义 Chainer 模型。

有关示例 Jupyter 笔记本，请参阅Chainer 示例笔记本在 Amazon SageMaker 示例 GitHub 存储库中。

有关文档，请参阅使用 Chainer 训练模型。

我有一个在 SageMaker 中训练过的 Chainer 模型，我想将它部署到托管终端节点。

有关更多信息，请参阅 。部署 Chainer 模型.

我有一个在 SageMaker 之外训练过的 Chainer 模型，我想将它部署到 SageMaker 终端节点。

有关更多信息，请参阅 。从模型数据部署终端节点.

我想查看 API 文档Amazon SageMaker Python 开发工具包Chainer 类。

有关更多信息，请参阅 。Chainer 类.

我想查找有关 SageMaker Chainer 容器的信息。

有关更多信息，请参阅 。SageMaker Chainer 容器 GitHub 仓库.

有关支持的 Chainer 版本的信息，以及有关编写 Chainer 训练脚本以及将 Chainer 评估程序和模型与 SageMaker 结合使用的一般信息，请参阅将 Chainer 与 SageMaker Python 开发工具包结合使用.

与 Amazon SageMaker 一起使用拥抱脸

Amazon SageMaker 使客户能够在 SageMaker 上使用用于自然语言处理的拥抱人脸模型 (NLP) 来训练、微调和运行推理。你可以使用 Hugge Face 进行训练和推理。这个功能可以通过开发 Hugge Face 获得AmazonDeep Learning 容器. 这些容器包括 Hugging Face 变形器、Tokenizer 和数据集库，它们允许您将这些资源用于训练和推理工作。有关可用的 Deep Learning Containers 映像的列表，请参阅可用的 Deep Learning Containers 映像. 这些 Deep Learning Containers 映像通过安全补丁进行维护和定期更新。

要将拥抱面部 Deep Learning Containers 与 SageMaker Python SDK 结合使用进行培训，请参阅拥抱面孔 SageMaker 估算器. 使用拥抱人脸估算器，您可以像使用任何其他 SageMaker 估计器一样使用拥抱脸模型。但是，使用 SageMaker Python SDK 是可选的。您还可以通过Amazon CLI和Amazon SDK for Python (Boto3).

有关 Hugging Face 及其中提供的模型的更多信息，请参阅拥抱 Face 文档.

培训

要进行训练，您可以使用 Hugging Face 中可用的数千种模型中的任意一种，并通过额外的培训来根据您的具体使用案例对它们进行微调。使用 SageMaker，您可以使用标准培训或利用SageMaker 分布式数据和模型并行培训. 与其他使用自定义代码的 SageMaker 培训作业一样，您可以通过将指标定义传递给 SageMaker Python SDK 来捕获自己的指标，如中所示定义训练指标（SageMaker Python 开发工具包）. 然后可以通过以下方式访问捕获CloudWatch而且作为熊猫`DataFrame`通过培训职位分析方法。一旦你的模型经过训练和微调，你可以像任何其他模型一样使用它来运行推理作业。

如何使用拥抱人脸估算器进行训练

您可以使用 SageMaker Python SDK 实现拥抱人脸估算器来进行培训作业。SageMaker Python SDK 是一个开源库，用于在 SageMaker 上训练和部署机器学习模型。有关 Hide Face 的更多信息，请参阅SageMaker Python SDK 文档。

借助 SageMaker Python 开发工具包，您可以在以下环境中使用 Hide Face 开发工具包运行训练作业：

- SageMaker Studio：Amazon SageMaker Studio 是第一个用于机器学习 (ML) 的完全集成开发环境 (IDE)。SageMaker Studio 提供了一个基于 Web 的单一可视化界面，您可以在其中执行准备、构建、训练和调整、部署和管理模型所需的所有机器学习开发步骤。有关在 Studio 中使用 Jupyter 笔记本的信息，请参阅使用 Amazon SageMaker Studio 笔记本.
- SageMaker 笔记本实例：Amazon SageMaker 笔记本实例是一个运行 Jupyter 笔记本应用程序的机器学习 (ML) 计算实例。借助此应用程序，您可以在笔记本实例中运行 Jupyter Notebook，以准备和处理数据，编写代码以训练模型，部署模型到 SageMaker 托管，以及测试和验证模型，而无需调试器、模型监控和基于 Web 的 IDE 等 SageMaker Studio 功能。
- 截图等)：如果你有连接Amazon并拥有适当的 SageMaker 权限，您可以在本地使用 SageMaker Python SDK 为 SageMaker 中的 Hugge Face 启动远程训练和推理作业Amazon. 这适用于您的本地计算机，以及其他Amazon具有连接 SageMaker Python SDK 和适当权限的服务。

推理

为了进行推理，您可以使用训练有素的拥抱人脸模型或其中一个预先训练的 Hugging Face 模型来部署 SageMaker 的推理作业。通过这种协作，您只需要一行代码即可使用 SageMaker 部署训练过的模型和预训练模型。您还可以运行推理作业，而无需编写任何自定义推理代码。使用自定义推理代码，您可以通过提供自己的 Python 脚本来自定义推理逻辑。

如何使用 Hugging Face Deep Learning Containers 部署推理作业

您有两个选项可以使用 SageMaker 运行推理。你可以使用你训练的模型来运行推理，也可以部署预先训练的 Hugging Face 模型。

- 使用训练有素的模型运行推理：用自己训练的模型运行推理有两种选择。您可以使用使用 SageMaker Hugging Face Deep Learning Containers 使用现有的拥抱人脸模型训练的模型运行推理，也可以自带现有的拥抱人脸模型并使用 SageMaker 进行部署。当您使用使用 SageMaker Hugge Face Estemator 训练的模型运行推理时，您可以在训练完成后立即部署模型，也可以将训练过的模型上传到 Amazon S3 存储桶，然后在以后运行推理时将其摄取。如果您自带现有的 Hugging Face 模型，则必须将训练过的模型上传到 Amazon S3 存储桶，然后在运行推理时摄取该存储桶，如中所示部署你的拥抱脸变压器作为推理示例.
- 使用预先训练的 HuggingFace 模型运行推理：您可以使用成千上万个预训练的 HugGING Face 模型之一来运行推理工作，而无需额外的培训。要运行推理，您可以从列表中选择预训练的模型拥抱 Face 模型，如中所述部署预先训练的拥抱脸变压器以获取推理示例.

您需要做什么？

Hugging Face 笔记本存储库中的以下 Jupyter 笔记本说明了如何在各种使用案例中将 Hugging Face Deep Learning Containers 与 SageMaker 结合使用。

我想在 PyTorch 中使用 SageMaker 中的 Hugge Face 训练和部署文本分类模型。

有关示例 Jupyter 笔记本，请参阅PyTorch 入门.

我想在 SageMaker 中使用 TensorFlow 中的 Hugge Face 训练和部署文本分类模型。

有关示例 Jupyter 笔记本，请参阅TensorFlow 入门示例.

我想使用 Hugging Face 和 SageMaker 分布式运行具有数据并行性的分布式训练。

有关示例 Jupyter 笔记本，请参阅分布式训练示例.

我想使用 Hugging Face 和 SageMaker 分布式运行具有模型并行性的分布式训练。

有关示例 Jupyter 笔记本，请参阅模型并行性示例.

我想要在 SageMaker 中使用 Hugging Face 训练和部署模型。

有关示例 Jupyter 笔记本，请参阅Spot 实例示例.

在 SageMaker 中使用 Hugge Face 训练文本分类模型时，我想捕获自定义指标并使用 SageMaker 检查点。

有关示例 Jupyter 笔记本，请参阅使用自定义指标训练示例.

我想在 SageMaker 中使用 Hugge Face 训练分布式问答 TensorFlow 模型。

有关示例 Jupyter 笔记本，请参阅分布式 TensorFlow 训练示例.

我想要在 SageMaker 中使用 Hugging Face 训练分布式总结模型。

有关示例 Jupyter 笔记本，请参阅分布式总结培训示例.

我想在 SageMaker 中使用 Hugge Face 训练图像分类模型。

有关示例 Jupyter 笔记本，请参阅视觉变压器培训示例.

我想在 SageMaker 中部署我训练有素的拥抱脸模型。

有关示例 Jupyter 笔记本，请参阅部署你的拥抱脸变压器作为推理示例.

我想在 SageMaker 中部署预先训练的拥抱脸模型。

有关示例 Jupyter 笔记本，请参阅部署预先训练的拥抱脸变压器以获取推理示例.

将 PyTorch 与 Amazon SageMaker 结合使

您可以通过，使用自定义 PyTorch 代码训练和部署模型。借助 SageMaker Python 开发工具包 PyTorch 估算器和模型以及 SageMaker 开源 PyTorch 容器，在 SageMaker 中编写和运行 PyTorch 脚本变得更加轻松。

您需要做什么？

我想要在 SageMaker 中训练自定义 PyTorch 模型。

有关示例 Jupyter 笔记本，请参阅PyTorch 示例笔记本在 Amazon SageMaker 示例 GitHub 存储库中。

有关文档，请参阅使用 PyTorch 训练模型。

我有一个在 SageMaker 中训练过的 PyTorch 模型，我想将它部署到托管终端节点。

有关更多信息，请参阅 。部署 PyTorch 模型.

我有一个在 SageMaker 之外训练过的 PyTorch 模型，我想将它部署到 SageMaker 终端节点。

有关更多信息，请参阅 。从模型数据部署终端节点.

我想查看 API 文档Amazon SageMaker Python 开发工具包PyTorch 类。

有关更多信息，请参阅 。PyTorch 类.

我想查找 SageMaker PyTorch 容器存储库。

有关更多信息，请参阅 。SageMaker PyTorch 容器 GitHub 库.

我想查找有关支持的 PyTorch 版本的信息AmazonDeep Learning Containers。

有关更多信息，请参阅 。可用的深度学习容器映像.

有关编写 PyTorch 训练脚本以及将 PyTorch 估算器和模型与 SageMaker 结合使用的一般信息，请参将 PyTorch 与 SageMaker Python 开发工具包结合使用.

Amazon SageMaker R 用户指南

该文档介绍了使用 R 以利用 Amazon SageMaker 功能的各种方法。该指南介绍了 SageMaker 的内置 R 内核，介绍了如何在 SageMaker 上开始使用 R，并最后介绍了几个示例笔记本。

这些示例分为三个级别：初级、中级和高级。他们从SageMaker 上的 R 入门中，在 SageMaker 上使用 R 进行端到端机器学习，然后以更高级的主题结束，例如使用 R 脚本的 SageMaker 处理以及的自带 (BYO) R 算法。

有关如何将自己的自定义 R 图像带到 Studio 的信息，请参阅自带 SageMaker 图像 (p. 101). 有关类似的博客文章，请参阅将自己的 R 环境带到 Amazon SageMaker Studio.

SageMaker 中的 rStudio Support

Amazon SageMaker 支持 rStudio 作为与 Amazon SageMaker 域集成的完全托管的集成开发环境 (IDE)。借助 rStudio 集成，您可以在域中启动 rStudio 环境，以便在 SageMaker 资源上运行 rStudio 工作流程。有关更多信息，请参阅 Amazon SageMaker 上的 rStudio (p. 131)。

SageMaker 中的 R 内核

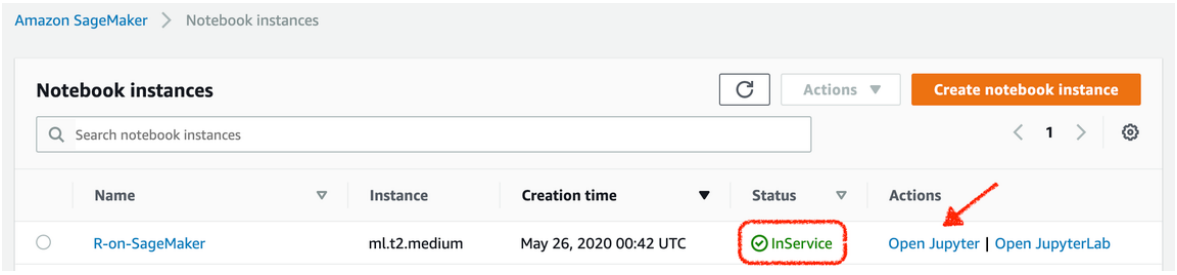
使用预装的 R 内核 SageMaker 笔记本实例支持 R。此外，R 内核具有网状库，这是 R 到 Python 的接口，因此，您可以从 R 脚本中使用 SageMaker Python 开发工具包的功能。

- 网状图书馆：提供 R 到的接口Amazon SageMaker Python 开发工具包. reticulate 程序包在 R 和 Python 对象之间转换。

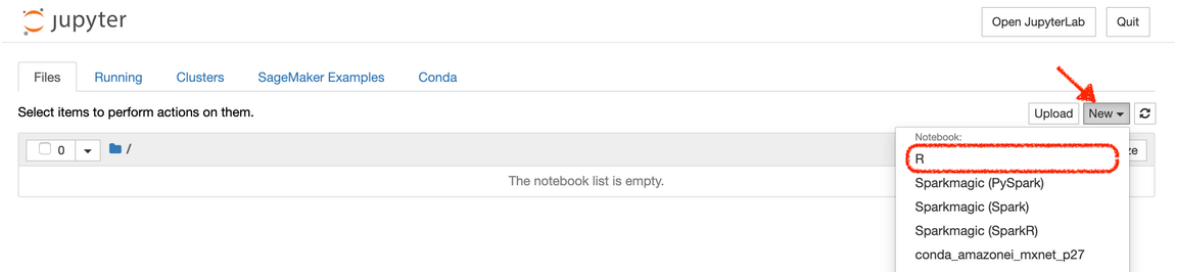
在 SageMaker 中开始使用 R

- 使用 t2.medium 实例类型和默认存储大小创建笔记本实例。如果您打算继续将该实例用于更高级的示例，或者以后创建更大的实例，您可以选择更快的实例和更多的存储空间。

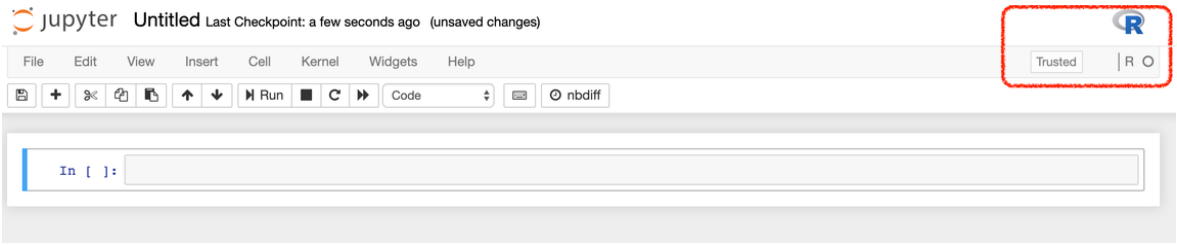
- 等到笔记本状态变为“In Service”(正在使用)，然后单击 Open Jupyter (打开 Jupyter)。



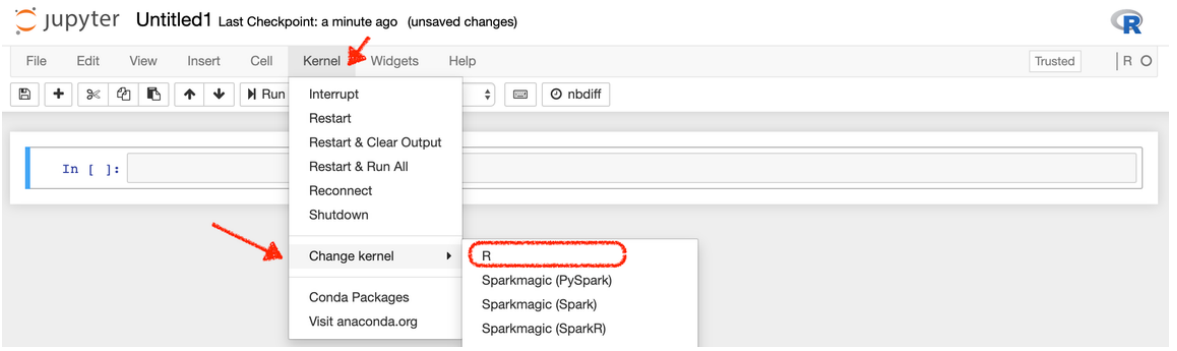
- 从可用的环境列表中创建一个具有 R 内核的新笔记本。



- 在创建了新的笔记本时，您将会在笔记本环境的右上角看到 R 徽标，并且还会在该徽标下看到内核为 R。这表明 SageMaker 已成功为该笔记本启动 R 内核。



- 或者，在您位于 Jupyter 笔记本时，您可以使用 Kernel (内核) 菜单，然后从 Change Kernel (更改内核) 选项中选择 R。



示例笔记本

先决条件

SageMaker 上的 R 入门：该示例笔记本介绍了如何使用 Amazon SageMaker R 内核开发 R 脚本。在本笔记本中，您可以设置 SageMaker 环境和权限，下载鲍鱼数据集来自 的UCI Machine Learning 存储库，对数据执行一些基本的处理和可视化，然后将数据另存为 .csv 格式到 S3。

初级

使用 R 内核 SageMaker Batch 转换：该示例笔记本介绍了如何使用 SageMaker 的转换器 API 和XGBoost 算法。 该笔记本也使用鲍鱼数据集。

中级

R 中 xgBoost 的超参数优化：该示例笔记本扩展了以前使用鲍鱼数据集和 XGBoost 的初级笔记本。它介绍了如何使用模型调整超参数优化。 您还会了解如何使用批量转换进行批量预测，以及如何创建模型终端节点以进行实时预测。

Amazon SageMaker 与 R 进行处理：SageMaker 处理您可以预处理、后处理和运行模型评估工作负载。该示例说明了如何创建 R 脚本以编排处理作业。

高级

在 SageMaker 中训练和部署自己的 R 算法：您是否已具有 R 算法，并且想将它引入 SageMaker 以优化、训练或部署它？ 该示例介绍了如何使用自定义 R 程序包自定义 SageMaker 容器，一直到使用托管终端节点推断 R 源模型。

将 Scikit-learn 与 Amazon SageMaker 结合使用

您可以通过，使用自定义 Amazon SageMaker cikit-learn 代码训练和部署模型。SageMaker Python 开发工具包 Scikit-learn 评估程序和模型以及 SageMaker 开源 Scikit-learn 容器可让在 Scikit-learn 脚本更容易中编写和在 SageMaker 中运行 Scikit-learn 脚本更容易。

您需要做什么？

我想在 SageMaker 中使用 Scikit-learn 进行数据处理、功能设计或模型评估。

有关示例 Jupyter 笔记本，请参阅 https://github.com/awslabs/amazon-sagemaker-examples/tree/master/sagemaker_processing/scikit_learn_data_processing_and_model_evaluation。

有关文档，请参阅ReadTheDocs.

我想要在 SageMaker 中训练自定义 Scikit-learn 模型。

有关示例 Jupyter 笔记本，请参阅 https://github.com/awslabs/amazon-sagemaker-examples/tree/master/sagemaker-python-sdk/scikit_learn_iris。

有关文档，请参阅使用 Scikit-learn 训练模型。

我有一个在 SageMaker 中训练过的 Scikit-learn 模型，我想将它部署到托管终端节点。

有关更多信息，请参阅 。部署 Scikit-learn 模型.

我有一个在 SageMaker 之外训练过的 Scikit-learn 模型，我想将它部署到 SageMaker 终端节点。

有关更多信息，请参阅 。从模型数据部署终端节点.

我想查看 API 文档Amazon SageMaker Python 开发工具包Scikit-learn 类。

有关更多信息，请参阅 。Scikit-learn 类.

我想查看有关 SageMaker Scikit-learn 容器的信息。

有关更多信息，请参阅 。SageMaker Scikit-learn 容器 GitHub 存储库.

有关编写 Scikit-learn 训练脚本以及将 Scikit-learn 估算器和模型与 SageMaker 结合使用的一般信息，请参阅将 Scikit-learn 与 SageMaker Python 开发工具包结合使用.

Amazon SageMaker Scikit-learn 容器支持的 Scikit-learn 版本：`0.20.0`、`0.23-1`。

Amazon SageMaker 使用 SparkML SparkML SparkML Services

这些区域有：Amazon SageMaker Python 开发工具包SparkML SparkML Services 模型和预测器以及 Amazon SageMaker 开源 SparkML Services 容器支持部署在 SageMaker 中使用 MLeap 序列化的 Apache Spark ML 管道，以获取推论。

有关使用 SparkML Services 容器将模型部署到 SageMaker 的信息，请参阅。SageMaker Spark ML 容器 GitHub 存储库. 有关的信息Amazon SageMaker Python 开发工具包SparkML Services 模型和预测器，请参阅SparkML 服务模型和预测变量 API 文档.

将 TensorFlow 与 Amazon SageMaker 结合使用

您可以通过 Amazon SageMaker 通过自定义 TensorFlow 代码训练和部署模型。借助 SageMaker Python 开发工具包 TensorFlow 评估程序和模型以及 SageMaker 开源 TensorFlow 容器，在 SageMaker 中编写和运行 TensorFlow 脚本变得更加轻松。

使用 TensorFlow 版本 1.11 及更高版本

对于 TensorFlow 版本 1.11 及更高版本，Amazon SageMaker Python 开发工具包支持脚本模式训练脚本。

您需要做什么？

我想要在 SageMaker 中训练自定义 TensorFlow 模型。

有关 Jupyter 笔记本的示例，请参阅 TensorFlow 脚本模式训练和服务。

有关文档，请参阅使用 TensorFlow 训练模型。

我有一个在 SageMaker 中训练过的 TensorFlow 模型，我想将它部署到托管终端节点。

有关更多信息，请参阅 。部署 TensorFlow 服务模型.

我有一个在 SageMaker 之外训练过的 TensorFlow 模型，我想将它部署到 SageMaker 终端节点。

有关更多信息，请参阅 。直接从模型项目部署.

我想查看 API 文档Amazon SageMaker Python 开发工具包TensorFlow 类。

有关更多信息，请参阅 。TensorFlow 评估程序.

我想查找 SageMaker TensorFlow 容器存储库。

有关更多信息，请参阅 。SageMaker TensorFlow 容器 GitHub 仓库.

我想查找有关支持的 TensorFlow 版本的信息AmazonDeep Learning Containers。

有关更多信息，请参阅 。可用的深度学习容器映像.

有关编写 TensorFlow 脚本模式训练脚本以及将 TensorFlow 脚本模式评估程序和模型与 SageMaker 结合使用的一般信息，请参阅将 TensorFlow 与 SageMaker Python 开发工具包结合使用.

对版本 1.11 及更低版本使用 TensorFlow 传统模式

这些区域有：Amazon SageMaker Python 开发工具包提供了支持 TensorFlow 版本 1.11 及更低版本的传统模式。在以下情况下，使用传统模式 TensorFlow 训练脚本在 SageMaker 中运行 TensorFlow 作业

- 您有传统模式脚本，且不想转换为脚本模式。
- 您想使用早于 1.11 的 TensorFlow 版本。

有关编写传统模式 TensorFlow 脚本以与 SageMaker Python 开发工具包结合使用的信息，请参阅TensorFlow SageMaker 评估程序和模型.

将 Triton 推理服务器与 Amazon SageMaker 结合使用

SageMaker 使客户能够通过 NVIDIA Triton 推理服务器使用自定义代码部署模型。此功能可以通过开发Triton 推理服务器容器. 这些容器包括 NVIDIA Triton 推理服务器、对常见机器学习框架的支持以及可让您在 SageMaker 上优化性能的有用环境变量。有关所有可用的 Deep Learning Containers 映像的列表，请参阅可用的 Deep Learning Containers 映像. Deep Learning Containers 映像通过安全补丁进行维护和定期更新。

您可以将 Triton 推理服务器容器与 SageMaker Python SDK 结合使用，就像使用 SageMaker 模型中的任何其他容器一样。但是，使用 SageMaker Python SDK 是可选的。你可以将 Triton 推理服务器容器与Amazon CLI和Amazon SDK for Python (Boto3).

有关 NVIDIA Triton 推理服务器的更多信息，请参阅Triton 文档.

推理

为了进行推理，您可以将训练有素的机器学习模型与 Triton 推理服务器结合使用，在 SageMaker 上部署推理作业。

Triton 推理服务器容器的一些关键功能包括：

- Support 多个框架：Triton 可用于部署来自所有主要机器学习框架的模型。Triton 支持 TensorFlow GraphdeF 和 SavedModel、ONNX、PyTorch TorchScript、TensorRT 和自定义 Python /C++ 模型格式。
- 模拟管道：Triton 模型合奏代表一个模型的管道，具有前/后处理逻辑以及它们之间输入和输出张量的连接。向整体发出单个推理请求会触发整个管道的执行。
- 并发模型执行：同一模型的多个实例可以在同一个 GPU 或多个 GPU 上同时运行。
- 动态批处理：对于支持批处理的模型，Triton 有多个内置的调度和批处理算法，这些算法将单个推理请求结合在一起，以提高推理吞吐量。这些调度和批处理决策对于请求推断的客户来说是透明的。
- 多样化的 CPU 和 GPU 支持：这些模型可以在 CPU 或 GPU 上执行，以获得最大的灵活性并支持异构计算要求。

您需要做什么？

我想在 SageMaker 中部署我训练有素的 PyTorch 模型。

有关示例 Jupyter 笔记本，请参阅使用 Triton 推理服务器示例部署 PyTorch Resnet50 模型.

我想在 SageMaker 中部署我训练有素的 Huging Face 模型。

有关示例 Jupyter 笔记本，请参阅使用 Triton 推理服务器示例部署 PyTorch BERT 模型.

支持的区域和配额

对于Amazon亚马逊支持的地区 SageMaker 以及每个区域中可用的 Amazon Elastic Compute Cloud (Amazon EC2) 实例类型，请参阅Amazon SageMaker 定价.

有关的列表 SageMaker 每个区域的服务终端节点和 SageMaker 每种实例类型的服务配额，请参阅亚马逊 SageMaker 终端节点和配额.

Note

以下 SageMaker 该功能不在亚太地区（大阪）区域中可用：

- 亚马逊 SageMaker Autopilot
- 澄清
- SageMaker Edge 管理器
- Ground Truth
- 超参数调整 (HPO)
- 亚马逊 SageMaker 推理推理器
- JumpStart
- 亚马逊 SageMaker 模型监控器
- 强化学习

亚马逊 SageMaker 所有工作室都可用Amazon亚马逊支持的地区 SageMaker 除外Amazon GovCloud （美国）地区。

Amazon 上的 rStudio SageMaker 可以在以下内容中使用AmazonAmazon SageMaker 支持的区域。

- ap-northeast-1
- ap-northeast-2
- ap-south-1
- ap-southeast-1
- ap-southeast-2
- ca-central-1
- eu-central-1
- eu-north-1
- eu-west-1
- eu-west-2
- eu-west-3
- sa-east-1
- us-east-1
- us-east-2
- us-west-1
- us-west-2

亚马逊 SageMaker 管道在所有Amazon支持的区域Amazon除外Amazon GovCloud （美国）地区。SageMaker 项目可在Amazon地区 CodePipeline 可用。有关 的更多信息 CodePipeline 区域可用性，请参阅Amazon区域服务清单.

亚马逊 SageMaker 通常可以使用无服务器推理Amazon其中的商业区域 SageMaker 可用（除外）Amazon中国地区）。

关于服务配额

根据你的活动和资源随时间的推移使用情况， SageMaker 配额可能不同于默认配额 SageMaker 上列出的配额亚马逊 SageMaker 终端节点和配额中的Amazon一般参考. 如果您遇到超出配额的错误消息，并且您需要扩展您的配额，请扩展您的配额， SageMaker 资源，请执行中的步骤请求提升服务配额以下项目： SageMaker 资源 (p. 29)此页面上的步骤，请求提高配额，从AmazonSupport。这些区域有： SageMaker 服务团队审核和评估请求 case-by-case 基础。有关更多信息，请参阅如何解决？ ResourceLimitExceeded Amazon SageMaker 中的错误？中的Amazon Web Services Support知识中心和AmazonService Quotas中的Amazon一般参考.

请求提升服务配额以下项目： SageMaker 资源

按照此过程中的步骤操作，请求配额（也称为）限制) 提高限制 SageMaker 资源的费用。

要请求提升服务配额，请执行

1. 登录到Amazon Web Services Support控制台在Amazon Web Services SupportCenter.
2. 在左侧导航窗格中，选择。您的支持案例，然后选择。创建案例.
3. 适用于限制类型中的案例细节部分，选择具体的 SageMaker 组件。
4. 适用于请求 1中的请求部分，选择。区域、资源类型, 和限制对于您想要提高配额的资源，请执行。
5. 适用于新限制值中，键入新配额的整数值。
6. 适用于案例描述，包含以下内容：

   a. 显示当前配额的错误消息。例如，错误消息的内容应与以下内容类似：

   ```
   ResourceLimitExceeded: An error occurred (ResourceLimitExceeded) when calling
   the CreateTrainingJob operation: The account-level service limit 'ml.p3dn.24xlarge
   for training job usage' is 0 Instances, with current utilization of 0 Instances
   and a request delta of 1 Instances.
   Please contact Amazon support to request an increase for this limit.
   ```

   b. 简要描述需要增加服务配额的使用案例。
7. 如果要为其他资源添加更多配额增加请求，请选择添加另一个请求然后重复前面的步骤。
8. 选择提交以完成请求。

开始使用 Amazon SageMaker

您必须先注册，然后才能使用 Amazon SageMakerAmazon账户并按照中的步骤创建 IAM 管理员用户设置 Amazon SageMaker 先决条件 (p. 30).

亚马逊 SageMaker Studio Lab 不需要Amazon账户或 IAM 集成。

完成这些任务后，请根据您的使用案例继续执行以下主题之一。

- 注册到 Amazon SageMaker 域 (p. 32)：按照以下步骤创建域名，使您可以访问亚马逊 SageMaker Amazon SageMaker 上的工作室和 rStudio。有关域的更多信息，请参阅亚马逊 SageMaker Machine Learning 环境 (p. 75).
- SageMaker JumpStart (p. 38)：按照以下步骤开始使用 SageMaker JumpStart 并了解 SageMaker 通过精心策划的一键式解决方案、例如笔记本电脑和可部署的预训练模型实现的功能和功能。使用 SageMaker JumpStart，这是亚马逊的一项功能 SageMaker Studio，你必须先加入亚马逊 SageMaker 域。
- Amazon 入门 SageMaker 笔记本实例 (p. 58)：按照以下步骤来训练和部署 Machine Learning (ML) 模型 SageMaker 笔记本实例。通过在 Amazon Elastic Compute Cloud (Amazon EC2) 上启动 Jupyter 服务器，并提供预配置的内核，SageMaker 笔记本实例有助于创建环境。有关更多信息，请参阅 使用 Amazon SageMaker 笔记本实例 (p. 147)。
- ???：按照以下步骤开始与亚马逊合作 SageMaker Lab Studio Lab 是一项免费服务，可让您访问Amazon 计算资源，在基于开源 JupyterLab 的环境中，无需Amazonaccount.

主题

- 设置 Amazon SageMaker 先决条件 (p. 30)
- 注册到 Amazon SageMaker 域 (p. 32)
- SageMaker JumpStart (p. 38)
- Amazon 入门 SageMaker 笔记本实例 (p. 58)

设置 Amazon SageMaker 先决条件

在本节中，您将注册Amazon创建账户Amazon Identity and Access Management(IAM) 管理员用户。

如果您不熟练使用 SageMaker，我们建议您先阅读Amazon SageMaker 的工作原理 (p. 3).

主题

- 创建 Amazon 账户 (p. 30)
- 创建 IAM 管理员用户和组 (p. 31)

创建 Amazon 账户

在本节中，您将注册 Amazon 账户。如果您已有 Amazon 账户，请跳过此步骤。

注册 Amazon Web Services (Amazon) 您的Amazon所有账户自动注册Amazon服务，包括 SageMaker。您只需为使用的服务付费。

创建 Amazon 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

请记下您的 Amazon 账户 ID，您将在下一个任务中用到它。

创建 IAM 管理员用户和组

创建时Amazon账户时，您将获得一个单点登录身份，此身份对所有Amazon账户中的服务和资源。此身份称作 Amazon 账户根用户。登录到Amazon使用用于创建账户的电子邮件地址和密码控制台，您将获得对所有Amazon您的账户中的资源。

强烈建议您不 使用根用户执行日常任务，即使是管理任务。相反，请遵循创建单独的 IAM 用户，一个 Amazon Identity and Access Management(IAM) 管理员用户。然后请妥善保存根用户凭证，仅用它们执行少数账户和服务管理任务。

创建管理员用户

1. 在 Amazon 账户中创建管理员用户。有关说明，请参阅创建您的第一个 IAM 用户和管理员组中的IAM 用户指南.

   Note

   我们假定您将管理员用户凭证用于本指南中的练习和过程。如果您选择创建并使用另一个 IAM 用户，请向该用户授予最低权限。有关更多信息，请参阅 使用身份进行身份验证 (p. 2691)。
2. 确保您的管理员用户拥有AmazonSageMakerFullAccess策略，以及包含以下内容的策略来创建 SageMaker 域。有关创建 IAM 策略的更多信息，请参阅创建 IAM 策略.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:*"
            ],
            "Resource": [
                "arn:aws:sagemaker:*:*:domain/*",
                "arn:aws:sagemaker:*:*:user-profile/*",
                "arn:aws:sagemaker:*:*:app/*",
                "arn:aws:sagemaker:*:*:flow-definition/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "servicecatalog:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

注册到 Amazon SageMaker 域

Amazon SageMaker 域由关联的 Amazon Elastic File System (Amazon EFS) 卷、授权用户列表以及各种安全性、应用程序、策略和 Amazon Virtual Private Cloud (Amazon VPC) 配置组成。使用 Amazon SageMaker Amazon Studio SageMaker Studio 笔记本电脑和 rStudio，你必须完成亚马逊 SageMaker 使用域入职过程 SageMaker 控制台。有关亚马逊的更多信息 SageMaker 域名，请参阅亚马逊 SageMaker Machine Learning 环境 (p. 75).

创建 Amazon 最简单的方式 SageMaker 域名是关注快速设置过程。快速设置使用的默认设置与标准设置程序。这些设置包括可共享的笔记本和公共互联网访问。如需更好地进行控制（包括使用的选项）IAM Identity Center身份验证和 rStudio，请使用标准设置程序。

主题

- 注册到 Amazon SageMaker 域使用快速设置 (p. 32)
- 注册到亚马逊 SageMaker 使用 IAM 的域 (p. 33)
- 选择 VPC。 (p. 35)
- 删除 Amazon SageMaker 域 (p. 36)

注册到 Amazon SageMaker 域使用快速设置

本主题介绍如何加入亚马逊 SageMaker 使用域快速设置过程，它使用Amazon Identity and Access Management(IAM) 身份验证。有关如何使用标准 IAM 过程进行注册的信息，请参阅。使用 IAM 进行注册 (p. 33).

使用快速设置过程。

使用注册到域快速设置

1. 打开SageMaker 控制台.
2. 选择控制面板在页面左上角。
3. 在存储库的设置 SageMaker 域页面，选择快速设置.
4. UNDER用户资料，对于名称保留默认名称或创建新名称。名称最多可以有 63 个字符。有效字符：A-Z、a-z、0-9 和-（连字符）。
5. 适用于默认执行角色从角色选择器中选择选项。这是分配给的默认角色亚马逊 SageMaker 域用户配置文件。

   如果选择输入自定义 IAM 角色 ARN，角色必须至少具有附加的信任策略，该策略可授予 SageMaker 担任角色的权限。有关更多信息，请参阅 SageMaker 角色 (p. 2722)。

   如果您选择创建新角色，则将打开创建 IAM 角色对话框：

   - 对于您指定的 S3 存储桶，请指定笔记本的用户可以访问的其他 S3 存储桶。如果您不希望添加对更多存储桶的访问权限，请选择无。
   - 选择创建角色. SageMaker 创建新 IAM`AmazonSageMaker-ExecutionPolicy`角色使用AmazonSageMakerFullAccess附加策略。
6. 选择 Submit（提交）。
7. 在弹出窗口中，选择要使用的 Amazon Virtual Private Cloud (Amazon VPC) 和子网。
8. 选择 Save and continue。

   Note

   如果收到需要创建 VPC 的错误消息，请参阅选择 VPC。 (p. 35).

当状态变为准备就绪时，将启用并选择您指定的用户名。这些区域有：添加用户和删除用户按钮，以及启动应用链接也已启用。

9. 从启动应用，选择工作室. 这些区域有：亚马逊 SageMaker 工作室显示加载页面。

   Studio 打开时，您可以开始使用它。

现在你已经登上了 SageMaker Studio，以后可使用以下步骤访问 Studio。

在注册后访问 Studio

1. 打开SageMaker 控制台.
2. 选择控制面板在页面左上角。
3. 在存储库的控制面板，选择您的用户名，然后选择启动应用. Select工作室.

添加更多用户

1. 在存储库的控制面板，选择添加用户.
2. 重复第一个过程中的步骤 4 和 5：“加入亚马逊” SageMaker 使用快速设置的域。”
3. 选择 Submit（提交）。

有关使用 SageMaker 工作室，请参阅SageMaker Studio (p. 77).

注册到亚马逊 SageMaker 使用 IAM 的域

本主题介绍如何加入亚马逊 SageMaker 域使用标准设置程序Amazon Identity and Access Management(IAM) 身份验证。要更快地使用 IAM 上载，请参阅使用快速安装注册 (p. 32).

使用 IAM 注册到域

1. 打开SageMaker 控制台.
2. 选择控制面板在页面的左上角。
3. 在存储库的设置 SageMaker 域在页面上，选择标准设置.
4. 选择 Configure (配置)。

第 1 步：常规设置

1. 适用于身份验证，选择AmazonIdentity and Access Management (IAM).
2. UNDER许可，对于IAM 角色从角色选择器中选择选项。

   如果选择输入自定义 IAM 角色 ARN，角色必须至少具有附加的信任策略，该策略可授予 SageMaker 担任角色的权限。有关更多信息，请参阅 SageMaker 角色 (p. 2722)。

   如果您选择创建新角色，则将打开创建 IAM 角色对话框：

   a. 对于您指定的 S3 存储桶，请指定笔记本的用户可以访问的其他 S3 存储桶。如果您不希望添加对更多存储桶的访问权限，请选择无。
   b. 选择创建角色. SageMaker 创建新 IAM`AmazonSageMaker-ExecutionPolicy`角色使用AmazonSageMakerFullAccess附加策略。
3. UNDER网络和存储中，指定以下内容：

   - 有关 VPC 信息 — 有关更多信息，请参阅选择 VPC。 (p. 35).

- （可选）存储加密密密钥– SageMaker 使用Amazon KMS key加密您的 Amazon Elastic File System (Amazon EFS) 和 Amazon Elastic Block Store (Amazon EBS) 文件系统。默认情况下，它使用 Amazon 托管式密钥. 要使用客户托管的密钥，请输入其密钥 ID 或 Amazon 资源名称 (ARN)。有关更多信息，请参阅 使用加密保护静态数据 (p. 2686)。

  Note

  传输途中的加密仅适用于亚马逊 SageMaker Studio。

4. 选择 Next（下一步）。

步骤 2: Studio 设置

1. UNDER共享笔记本配置中，接受默认的笔记本共享配置或自定义选项。
2. UNDERSageMaker 项目和 JumpStart，接受默认项目 JumpStart 设置或自定义管理员和用户是否可以创建项目并使用 Jumpstart。有关更多信息，请参阅 使用项目所需的 SageMaker Studio 权限 (p. 2511)。
3. 选择 Next（下一步）。

步骤 3: RStudio 设置

1. UNDERRStudio 工作台中，验证是否自动检测到您的 rStudio 许可证。有关获取 RStudio 许可并使用 SageMaker 激活该许可证的更多信息，请参阅rStudio 许可证 (p. 133).
2. 选择要在其上启动 RStudio 服务器的实例类型。有关更多信息，请参阅 RStudioServerPro 实例类型 (p. 134)。
3. UNDER许可中，创建您的角色，也可以选择现有角色。此角色必须具有以下权限策略。此政策允许 rStudioServerPro 应用程序访问必要的资源，并允许亚马逊 SageMaker 当现有的 rStudioServerPRO 应用程序位于`Deleted`要么`Failed`状态。有关向角色添加权限的信息，请参阅修改角色权限策略（控制台）.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "license-manager:ExtendLicenseConsumption",
                "license-manager:ListReceivedLicenses",
                "license-manager:GetLicense",
                "license-manager:CheckoutLicense",
                "license-manager:CheckInLicense",
                "logs:CreateLogDelivery",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DeleteLogDelivery",
                "logs:Describe*",
                "logs:GetLogDelivery",
                "logs:GetLogEvents",
                "logs:ListLogDeliveries",
                "logs:PutLogEvents",
                "logs:PutResourcePolicy",
                "logs:UpdateLogDelivery",
                "sagemaker:CreateApp"
            ],
            "Resource": "*"
        }
    ]
}
```

4. UNDERRStudio Connect中，添加 rStudio Connect 服务器的 URL。rStudio Connect 是 Shine 应用程序、R Markdown 报告、仪表板、绘图等的发布平台。当您在 Amazon SageMaker 上加入 RStudio 时，不会创建 rStudio Connect 服务器。您必须在 EC2 实例上创建 RStudio Connect 服务器才能使用与 Amazon SageMaker 进行 Connect。有关更多信息，请参阅 rStudio Connect 网址 (p. 134)。
5. UNDERrStudio Package 管理器中，添加 rStudio Package 管理器的 URL。 SageMaker 当您加载 rStudio 时，为 Package 件包管理器创建默认的软件包存储库。有关 rStudio Package 管理器的更多信息，请参阅rStudio Package 管理器 (p. 135).
6. 选择 Submit (提交)。

现在你已经加载了SageMaker 域，以后可使用以下步骤访问 Studio 或 rStudio。

访问SageMaker 域注册后

1. 打开SageMaker 控制台.
2. 选择控制面板在页面的左上角。
3. 在存储库的控制面板，选择您的用户名，然后选择启动应用. 选择工作室或 rStudio。

有关使用的信息 SageMaker 工作室，请参阅SageMaker Studio (p. 77).

有关使用 rStudio 的信息，请参阅Amazon SageMaker 上的 rStudio (p. 131).

选择 VPC。

本主题提供有关在您登录到 Amazon 时选择 Amazon Virtual Private Cloud (VPC) 的详细信息。 SageMaker 域。有关入职的更多信息，请访问 SageMaker 域名，请参阅注册到 Amazon SageMaker 域 (p. 32).

默认情况下， SageMaker 域使用两个 VPC。一个 VPC 由亚马逊管理 SageMaker 并提供直接互联网访问。您可以指定另一个 VPC，它提供域与 Amazon Elastic File System (EFS) 卷之间的加密流量。

您可以更改此行为，以便 SageMaker 通过指定的 VPC 发送所有流量。选择此选项时，必须提供与 SageMaker API 和 SageMaker 运行时间和各种AmazonStudio 和您的 Studio 笔记本使用的服务，例如 Amazon Simple Storage Service (Amazon S3) 和 Amazon CloudWatch。

当你登机时 SageMaker 域名，你告诉 SageMaker 通过将网络访问类型设置为通过 VPC 发送所有流量仅限 VPC.

指定 VPC 信息

当您在以下过程中指定 VPC 实体（即 VPC、子网或安全组）时，将根据当前实体的数量显示三个选项之一 Amazon区域。行为如下：

- 一个实体 — SageMaker 使用该实体。这一点无法更改。
- 多个实体 — 您必须从下拉列表中选择实体。
- 没有实体 — 您必须创建一个或多个实体才能使用域。选择创建 <entity>在新的浏览器选项卡中打开 VPC 控制台。在创建实体后，返回域试用页面以继续入职过程。

此程序是亚马逊的一部分 SageMaker 选择时的域入职流程标准设置. 您的 VPC 信息在网络部分。

1. 选择 VPC。
2. 选择一个或多个子网。如果你不选择任何子网， SageMaker 将使用 VPC 中的所有子网。
3. 选择网络访问类型。

    - 仅限公共互联网— 非 EFS 流量经过 SageMaker 托管 VPC，允许互联网访问。域和您的 Amazon EFS 卷之间的流量是通过指定的 VPC 进行的。
    - 仅限 VPC— All SageMaker 流量通过指定的 VPC 和子网。默认情况下，禁用 Internet 访问。

4. 选择安全组。如果你选择仅限公共互联网，此为可选步骤。如果你选择仅限 VPC，此为必填步骤。

   Note

   有关允许的最大安全组数量，请参阅UserSettings.

对于中的 VPC 要求仅限 VPC模式，请参阅Connect (连接) SageMaker VPC 中的 Studio 笔记本电脑到外部资源 (p. 2781).

删除 Amazon SageMaker 域

当你在亚马逊上机时 SageMaker Amazon 使用 IAM 身份验证的域名 SageMaker 为您的账户创建域。域由授权用户列表、配置设置和 Amazon Elastic File System (Amazon EFS) 卷组成，其中包含笔记本、资源和构件等用户数据。用户可以拥有多个应用程序（应用），这些应用程序支持用户的笔记本、终端和控制台的读取和执行体验。

您的文件作为备份保存在 Amazon EFS 卷中。此备份包括已装入目录中的文件，即`/home/sagemaker-user`对于 Jupyter 和`/root`为您的内核。从这些挂载的目录中删除文件时，内核或应用程序可能会将已删除的文件移动到隐藏的垃圾文件夹中。如果垃圾文件夹位于已装入的目录中，则这些文件将复制到 Amazon EFS 卷中，并将产生费用。为避免这些 Amazon EFS 费用，您必须识别并清理垃圾文件夹位置。默认应用程序和内核的垃圾文件夹位置为`~/.local/`. 这可能会有所不同，具体取决于用于自定义应用或内核的 Linux 发行版。有关 Amazon EFS 卷的更多信息，请参阅在中管理 EFS 存储卷 SageMaker 工作室 (p. 128).

要退回亚马逊 SageMaker 要到您注册之前的状态，您必须删除此域。您可以使用删除域Amazon控制台，Amazon Command Line Interface(Amazon CLI)，或者 SageMaker SDK。当您使用Amazon控制台删除域时，Amazon EFS 卷将被分离但不会删除。默认情况下，当您使用Amazon CLI或者软件开发工具包删除域名。但是，当你使用Amazon CLI或者 SDK，您可以将`RetentionPolicy`到`HomeEfsFileSystem=Delete`删除 EFS 卷以及域。

要删除域，则域中不能包含任何用户配置文件。要删除用户配置文件，则配置文件中不能包含任何非失败应用程序。

在删除这些资源时，会出现以下情况：

- 应用程序 — 保存用户主目录中的数据（文件和笔记本）。未保存的笔记本数据将丢失。
- 用户配置文件 — 用户无法再登录到域，并且无法访问其主目录，但不会删除数据。管理员可以从 Amazon EFS 卷中检索数据，在该卷中，数据存储在用户的Amazonaccount.

  Note

  您必须具有管理员权限才能删除域。

您只能删除状态为的应用程序`InService`，它显示为Ready在域中。无需删除状态为 `Failed` 的应用程序即可删除包含的域。在域中，尝试删除处于失败状态的应用会导致出现错误。

主题

- 删除 Amazon SageMaker 域 (控制台) (p. 36)
- 删除 Amazon SageMaker 域 (CLI) (p. 37)

删除 Amazon SageMaker 域 (控制台)

如需删除域

1. 打开SageMaker 控制台.
2. 选择控制面板在页面左侧。

3. 对 User name (用户名) 列表中的每个用户重复以下步骤。

   a. 选择用户。

   b. 在 User Details (用户详细信息) 页面上，对于 Apps (应用程序) 列表中的每个非失败应用程序，选择 Delete app (删除应用程序)。

   c. 在 Delete app (删除应用) 对话框中，选择 Yes, delete app (是，删除应用)，在确认字段中键入 delete，然后选择 Delete (删除)。

   d. 当状态对于所有应用程序显示为Deleted (已删除)，选择编辑.

   e. 从编辑用户在页面上，选择删除用户.

   f. 在 Delete user (删除用户) 对话框中，选择 Yes, delete user (是，删除用户)，在确认字段中键入 delete，然后选择 Delete (删除)。

   Important

   删除用户后，他们将失去对包含其数据（包括笔记本和其他项目）的 Amazon EFS 卷的访问权限。数据不会被删除，管理员可以访问。

4. 删除所有用户后，选择删除域.
5. 在存储库的删除域对话框中，选择是的，删除域，键入删除在确认字段中，然后选择Delete.

删除 Amazon SageMaker 域 (CLI)

如需删除域

1. 检索您账户中的域列表。

```
aws --region Region sagemaker list-domains
```

2. 检索要删除的域的应用程序列表。

```
aws --region Region sagemaker list-apps \
    --domain-id-equals DomainId
```

3. 删除列表中的所有应用程序。

```
aws --region Region sagemaker delete-app \
    --domain-id DomainId \
    --app-name AppName \
    --app-type AppType \
    --user-profile-name UserProfileName
```

4. 检索域中用户配置文件的列表。

```
aws --region Region sagemaker list-user-profiles \
    --domain-id-equals DomainId
```

5. 删除列表中的所有用户配置文件。

```
aws --region Region sagemaker delete-user-profile \
    --domain-id DomainId \
    --user-profile-name UserProfileName
```

6. 删除域。要同时删除 Amazon EFS 卷，请指定HomeEfsFileSystem=Delete.

```
aws --region Region sagemaker delete-domain \
```

```
    --domain-id DomainId \
    --retention-policy HomeEfsFileSystem=Retain
```

SageMaker JumpStart

SageMaker JumpStart 为各种问题类型提供预训练的开源模型，以帮助您开始使用机器学习。您可以在部署之前逐步训练和调整这些模型。 JumpStart 还提供了为常见使用案例设置基础架构的解决方案模板，以及使用 SageMaker 进行机器学习的可执行示例笔记本

您可以通过亚马逊的 JumpStart 登录页面访问预训练的模型、解决方案模板和示例 SageMaker Studio。以下步骤显示如何访问 JumpStart 使用亚马逊的模型和解决 SageMaker Studio。

您也可以使用 SageMaker Python 开发工具包。有关如何使用的信息 JumpStart 通过 API 以编程方式进行模型，请参阅使用 SageMaker 预训练模型的 JumpStart 算法.

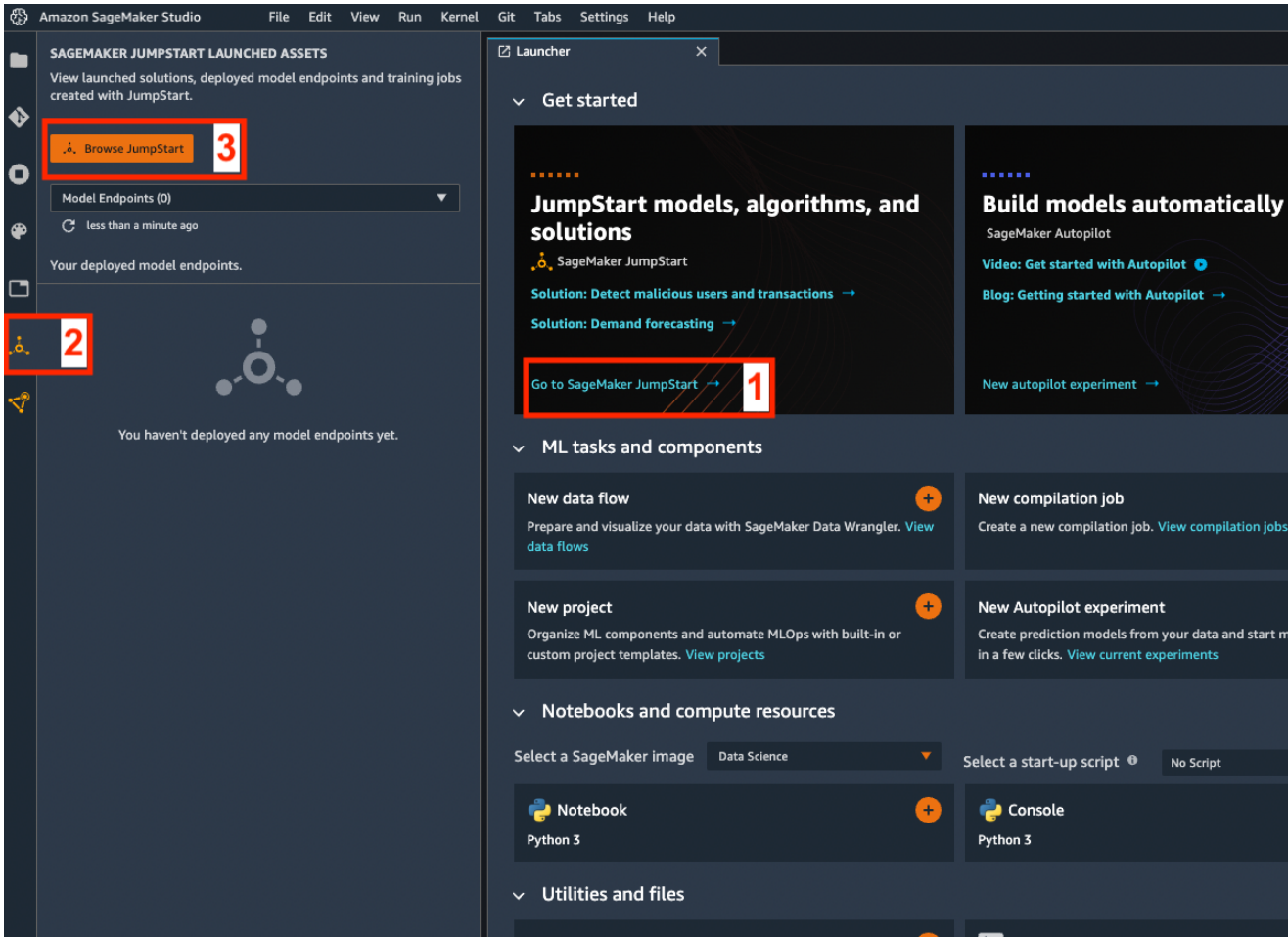
主题

- 打开 JumpStart (p. 38)
- 使用 JumpStart (p. 39)
- 方案模板 (p. 40)
- 模型 (p. 42)
- 部署模型 (p. 44)
- 模型部署配置 (p. 44)
- 微调模型 (p. 52)
- 微调数据源 (p. 52)
- 微调部署配置 (p. 53)
- 超参数 (p. 53)
- 训练输出 (p. 54)
- 亚马逊 SageMaker JumpStart 行业 (p. 54)

打开 JumpStart

在 Amazon 中 SageMaker 工作室，开放 JumpStart 使用以下操作之一：

- 这些区域有： JumpStart 中的启动程序开始使用部分。
- 这些区域有： JumpStart （ 图标) 在左侧边栏.
- 这些区域有：浏览 JumpStart按钮在已启动的资源窗格中。

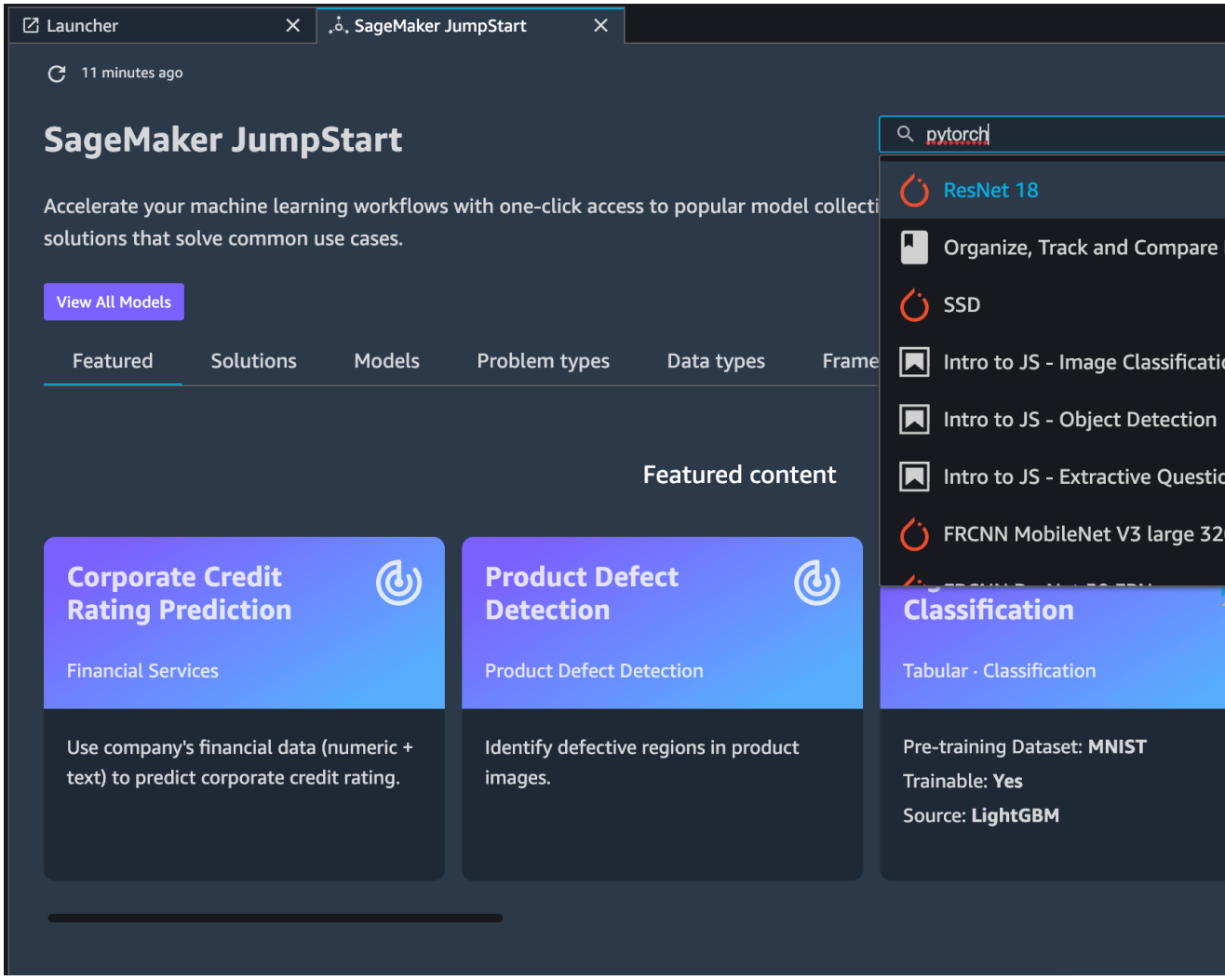


Important

在下载或使用第三方内容之前：您有责任查看并遵守任何适用的许可证条款，并确保它们在您的使用案例中可以接受。

使用 JumpStart

从 SageMaker JumpStart 登录页面，您可以浏览解决方案、型号、笔记本电脑和其他资源。您还可以查看当前启动的解决方案、终端节点和培训作业。使用 JumpStart 搜索栏中，您可以搜索感兴趣的主题。



您可以找到 JumpStart 通过使用搜索或浏览搜索面板后面的每个类别来资源：

- 精选— 最新或最常用的解决方案、模型和示例。
- 解决方案— 只需一步即可启动全面的机器学习解决方案 SageMaker 到其他Amazon服务。Select探索所有解决方查看所有可用的解决方案。
- 模型— 从文本、视觉和表格模型集合中找到适合您需求的模型。您可以按问题类型、数据类型和框架筛选集合。然后，只需一步即可部署和优化预训练的图像分类和物体检测模型。Select探索所有模型查看所有可用型号。
- 资源— 使用示例笔记本、博客和视频教程来学习并开始你的问题类型。
  - 示例笔记本— 运行使用的示例笔记本 SageMaker 诸如竞价型实例训练和对各种模型类型和使用案例的实验等功能。
  - 博客— 阅读由亚马逊托管的机器学习专家的详细信息和解决方案。
  - 视频教程— 观看视频教程 SageMaker 由亚马逊托管的机器学习专家提供的功能和机器学习用例。

方案模板

从 JumpStart 登录页面，获取为常见使用案例设置完整基础架构的解决方案模板。

当你选择解决方案模板时， JumpStart 显示解决方案的描述和启动按钮。选择的时候启动，JumpStart 创建了运行解决方案所需的所有资源。这包括培训和模型托管实例。

晚于 JumpStart 启动解决方案， JumpStart 显示打开笔记本按钮。选择按钮以使用提供的笔记本电脑并探索解决方案的功能。当生成工件时、在启动期间或运行提供的笔记本之后，它们将列在生成的构件表。你可以使用垃圾图标删除单个工件 ( )。您可以通过选择删除解决方案的所有资源删除解决方案资.

以下是可用的解决方案模板的完整列表：

使用案例	解决方案名称	描述	开始使用
时间序列预测	需求预测	使用三种先进的时间序列预测算法对多变量时间序列数据的需求进行预测：lstNet、预言家，和SageMaker DeepAR.	GitHub”
预测信用率	企业信用评级预测	使用多模式（长文本和表格式）机器学习进行高质量信用预测Amazon AutoGLON 表格.	GitHub”
	基于图表的信用评分	使用表格数据和企业网络预测企业信用评级，通过培训图神经网络GraphSage和Amazon AutoGLON 表格模型。	在亚马逊中查找 SageMaker Studio。
	解释信用决策	预测信用申请中的信用违约并使用lightgBm和SHAP（SHAPLEY 加法解释）.	GitHub”
从文档中提取数据并进行分析	隐私情绪分类	匿名处理文本以便在情绪分类中更好地保护用户隐私。	GitHub”
	文档摘要、实体和关系提取	使用变形金刚PyTorch中的库。	GitHub”
	识别手写	通过训练来识别图像中的手写文字对象检测模型和手写识别模型. 使用标签自己的数据SageMaker Ground Truth.	GitHub”
	在表格记录中填写缺少的值	通过训练 a 填充表格记录中的缺失值SageMaker Autopilot模型。	GitHub”
欺诈侦测	检测恶意用户和交易	使用自动检测交易中潜在的欺诈活动SageMaker XGBoost使用过度采样	GitHub”

使用案例	解决方案名称	描述	开始使用
		技术合成少数群体过采样（SMOTE）。	
	使用深图库进行金融交易中的欺诈检测	通过培训图形卷积网络使用深层图库和SageMaker XGBoost模型。	GitHub”
预测性维护	车队的预测性维护	通过卷积神经网络模型，使用车辆传感器和维护信息预测车队故障。	GitHub”
	制造业的预测性维护	通过训练一个来预测每个传感器的剩余使用寿命堆叠双向 LSTM 神经网络使用历史传感器读数的模型。	GitHub”
计算机视觉	图片中的商品缺陷检测	通过训练和识别商品图片中的缺陷区域对象检测模型.	GitHub”
预测流失	使用文本预测流失	使用数字、分类和文本要素预测流失BERT 编码器和随机森林分类器.	GitHub”
个性化推荐	具有深层图库的身份图中的实体解析	通过对在线广告进行跨设备实体链接，通过培训图形卷积网络和深层图库.	GitHub”
	购买模型	通过培训一个预测客户是否会进行购买SageMaker XGBoost模型。	GitHub”
强化学习	Battlesnake AI 比赛的强化学习	为训练和推理提供强化学习工作流程BattleSnakeAI 竞赛。	GitHub”
	Procgen 挑战的分布式强化学习	分布式强化学习入门套件NeurIPS 2020 年 Progen强化学习挑战。	GitHub”

模型

JumpStart 支持十五种最受欢迎的问题类型的模型。在受支持的问题类型中，Vision 和 NLP 相关类型共有 13 种。有八种问题类型支持增量训练和微调。有关增量训练和超参数调整的更多信息，请参阅SageMaker 自动模型优化。 JumpStart 还支持四种常见的表格数据建模算法。

你可以从 JumpStart 工作室中的着陆页。选择模型时，模型详细信息页面将提供有关该模型的信息，您可以通过几个步骤来训练和部署模型。描述部分描述了可以对模型执行的操作、预期的输入和输出类型以及微调模型所需的数据类型。

您还可以通过编程方式使用模型SageMaker Python SDK.

下表总结了问题类型列表以及指向其示例 Jupyter 笔记本的链接。有关的完整列表 JumpStart 模型，请参阅JumpStart 可用型号表.

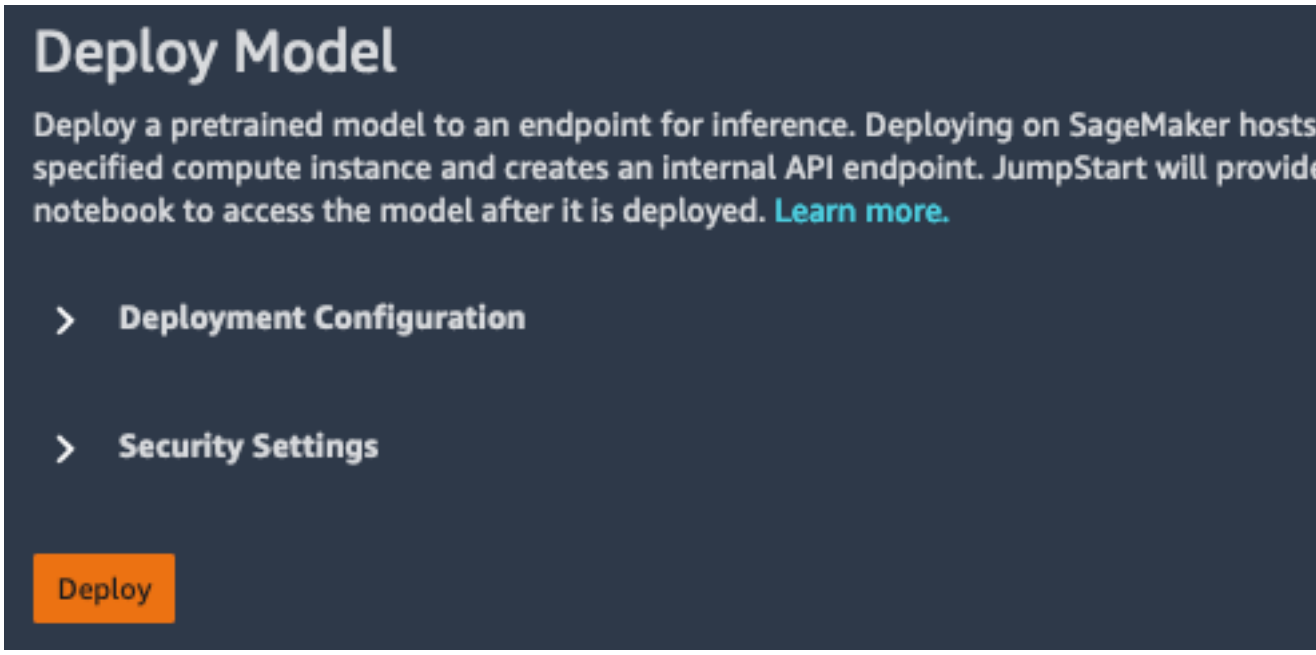
问题类型	通过预训练的模型支持推理	可在自定义数据集上训练	支持的框架	示例笔记本
图像分类	是	是	PyTorch，TensorFlow	简介 JumpStart -图像分类
对象检测	是	是	PyTorch、TensorFlow、MxNet	简介 JumpStart -对象检测
语义分割	是	是	MXNet	简介 JumpStart -语义分割
实例分段	是	是	MXNet	简介 JumpStart -实例细分
嵌入图像	是	否	TensorFlow，MxNet	简介 JumpStart -图像嵌入
文本分类	是	是	TensorFlow	简介 JumpStart -文本分类
句子对分类	是	是	TensorFlow，拥抱脸	简介 JumpStart -句子对分类
解答的问题	是	是	PyTorch	简介 JumpStart — 问题回答
命名实体识别	是	否	拥抱面部	简介 JumpStart -命名实体识别
文本总结	是	否	拥抱面部	简介 JumpStart -文本总结
文本生成	是	否	拥抱面部	简介 JumpStart -文本生成
机器翻译	是	否	拥抱面部	简介 JumpStart -机器翻译
文本嵌入	是	否	TensorFlow，MxNet	简介 JumpStart -文本嵌入
表格分类	是	是	lightgBm、catBoost、xgBoost、线性学习者	简介 JumpStart -表格分类-lightgBM、catBoost 简介 JumpStart -表格分类-xgBoost，线性学习者
表格回归	是	是	lightgBm、catBoost、xgBoost、线性学习者	简介 JumpStart -表格回归-lightgBm、catBoost 简介 JumpStart -表格回归-xgBoost，线性学习者

部署模型

当你从 JumpStart 部署模型时， SageMaker 托管模型并部署您可用于推理的终端节点。 JumpStart 还提供了一个示例笔记本，您可以在部署模型后用于访问模型。

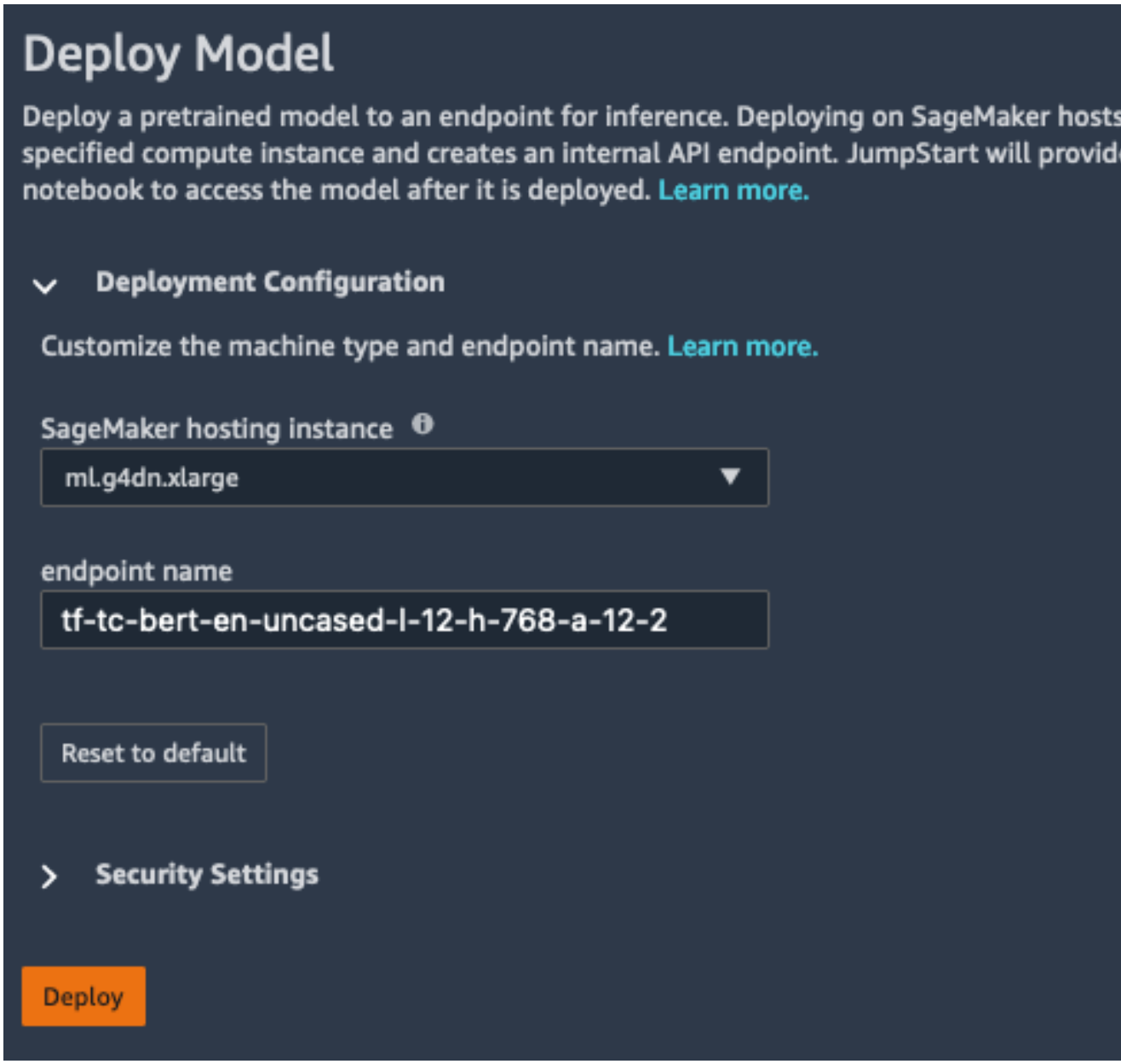
模型部署配置

选择模型后，部署模型窗格打开。选择部署配置来配置模型部署。

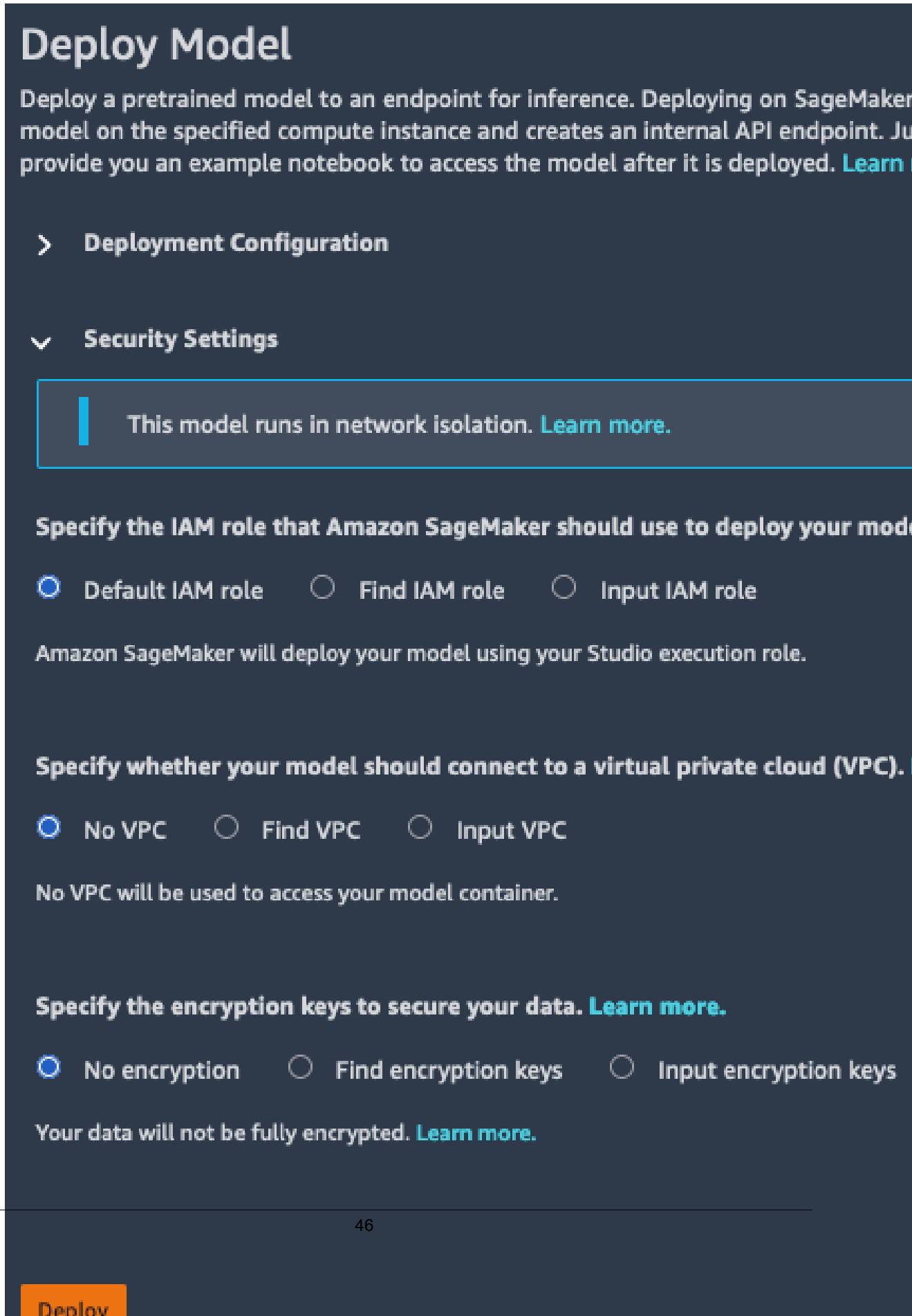


部署模型的默认实例类型取决于模型。实例类型是运行培训作业的硬件。在下面的示例中，`ml.g4dn.xlarge`实例是此特定 BERT 模型的默认值。

您还可以更改终端节点名称.



选择安全设置指定Amazon Identity and Access Management(IAM) 角色、Amazon Virtual Private Cloud (Amazon VPC) 和模型的加密密钥。


Deploy Model
Deploy a pretrained model to an endpoint for inference. Deploying on SageMaker
model on the specified compute instance and creates an internal API endpoint. Ju
provide you an example notebook to access the model after it is deployed. Learn
Deployment Configuration
Security Settings
This model runs in network isolation. Learn more.
Specify the IAM role that Amazon SageMaker should use to deploy your mod
Default IAM role
Find IAM role
Input IAM role
Amazon SageMaker will deploy your model using your Studio execution role.
Specify whether your model should connect to a virtual private cloud (VPC).
No VPC
Find VPC
Input VPC
No VPC will be used to access your model container.
Specify the encryption keys to secure your data. Learn more.
No encryption
Find encryption keys
Input encryption keys
Your data will not be fully encrypted. Learn more.

模型部署安全

使用 JumpStart 部署模型时，您可以为该模型指定 IAM 角色、Amazon VPC 和加密密钥。如果未为这些条目指定任何值：默认 IAM 角色是您的 Studio 运行时角色；使用默认加密；不使用任何 Amazon VPC。

IAM 角色

您可以选择作为培训作业和托管作业的一部分传递的 IAM 角色。 SageMaker 使用此角色访问训练数据和模型工件。如果您未选择 IAM 角色， SageMaker 使用 Studio 运行时角色部署模型。有关 IAM 角色的更多信息，请参阅适用于 Amazon SageMaker 的 Identity and Access Management (p. 2690).

您传递的角色必须有权访问模型所需的资源，并且必须包括以下所有内容。

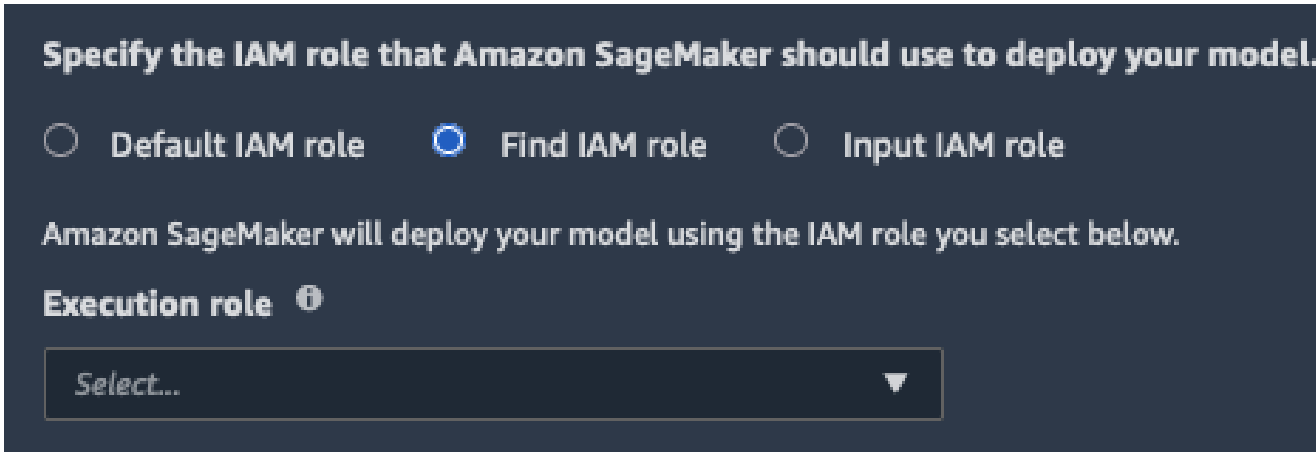
- 对于训练作业：CreateTrainingJob API：执行角色权限.
- 对于托管工作：CreateModel API：执行角色权限.

> **Note**
>
> 您可以缩小以下每个角色中授予的 Amazon S3 权限的范围。要执行此操作，请使用 Amazon Simple Storage Service (Amazon S3) 存储桶的 Amazon Resource Name (ARN) 和 JumpStart Amazon S3 存储桶。

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:ListMultipartUploadParts"
  ],
  "Resources": [
    "arn:aws:s3:<region>::bucket/jumpstart-cache-prod-<region>/*",
    "arn:aws:s3:<region>:<account>:bucket/*",
  ]
},{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
  ],
  "Resources": [
    "arn:aws:s3:<region>::bucket/jumpstart-cache-prod-<region>",
    "arn:aws:s3:<region>:<account>:bucket",
  ]
```

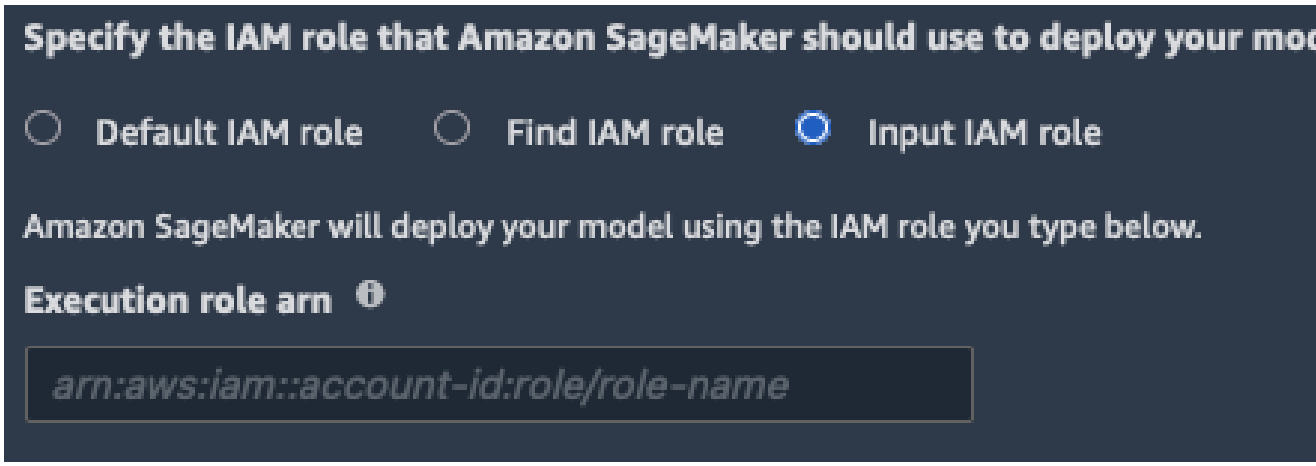
查找 IAM 角色

如果选择此选项，则必须从下拉列表中选择一个现有 IAM 角色。



输入 IAM 角色

如果选择此选项，则必须为现有 IAM 角色手动输入 ARN。如果您的 Studio 运行时角色或 Amazon VPC 阻止`iam:list*` 调用时，您必须使用此选项才能使用现有 IAM 角色。



Amazon VPC

全部 JumpStart 模型在网络隔离模式下运行。创建模型容器之后，不能再进行调用。您可以选择作为培训作业和托管工作的一部分传递的 Amazon VPC。 SageMaker 使用此 Amazon VPC 从 Amazon S3 存储桶中推送和提取资源。此 Amazon VPC 不同于限制从您的 Studio 实例访问公共互联网的亚马逊 VPC。有关 Studio Amazon VPC 的更多信息，请参阅Connect (连接) SageMaker VPC 中的 Studio 笔记本电脑到外部资源 (p. 2781).

您传递的 Amazon VPC 不需要访问公共互联网，但确实需要访问 Amazon S3。Amazon S3 的 Amazon VPC 终端节点必须允许至少访问模型所需的以下资源。

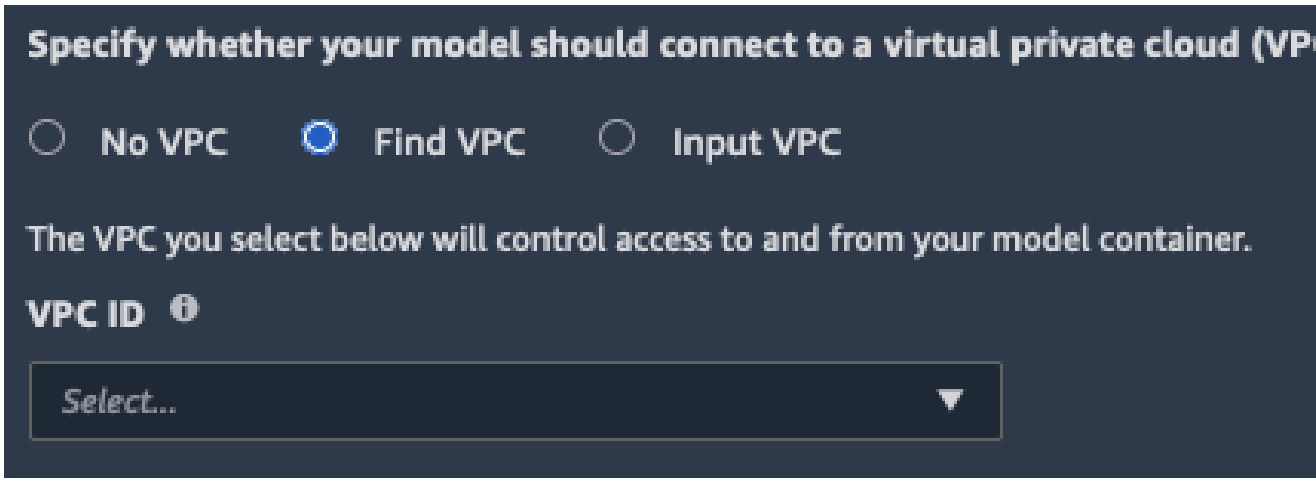
```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:ListMultipartUploadParts"
  ],
  "Resources": [
    "arn:aws:s3:<region>::bucket/jumpstart-cache-prod-<region>/*",
```

```
    "arn:aws:s3:<region>:<account>:bucket/*",
  ]
},{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
  ],
  "Resources": [
    "arn:aws:s3:<region>::bucket/jumpstart-cache-prod-<region>",
    "arn:aws:s3:<region>:<account>:bucket",
  ]
```

如果您没有选择亚马逊 VPC，则不会使用任何亚马逊 VPC。

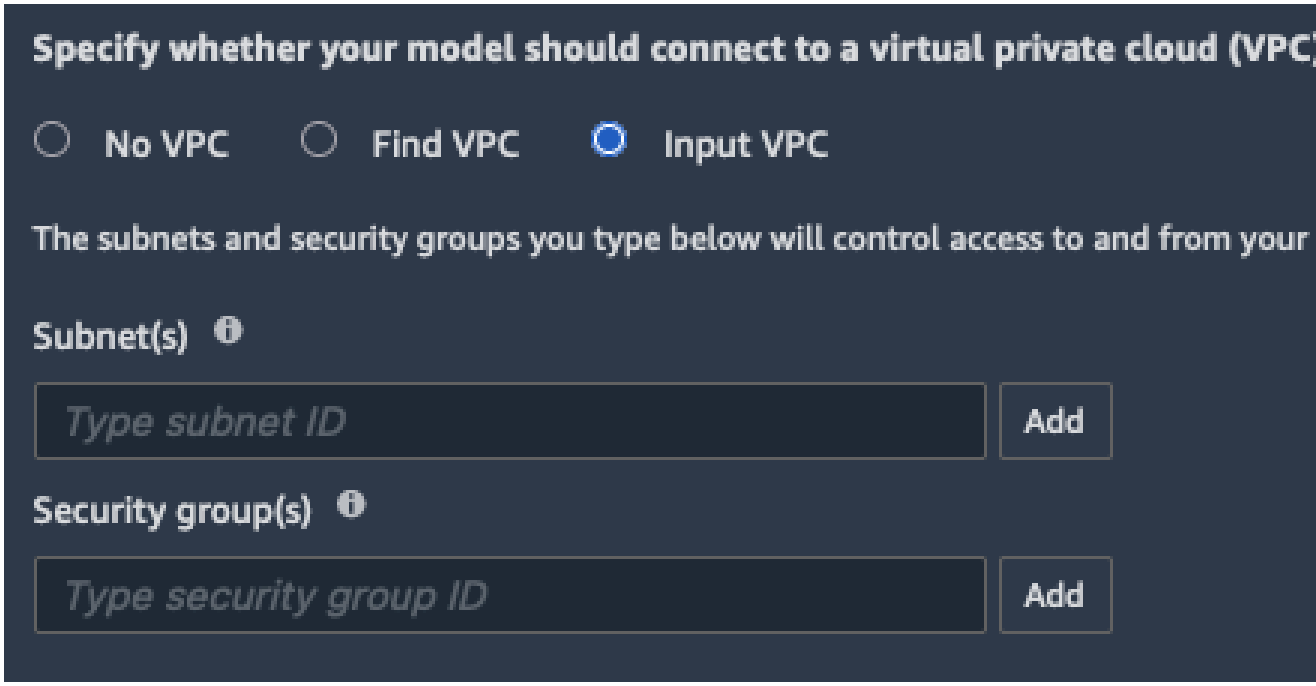
查找 VPC

如果选择此选项，则必须从下拉列表中选择现有的 Amazon VPC。选择 Amazon VPC 后，您必须为您的 Amazon VPC 选择子网和安全组。有关子网和安全组的更多信息，请参阅VPC 和子网的概述.



输入 VPC

如果选择此选项，则必须手动选择组成 Amazon VPC 的子网和安全组。如果您的 Studio 运行时角色或 Amazon VPC 阻止`ec2:list*`调用时，您必须使用此选项来选择子网和安全组。



加密密密钥

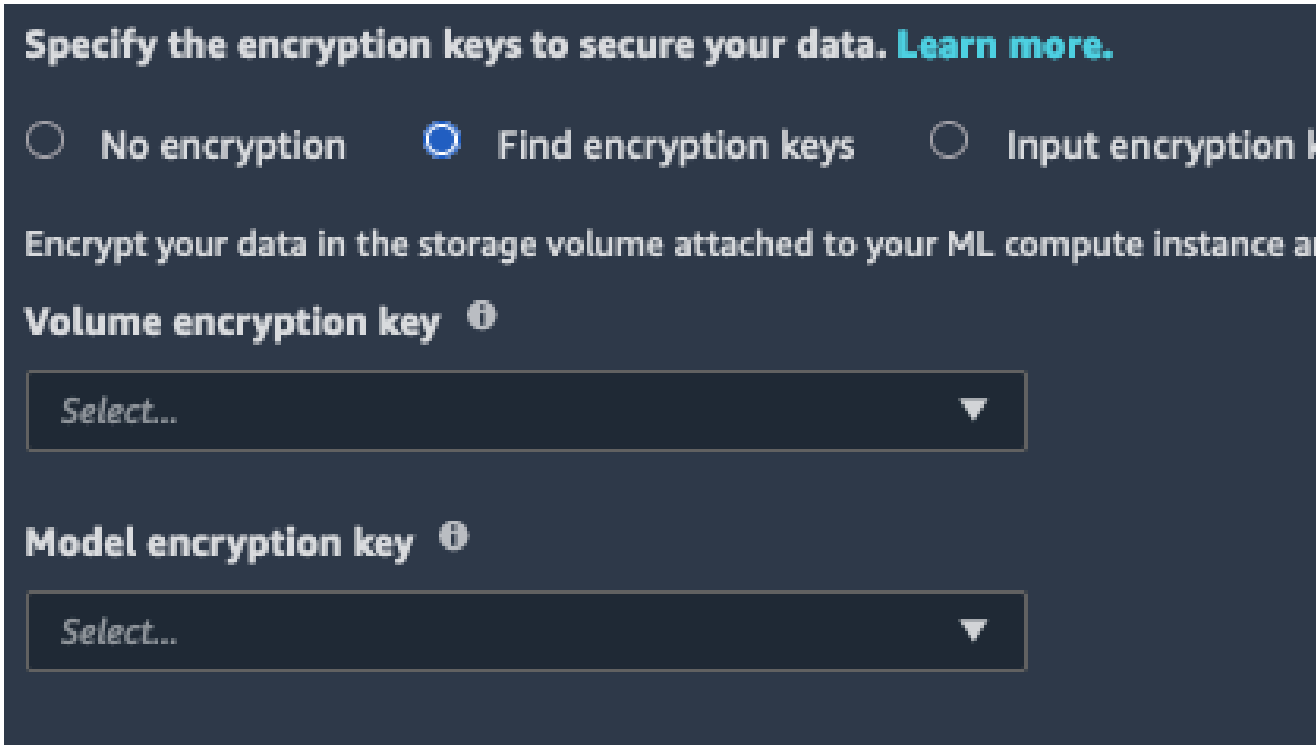
你可以选择Amazon KMS作为培训工作和托管工作的一部分传递的关键。 SageMaker 使用此密钥加密容器的 Amazon EBS 卷，并使用 Amazon S3 中重新打包的模型来托管作业和培训作业的输出。有关 的更多信息Amazon KMS钥匙，请参阅Amazon KMS密钥.

您传递的密钥必须信任您传递的 IAM 角色。如果您没有指定 IAM 角色，则Amazon KMS密钥必须信任你的 Studio 运行时角色。

如果你没有选择Amazon KMS键， SageMaker 为 Amazon EBS 卷中的数据和 Amazon S3 工件提供默认加密。

查找加密密钥

如果选择此选项，则必须选择现有的Amazon KMS从下拉列表中显示密钥。



输入加密密钥

如果选择此选项，则必须手动输入Amazon KMS钥匙。如果您的 Studio 执行角色或 Amazon VPC 阻止`kms:list*` 呼叫，你必须使用此选项来选择现有Amazon KMS钥匙。

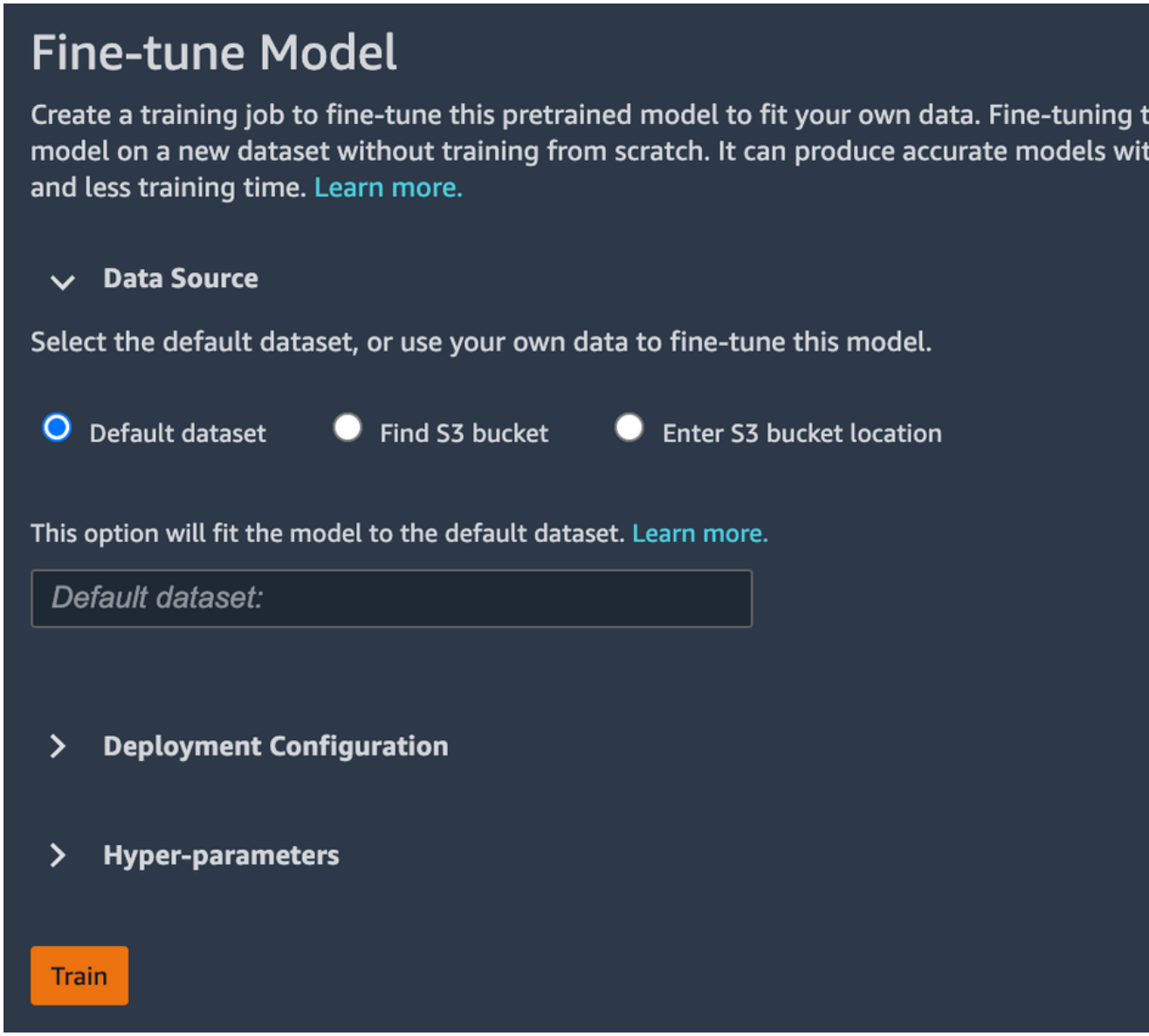
微调模型

微调在新数据集上训练预先训练的模型，而无需从头开始训练。这个过程也称为转移学习，可以生成具有较小数据集和更短训练时间的准确模型。

微调数据源

当您微调模型时，您可以使用默认数据集或选择位于 Amazon S3 存储桶中的自己的数据。

要浏览可用的存储桶，请选择查找 S3 存储桶. 这些存储桶受用于设置 Studio 帐户的权限的限制。您还可以通过选择以下方式指定 Amazon S3 URI输入 Amazon S3 存储桶位置.



Tip

要了解如何格式化存储桶中的数据，请选择了解更多. 模型的描述部分包含有关于输入和输出的详细信息。

对于文本模型：

- 存储桶必须有 data.csv 文件。
- 第一列必须是类别标签的唯一整数。例如：`1、2、3、4、n`
- 第二列必须是字符串。
- 第二列应包含与模型的类型和语言匹配的相应文本。

对于视觉模型：

- 存储桶的子目录必须与类数相同。
- 每个子目录都应以 .jpg 格式包含属于该类的图像。

> **Note**
>
> Amazon S3 存储桶必须位于同一位置Amazon Web Services 区域你跑的地方 SageMaker 因为 Studio SageMaker 不允许跨区域请求。

微调部署配置

推荐使用 p3 系列作为深度学习训练最快的系列，建议用于微调模型。下图显示了每种实例类型中 GPU 的数量。还有其他可供选择的选项，包括 p2 和 g4 实例类型。

实例类型	GPU
p3.2xlarge	1
p3.8xlarge	4
p3.16xlarge	8
p3dn.24xlarge	8

超参数

您可以自定义用于微调模型的训练作业的超参数。

如果您在不更改超参数的情况下将默认数据集用于文本模型，则会得到几乎相同的模型。对于视觉模型，默认数据集与用于训练预训练模型的数据集不同，因此您的模型因此不同。

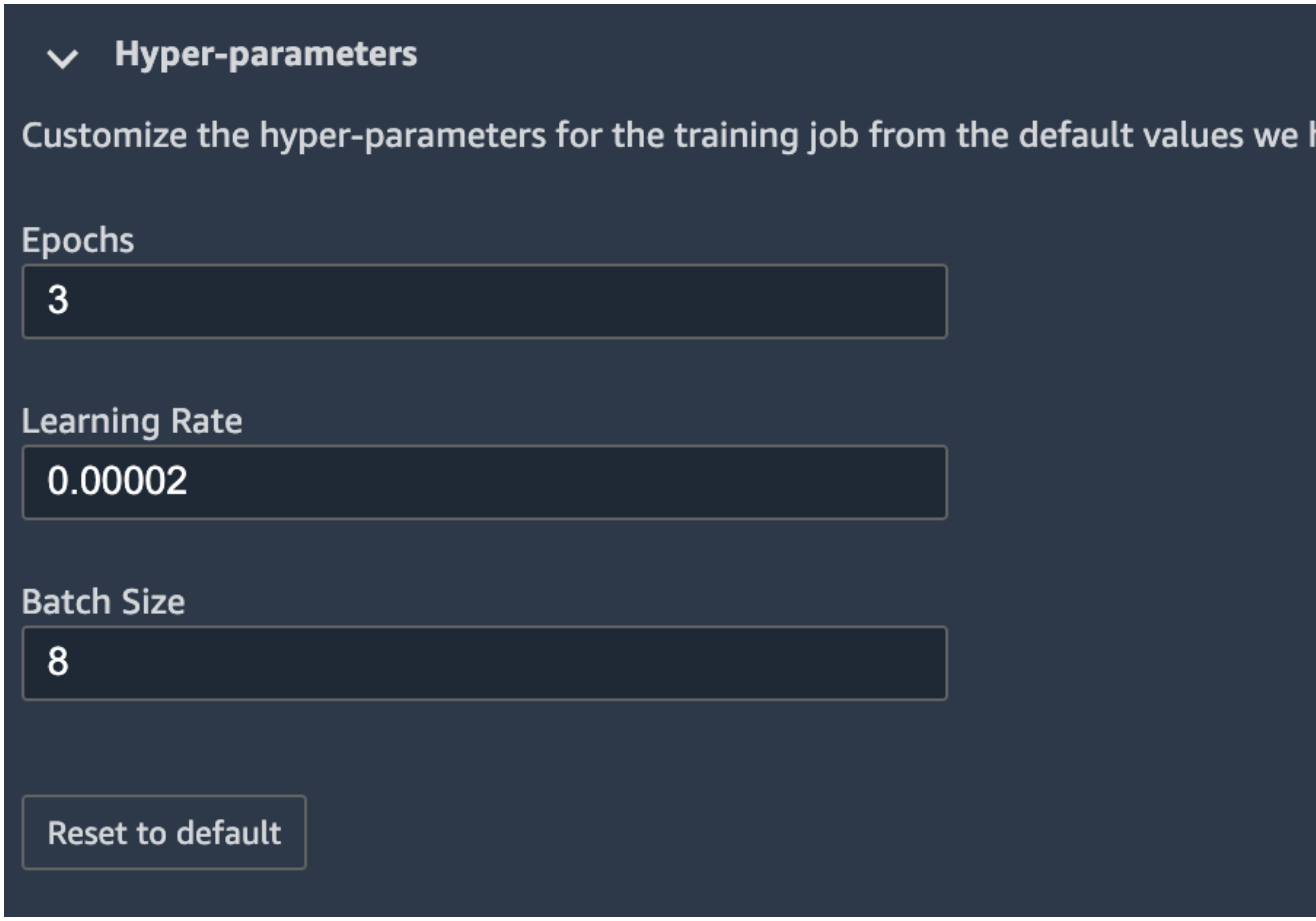
您有以下超参数选项：

- Epoch— 一个时代是一个循环遍历整个数据集。多个时间间隔完成一个批次，多个批次最终完成了一个纪元。运行多个纪元，直到模型的准确度达到可接受的水平，或者当错误率降至可接受的水平以下时。
- Learning rate (学习速率)— 值应在各个纪元之间更改的金额。随着模型的优化，其内部权重将被推动，并检查错误率以确定模型是否有所改善。典型的学习率为 0.1 或 0.01，其中 0.01 是一个小得多的调整，可能会导致训练需要很长时间才能收敛，而 0.1 要大得多，可能会导致训练过度。这是为训练模型而可能会调整的主要超参数之一。请注意，对于文本模型，较小的学习率（BERT 为 5e-5）可以生成更准确的模型。
- 批处理大小— 将在每个时间间隔内从数据集中选择的记录数，以发送到 GPU 进行训练。

  在图像示例中，每个 GPU 可能会发送 32 张图像，所以 32 张是批量大小。如果您选择具有多个 GPU 的实例类型，则该批次除以 GPU 的数量。建议的批量大小因您使用的数据和模型而异。例如，优化图像数据的方式与处理语言数据的方式不同。

  在部署配置部分的实例类型图表中，您可以看到每种实例类型的 GPU 数量。从标准推荐的批量大小开始（例如，对于视觉模型，32）。然后，将此值乘以您选择的实例类型中的 GPU 数量。例如，如果您使用

的是p3.8xlarge，这将是 32（批量大小）乘以 4（GPU），总计 128 个，因为批量大小会根据 GPU 的数量进行调整。对于 BERT 这样的文本模型，请尝试从 64 的批次大小开始，然后根据需要减少。



训练输出

当微调过程完成后，JumpStart 提供有关模型的信息：父模型、培训作业名称、培训作业亚马逊资源名称 (ARN)、培训时间和输出路径。输出路径是您可以在 Amazon S3 存储桶中找到新模型的位置。文件夹结构使用您提供的模型名称，模型文件位于/output子文件夹，它总是被命名model.tar.gz.

示例：s3://bucket/model-name/output/model.tar.gz

亚马逊 SageMaker JumpStart 行业

使用 SageMaker JumpStart 行业解决方案、模型和示例笔记本电脑可以了解 SageMaker 功能和功能通过精心策划的一步式解决方案和关于行业中心机器学习 (ML) 问题的示例笔记本。笔记本还介绍如何使用 SageMaker JumpStart 行业 Python SDK 可增强行业文本数据并微调预训练模型。

主题

- 亚马逊 SageMaker JumpStart 行业 Python 开发工 (p. 55)
- 亚马逊 SageMaker JumpStart 行业解决 (p. 55)
- 亚马逊 SageMaker JumpStart 行业模型 (p. 55)
- 亚马逊 SageMaker JumpStart 行业示例笔记 (p. 57)
- 亚马逊 SageMaker JumpStart 行业博客文章 (p. 58)

亚马逊 SageMaker JumpStart 行业 Python 开发工

SageMaker JumpStart 提供处理工具，用于管理行业数据集并通过名为 SageMaker JumpStart 行业 Python SDK。有关 SDK 的详细 API 文档，以及详细了解如何处理和增强行业文本数据集以提高 state-of-the-art 上的模型 SageMaker JumpStart，请参阅SageMaker JumpStart 行业 Python SDK 开源文档.

亚马逊 SageMaker JumpStart 行业解决

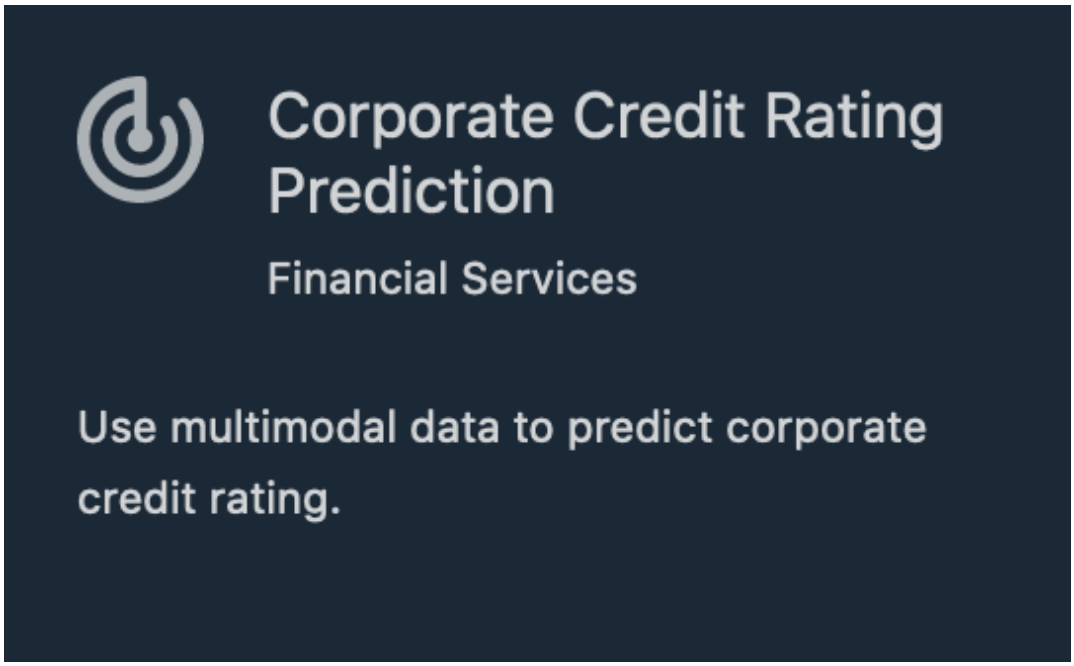
SageMaker JumpStart 行业提供以下解决方案笔记本：

- 企业信用评级预测 — 金融服务

该 SageMaker JumpStart 行业解决方案为文本增强型企业信用评级模型提供了模板。它展示了如何基于数字特征（在本例中为奥特曼著名的 5 财务比率）与美国证券交易委员会申报文件的文本结合使用模型，以实现信用评级预测的改善。除了 5 个奥特曼比率之外，您还可以根据需要添加更多变量或设置自定义变量。此解决方案笔记本展示了 SageMaker JumpStart 行业 Python SDK 有助于处理 SEC 申报文件中文本的 NLP 评分。此外，该解决方案还演示了如何使用增强型数据集训练模型以实现 best-in-class 模型，将模型部署到 SageMaker 终端节点进行生产，并实时接收改进的预测。

> Note
>
> 该解决方案笔记本仅用于演示目的。不应将它们作为财务或投资建议依赖。

以下屏幕截图显示了通过 SageMaker Studio 上的 JumpStart 页面。



> Note
>
> 这些区域有： SageMaker JumpStart 行业解决方案、模型卡和示例笔记本只能通过托管和运行 SageMaker Studio。登录到SageMaker 控制台，然后启动 SageMaker Studio。有关如何找到解决方案卡的更多信息，请参阅上一主题：SageMaker JumpStart.

亚马逊 SageMaker JumpStart 行业模型

SageMaker JumpStart 行业提供以下预培训强大优化的 BERT 方法（RoberTA）型号：

- 罗伯塔-秒基
- 罗伯塔-Sec-Wiki-Base
- 罗伯塔-秒大
- 罗伯塔-sec-Wiki-大

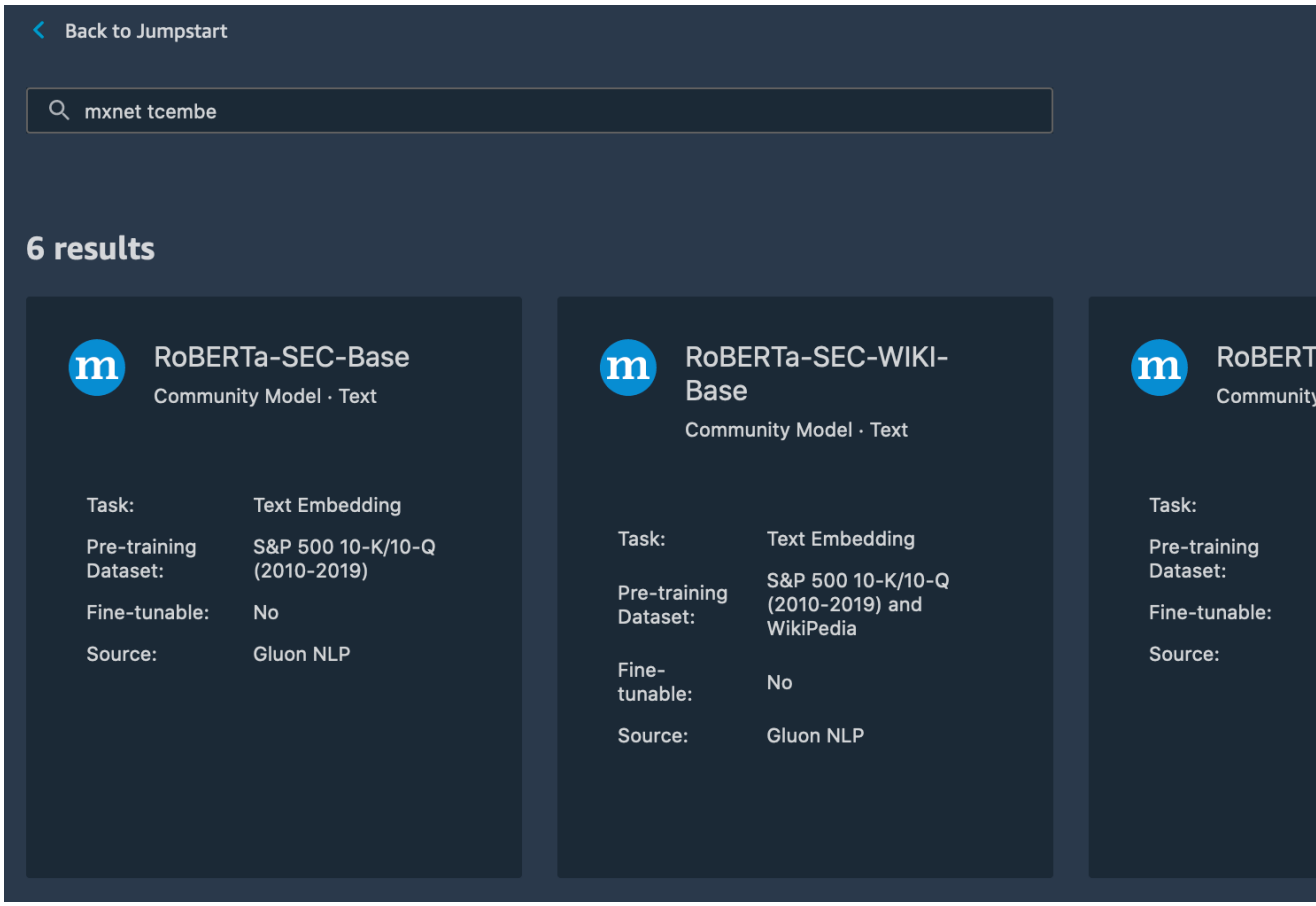
罗伯塔-秒基和罗伯塔-秒大型模型是基于GluonNLP 的 Roberta 模型并对标准普尔 500 证券交易委员会 2010 年十年代（2010 年至 2019 年）10-K/10-Q 报告进行了预培训。除此之外，SageMaker JumpStart 行业还提供了两个罗伯塔版本，罗伯塔-秒维基和罗伯塔-秒维基大型版本，它们在美国证券交易委员会的文件和维基百科通用文本方面进行了预培训。

通过部署模型卡 SageMaker JumpStart，您将能够访问他们对应的笔记本。配对的笔记本将引导您了解如何针对多模数据集的特定分类任务进行预训练的模型进行微调，这些任务由 SageMaker JumpStart 行业 Python SDK。

Note

该模型笔记本仅用于演示目的。不应将它们作为财务或投资建议依赖。

以下屏幕截图显示了通过 SageMaker 提供的预训练模型卡。 JumpStart Studio 上的页面。



Note

这些区域有： SageMaker JumpStart 行业解决方案、模型卡和示例笔记本只能通过托管和运行 SageMaker Studio。登录到SageMaker 控制台，然后启动 SageMaker Studio。有关如何找到模型卡的更多信息，请参阅上一主题：SageMaker JumpStart.

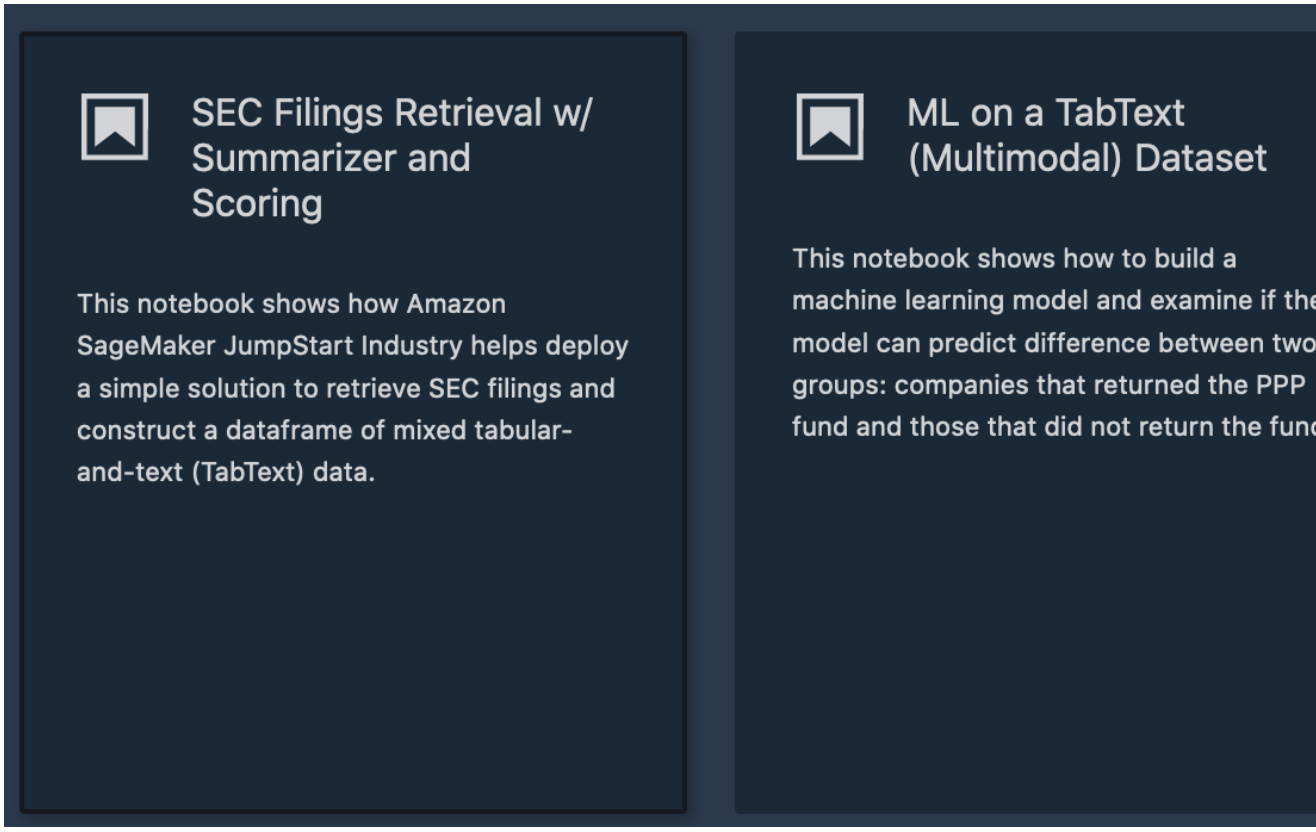
亚马逊 SageMaker JumpStart 行业示例笔记

SageMaker JumpStart 行业提供了解决以行业为中心的 ML 问题的以下实践示例：

- SEC 文件检索 W/ 摘要器和评分— 此示例介绍如何使用 SageMaker JumpStart 用于处理 SEC 申报的行业 Python SDK，例如基于 NLP 分数类型及其相应词汇表的文本总结和评分文本。要预览此笔记本的内容，请参阅根据 SEC 申报和 NLP 分数简单构建多模态数据集
- ML 在 TabText （多模式）数据集— 此示例说明如何将不同类型的数据集合成一个名为 TabText 并执行多模式 ML。要预览此笔记本的内容，请参阅上的 Machine Learning TabText Dataframe — 基于薪水保护计划的示例
- SEC 申报数据上的多类 ML— 此示例演示如何训练 AutoGluon 对多模式 (tabText) 数据集进行多类别分类任务的 NLP 模型，这些数据集是从 SEC 申报文件中精选的。根据 MDNA 文本列将 SEC 10K/Q申请分类为行业代码

> **Note**
>
> 示例笔记本仅用于演示目的。不应将它们作为财务或投资建议依赖。

以下屏幕截图显示了通过 SageMaker 提供的示例笔记本卡。 JumpStart Studio 上的页面。



> **Note**
>
> 这些区域有： SageMaker JumpStart 行业解决方案、模型卡和示例笔记本只能通过托管和运行 SageMaker Studio。登录到SageMaker 控制台，然后启动 SageMaker Studio。有关如何查找示例笔记本的更多信息，请参阅上一主题：SageMaker JumpStart.

要预览示例笔记本的内容，请参阅教程 — 财务中的SageMaker JumpStart 行业 Python SDK 文档.

亚马逊 SageMaker JumpStart 行业博客文章

对于彻底应用使用 SageMaker JumpStart 行业解决方案、模型、示例和 SDK，请参阅以下博客文章：

- 在亚马逊使用预训练的财务语言模型进行转移学习 SageMaker JumpStart
- 在亚马逊中使用多模式机器学习使用 SEC 文本进行评分 SageMaker JumpStart

Amazon 入门 SageMaker 笔记本实例

机器学习 (ML) 从业者使用 Amazon 的最佳方法之一 SageMaker 是使用训练和部署 ML 模型 SageMaker 笔记本实例。这些区域有： SageMaker 笔记本实例通过在 Amazon Elastic Compute Cloud (Amazon EC2) 上启动 Jupyter 服务器以及通过以下软件包提供预配置的内核，帮助创建环境：Amazon SageMaker Python 开发工具包，Amazon SDK for Python (Boto3)、Amazon Command Line Interface(Amazon CLI)、Conda、Pandas、深度学习框架库以及其他用于数据科学和机器学习的图书馆。

使用的 Machine Learning SageMaker Python 开发工具包

训练、验证、部署和评估中的 ML 模型 SageMaker 笔记本实例，请使用 SageMaker Python 开发工具包。这些区域有： SageMaker Python 开发工具包摘要Amazon SDK for Python (Boto3)和 SageMaker API 操作。它使您能够与其他人集成并协调其他Amazon服务，例如用于保存数据和模型构件的 Amazon Simple Storage Service (Amazon S3)、用于导入和维护 ML 模型的 Amazon Elastic Compute Cloud (Amazon EC2)。

您也可以利用 SageMaker 可帮助您处理完整机器学习周期的每个阶段的各个阶段的功能：数据标签、数据预处理、模型训练、模型部署、预测性能评估以及监控生产中模型的质量。

如果您是第一次 SageMaker 用户，我们建议你使用 SageMaker Python SDK，遵循 end-to-end ML 教程。要查找开源文档，请参阅亚马逊 SageMaker Python 开发工具包.

教程概述

本入门教程将引导您完成如何创建 SageMaker notebook 实例，打开一个带有预配置内核的 Jupyter 笔记本，并使用 Conda 环境进行机器学习，然后启动 SageMaker 要运行的会话 end-to-end ML 循环。您将了解如何将数据集保存到与 SageMaker 会话自动配对的默认 Amazon S3 存储桶，向 Amazon EC2 提交 ML 模型的培训作业，以及通过 Amazon EC2 托管或批量推断来部署训练模型以进行预测。

本教程明确显示了从 SageMaker 内置模型池。您可以使用美国成人普查数据集，然后你评估受过训练的人的表现 SageMaker 关于预测个人收入的 xgBoost 模型。

- SageMaker XGBoost— 该XGBoost模型适用于 SageMaker 环境并预先配置为 Docker 容器。SageMaker 提供了一套内置算法准备好使用 SageMaker 功能。要了解有关哪些 ML 算法适应 SageMaker 的更多信息，请参阅选择算法和使用 Amazon SageMaker 内置算法. 对于 SageMaker 内置算法 API 操作，请参阅第一方算法中的亚马逊 SageMaker Python 开发工具包.
- 成人普查数据集— 来自1994 年普查局数据库由 Ronny Kohavi 和 Barry Becker (数据挖掘和可视化，硅图形) 。这些区域有： SageMaker XgBoost 模型是使用此数据集进行训练的，以预测个人年收入是否超过 50,000 美元或更少。

主题

- 第 1 步：创建 Amazon SageMaker 笔记本实例 (p. 59)
- 第 2 步：创建 Jupyter 笔记本 (p. 60)

- 第 3 步：下载、浏览和转换数据集 (p. 60)
- 第 4 步：训练模型 (p. 64)
- 第 5 步：将模型部署到 Amazon EC2 (p. 68)
- 第 6 步：评估模型 (p. 70)
- 步骤 7：清除 (p. 73)

第 1 步：创建 Amazon SageMaker 笔记本实例

Amazon SageMaker 笔记本实例是一个完全托管的机器学习 (ML) Amazon Elastic Compute Cloud (Amazon EC2) 计算实例，它运行 Jupyter 笔记本应用程序。您可以使用笔记本实例创建和管理 Jupyter 笔记本以预处理数据以及训练和部署机器学习模型。

创建 SageMaker 笔记本实例

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 选择笔记本实例，然后选择创建笔记本实例.
3. 在 Create notebook instance (创建笔记本实例) 页面上，提供以下信息（如果未提到某个字段，请保留默认值）：

   a. 对于 Notebook instance name (笔记本实例名称)，请键入您的笔记本实例的名称。

   b. 对于 Instance type (实例类型)，选择 `ml.t2.medium`。这是笔记本实例支持的成本最低的实例类型，并且足以完成本练习。如果`ml.t2.medium`实例类型在您当前不可用Amazon区域，选择`ml.t3.medium`.

   c. 适用于IAM 角色，选择创建新角色，然后选择创建角色. 此 IAM 角色自动获取访问任何具有的 S3 存储桶的权限。`sagemaker`在名字里。它通过获取这些权限`AmazonSageMakerFullAccess`策略，哪 SageMaker 附加到角色。

      Note

      如果您想授予 IAM 角色访问 S3 存储桶的权限`sagemaker`在名称中，您需要附加`S3FullAccess`策略或将特定 S3 存储桶的权限限制为 IAM 角色。有关向 IAM 角色添加存储桶策略的更多信息和示例，请参阅存储桶策略示例.

   d. 选择创建笔记本实例。

      在几分钟内 SageMaker 启动 ML 计算实例 (在这种情况下为笔记本实例)，并向其附加 5 GB Amazon EBS 存储卷。笔记本实例有一个预配置的 Jupyter 笔记本服务器， SageMaker 和 AmazonSDK 库和一组 Anaconda 库。

      有关创建 SageMaker 笔记本实例，请参阅创建笔记本实例.

（可选）更改 SageMaker 笔记本实例设置

如果您想更改机器学习计算实例类型或 Amazon EBS 存储的大小 SageMaker 已创建的笔记本实例，您可以编辑笔记本实例设置。

要更改和更新 SageMaker 笔记本实例类型和 EBS 卷

1. 在存储库的笔记本实例中的页面 SageMaker 控制台中，选择笔记本实例。
2. 选择操作，选择停止，然后等到笔记本实例完全停止。
3. 在笔记本实例状态更改为之后Stopped，选择操作，然后选择更新设置.

   a. 适用于笔记本实例类型，请选择不同的 ML 实例类型。

b. 适用于卷大小（GB）中，键入另一个整数以指定新的 EBS 卷大小。

Note

因此，EBS 存储卷是加密的。 SageMaker 无法确定该卷上的可用空间量。因此，您可以在更新笔记本实例时增加卷大小，但无法减小卷大小。如果要减小正在使用的 ML 存储卷的大小，请创建一个具有所需大小的新笔记本实例。

4. 在页面底部，选择更新笔记本实例.
5. 更新完成后，启动具有新设置的笔记本实例。

有关更新的更多信息 SageMaker 笔记本实例设置，请参阅更新笔记本实例.

（可选）的高级设置 SageMaker 笔记本实例

以下教程视频演示如何设置和使用。 SageMaker 通过笔记本实例 SageMaker 具有高级选项的控制台，例如 SageMaker 生命周期配置和导入 GitHub 存储库。（长度：26:04）

有关的完整文档 SageMaker 笔记本实例，请参阅使用 Amazon SageMaker 笔记本实例.

第 2 步：创建 Jupyter 笔记本

要开始编写脚本以进行训练和部署模型，请在 SageMaker 笔记本实例。使用 Jupyter 笔记本，您可以在访问 SageMaker 功能和Amazon基础设施。

创建 Jupyter 笔记本

1. 按如下方式打开笔记本实例：
   a. 登录到 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
   b. 在存储库的笔记本实例页面上，通过选择以下任一方法打开笔记打开 JupyterLab(对于) JupyterLab 或接口打开 Jupyter对于经典的 Jupyter 视图。

      Note

      如果笔记本实例状态显示Pending中的状态列中，您的笔记本实例仍在创建中。状态将更改为InService笔记本实例准备就绪时，将使用该实例。
2. 按下面所示创建笔记本：
   - 如果你在 JupyterLab 查看，在File (文件)菜单，选择New，然后选择笔记本. 对于 Select Kernel (选择内核)，请选择 conda_python3。该预装环境包括默认的 Anaconda 安装和 Python 3。
   - 如果在经典的 Jupyter 视图中打开了笔记本，请在文件选项卡上，选择New，然后选择conda_python3. 该预装环境包括默认的 Anaconda 安装和 Python 3。
3. 按如下方式保存笔记本：
   - 在 JupyterLab 查看，选择File (文件)，选择将笔记本另存为...，然后重命名笔记本。
   - 在 Jupyter 经典视图中，选择File (文件)，选择另存为...，然后重命名笔记本。

第 3 步：下载、浏览和转换数据集

在此步骤中，你加载成人普查数据集使用到你的笔记本实例SHAP（SHAPLEY 加法解释）库，请查看数据集，转换数据集并将数据集上传到 Amazon S3。

要运行以下示例例子，请将示例代码粘贴到笔记本实例的单元格中。

使用 SHAP 加载成人普查数据集

使用 SHAP 库，导入成人人口普查数据集，如下所示：

```
import shap
X, y = shap.datasets.adult()
X_display, y_display = shap.datasets.adult(display=True)
feature_names = list(X.columns)
feature_names
```

Note

如果当前的 Jupyter 内核没有 SHAP 库，请通过运行以下命令进行安装conda命令：

```
%conda install -c conda-forge shap
```

如果您使用的是 JupyterLab，则必须在安装和更新完成后手动刷新内核。运行以下 iPython 脚本关闭内核（内核将自动重启）：

```
import IPython
IPython.Application.instance().kernel.do_shutdown(True)
```

这些区域有：`feature_names`list 对象应返回以下功能列表：

```
['Age',
 'Workclass',
 'Education-Num',
 'Marital Status',
 'Occupation',
 'Relationship',
 'Race',
 'Sex',
 'Capital Gain',
 'Capital Loss',
 'Hours per week',
 'Country']
```

Tip

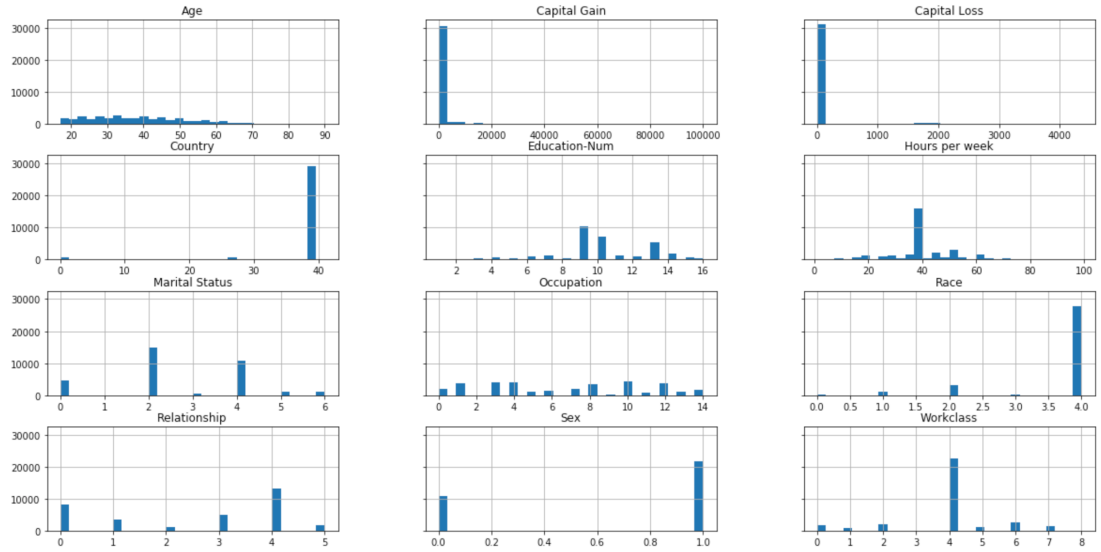
如果您从未标签的数据开始，则可以使用亚马逊 SageMaker Ground Truth 可在几分钟内创建数据标签工作流程。要了解更多信息，请参阅标记数据.

概述数据集

运行以下脚本以显示数据集的统计概览和数字要素的直方图。

```
display(X.describe())
hist = X.hist(bins=30, sharey=True, figsize=(20, 10))
```

	Age	Workclass	Education-Num	Marital Status	Occupation	Relationship	Race	Sex	Capital Gain	Capital Loss	Hours per week	Country
count	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581646	3.868892	10.080679	2.611836	6.572740	2.494518	3.665858	0.669205	1077.649170	87.303833	40.437454	36.718866
std	13.640442	1.455960	2.572562	1.506222	4.228857	1.758232	0.848806	0.470506	7385.911621	403.014771	12.347933	7.823782
min	17.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
25%	28.000000	4.000000	9.000000	2.000000	3.000000	0.000000	4.000000	0.000000	0.000000	0.000000	40.000000	39.000000
50%	37.000000	4.000000	10.000000	2.000000	7.000000	3.000000	4.000000	1.000000	0.000000	0.000000	40.000000	39.000000
75%	48.000000	4.000000	12.000000	4.000000	10.000000	4.000000	4.000000	1.000000	0.000000	0.000000	45.000000	39.000000
max	90.000000	8.000000	16.000000	6.000000	14.000000	5.000000	4.000000	1.000000	99999.000000	4356.000000	99.000000	41.000000



Tip

如果您想使用需要清理和转换的数据集，可以使用亚马逊简化和简化数据预处理和要素工程 SageMaker DATA Wangler。要了解更多信息，请参阅使用亚马逊准备 ML 数据 SageMaker DATA Wrangler.

将数据集拆分为训练、验证和测试数据集

使用 Sklearn，将数据集拆分为一个训练集和一个测试集。该数据集使用固定随机种子随机进行随机排序：80% 的数据集用于训练集，20% 用于测试集。

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
X_train_display = X_display.loc[X_train.index]
```

拆分培训集以分开验证集。75% 的培训集成为最终训练集，其余的是验证集。

```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25,
 random_state=1)
X_train_display = X_display.loc[X_train.index]
X_val_display = X_display.loc[X_val.index]
```

使用熊猫包，通过将数字要素与真实标签连接起来，明确对齐每个数据集。

```
import pandas as pd
train = pd.concat([pd.Series(y_train, index=X_train.index,
                             name='Income>50K', dtype=int), X_train], axis=1)
```

```
validation = pd.concat([pd.Series(y_val, index=X_val.index,
                                  name='Income>50K', dtype=int), X_val], axis=1)
test = pd.concat([pd.Series(y_test, index=X_test.index,
                                  name='Income>50K', dtype=int), X_test], axis=1)
```

检查数据集是否按预期进行拆分和结构化：

```
train
```

	Income>50K	Age	Workclass	Education-Num	Marital Status	Occupation	Relationship	Race	Sex	Capital Gain	Capital Loss	Hours per week	Country
10911	1	47.0	4	9.0	2	3	4	4	1	0.0	0.0	40.0	39
17852	0	31.0	4	13.0	2	7	4	3	1	0.0	0.0	36.0	26
29165	1	32.0	4	10.0	2	13	5	4	0	0.0	0.0	32.0	39
30287	0	58.0	4	9.0	2	3	4	2	1	0.0	0.0	40.0	39
24019	0	17.0	4	6.0	4	6	3	4	1	0.0	0.0	20.0	39
...	...	...	...	...	...	...	...	...	...	...	...	...	...
21168	0	43.0	4	8.0	2	14	4	4	1	0.0	0.0	40.0	39
6452	0	26.0	4	9.0	4	7	0	4	1	0.0	0.0	52.0	39
31352	0	32.0	7	14.0	2	10	4	4	1	0.0	0.0	50.0	39
6575	0	45.0	4	9.0	4	6	0	4	1	0.0	0.0	40.0	39
23608	0	23.0	4	9.0	4	1	1	4	0	0.0	0.0	40.0	39

19536 rows × 13 columns

```
validation
```

	Income>50K	Age	Workclass	Education-Num	Marital Status	Occupation	Relationship	Race	Sex	Capital Gain	Capital Loss	Hours per week	Country
16530	0	25.0	4	4.0	2	6	4	4	1	0.0	0.0	40.0	26
26723	0	41.0	6	9.0	2	5	5	4	0	0.0	0.0	40.0	39
3338	0	79.0	0	9.0	6	0	0	2	0	0.0	0.0	30.0	39
19367	1	43.0	2	15.0	2	10	4	4	1	15024.0	0.0	45.0	39
30274	0	51.0	5	9.0	4	12	2	4	1	0.0	0.0	40.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1604	0	46.0	7	9.0	2	13	4	4	1	0.0	0.0	40.0	39
5937	1	71.0	4	10.0	6	12	0	4	1	0.0	0.0	35.0	39
11034	0	36.0	4	9.0	5	14	2	4	1	0.0	0.0	60.0	26
2819	0	31.0	4	9.0	4	8	0	4	0	0.0	0.0	40.0	39
14152	1	37.0	4	10.0	2	12	4	4	1	0.0	0.0	50.0	11

6512 rows × 13 columns

```
test
```

	Income>50K	Age	Workclass	Education-Num	Marital Status	Occupation	Relationship	Race	Sex	Capital Gain	Capital Loss	Hours per week	Country
9646	0	62.0	6	4.0	6	8	0	4	0	0.0	0.0	66.0	39
709	0	18.0	4	7.0	4	8	2	4	1	0.0	0.0	25.0	39
7385	1	25.0	4	13.0	4	5	3	4	1	27828.0	0.0	50.0	39
16671	0	33.0	4	9.0	2	10	4	4	1	0.0	0.0	40.0	39
21932	0	36.0	4	7.0	4	7	1	4	0	0.0	0.0	40.0	39
...	...	...	...	...	...	...	...	...	...	...	...	...	...
5889	1	39.0	4	13.0	2	10	5	4	0	0.0	0.0	20.0	39
25723	0	17.0	4	6.0	4	12	3	4	0	0.0	0.0	20.0	39
29514	0	35.0	4	9.0	4	14	3	4	1	0.0	0.0	40.0	39
1600	0	30.0	4	7.0	2	3	4	4	1	0.0	0.0	45.0	39
639	1	52.0	6	16.0	2	10	4	4	1	0.0	0.0	60.0	39

6513 rows × 13 columns

将训练和验证数据集转换为 CSV 文件

将`train`和`validation`dataframe 对象到 CSV 文件，以匹配 xgBoost 算法的输入文件格式。

```
# Use 'csv' format to store the data
# The first column is expected to be the output column
train.to_csv('train.csv', index=False, header=False)
validation.to_csv('validation.csv', index=False, header=False)
```

将数据集上载到 Amazon S3

使用 SageMaker 和 Boto3，将训练和验证数据集上传到默认的 Amazon S3 存储桶。S3 存储桶中的数据集将被 Amazon EC2 上计算优化的 SageMaker 实例用于培训。

以下代码为当前的代码设置默认 S3 存储桶 URI SageMaker 会话，创建一个新的`demo-sagemaker-xgboost-adult-income-prediction`文件夹，然后将训练和验证数据集上传到`data`子文件夹。

```
import sagemaker, boto3, os
bucket = sagemaker.Session().default_bucket()
prefix = "demo-sagemaker-xgboost-adult-income-prediction"

boto3.Session().resource('s3').Bucket(bucket).Object(
    os.path.join(prefix, 'data/train.csv')).upload_file('train.csv')
boto3.Session().resource('s3').Bucket(bucket).Object(
    os.path.join(prefix, 'data/validation.csv')).upload_file('validation.csv')
```

运行以下命令Amazon CLI以检查 CSV 文件是否已成功上传到 S3 存储桶。

```
! aws s3 ls {bucket}/{prefix}/data --recursive
```

它应返回以下输出：

```
2021-01-14 17:52:09     786285 demo-sagemaker-xgboost-adult-income-prediction/data/train.csv
2021-01-14 17:52:10     262122 demo-sagemaker-xgboost-adult-income-prediction/data/validation.csv
```

第 4 步：训练模型

这些区域有：亚马逊 SageMaker Python 开发工具包提供框架估计器和通用估计器来训练模型，同时协调机器学习 (ML) 生命周期，访问 SageMaker 功能以进行训练和Amazon基础设施，如 Amazon EElastic Container Registry (Amazon ECR)、Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Simple

Storage Service (Amazon S3)。有关 的更多信息 SageMaker 内置框架估计器和算法估计器，请参阅框架和第一方算法分别在亚马逊 SageMaker Python 开发工具包文档中）。

主题

- 选择训练算法 (p. 65)
- 创建并运行训练 Job (p. 65)

选择训练算法

要为数据集选择正确的算法，通常需要评估不同的模型，以找到最适合数据的模型。为简单起见，SageMakerXGBoost 算法 (p. 1731)在本教程中使用内置算法，而无需预先评估模型。

> Tip
>
> 如果您想 SageMaker 要为您的表格数据集找到合适的模型，请使用亚马逊 SageMaker 自动驾驶机器学习解决方案。有关更多信息，请参阅 使用亚马逊自动开发模型 SageMaker Autopilot (p. 170)。

创建并运行训练 Job

在你找出要使用哪种模型之后，开始构造 SageMaker 用于训练的估计器。本教程使用 xGBoost 内置算法进行 SageMaker 通用估计器。

运行模型训练作业

1. 导入亚马逊 SageMaker Python 开发工具包然后首先从当前的基本信息中检索 SageMaker 会话。

```
import sagemaker

region = sagemaker.Session().boto_region_name
print("Amazon Region: {}".format(region))

role = sagemaker.get_execution_role()
print("RoleArn: {}".format(role))
```

   这将返回以下信息：

   - `region`— 最新的AmazonSageMaker 笔记本实例运行的区域。
   - `role`— 笔记本实例所使用的 IAM 角色。

   > Note
   >
   > 检查 SageMaker 通过运行 Python SDK 版本`sagemaker.__version__`. 本教程的基础是`sagemaker>=2.20`. 如果 SDK 已过时，请运行以下命令安装最新版本：
   >
   > ```
   > ! pip install -qU sagemaker
   > ```
   >
   > 如果你在退出时运行此安装 SageMaker Studio 或笔记本实例，您需要手动刷新内核才能完成应用版本更新。

2. 使用`sagemaker.estimator.Estimator`类。在以下示例代码中，xgBoost 估计器被命名为`xgb_model`.

```
from sagemaker.debugger import Rule, rule_configs
from sagemaker.session import TrainingInput
```

```
s3_output_location='s3://{}/{}/{}'.format(bucket, prefix, 'xgboost_model')

container=sagemaker.image_uris.retrieve("xgboost", region, "1.2-1")
print(container)

xgb_model=sagemaker.estimator.Estimator(
    image_uri=container,
    role=role,
    instance_count=1,
    instance_type='ml.m4.xlarge',
    volume_size=5,
    output_path=s3_output_location,
    sagemaker_session=sagemaker.Session(),
    rules=[Rule.sagemaker(rule_configs.create_xgboost_report())]
)
```

构造 SageMaker 估计器中，指定以下参数：

- `image_uri`— 指定训练容器映像 URI。在此示例中， SageMaker xgBoost 训练容器 URI 是使用`sagemaker.image_uris.retrieve`.
- `role`— 该Amazon Identity and Access Management(IAM) 角色 SageMaker 用于代表您执行任务（例如，读取训练结果，从 Amazon S3 调用模型项目以及将训练结果写入到 Amazon S3）。
- `instance_count`和`instance_type`— 要用于模型训练的 Amazon EC2 ML 计算实例的类型和数量。在本培训练习中，您将使用单一的`ml.m4.xlarge`实例，它具有 4 个 CPU、16 GB 内存、一个亚马逊 Elastic Block Store (Amazon EBS) 存储和高网络性能。有关 EC2 计算实例类型的更多信息，请参阅Amazon EC2 实例类型. 有关计费的更多信息，请参阅亚马逊 SageMaker 定价.
- `volume_size`— 要附加到训练实例的 EBS 存储卷的大小（以 GB 为单位）。如果您使用，该数据必须足够大以存储训练数据`File`模式 ()`File`默认情况下模式打开）。
- `output_path`SageMaker 存储模型构件和训练结果的 S3 存储桶的路径。
- `sagemaker_session`— 管理与交互的会话对象 SageMaker API 操作和其他Amazon培训作业使用的服务。
- `rules`— 指定一个列表 SageMaker 调试程序内置规则。在此示例中，`create_xgboost_report()`规则创建一个 xgBoost 报告，提供有关训练进度和结果的见解。有关更多信息，请参阅 SageMaker 调试器 xgBoost 培训报告 (p. 1907)。

  Tip

  如果你想运行大型深度学习模型的分布式训练，例如卷积神经网络 (CNN) 和自然语言处理 (NLP) 模型，请使用 SageMaker 分发用于数据并行性或模型并行性。有关更多信息，请参阅 Amazon SageMaker 分布式培训库 (p. 1964)。

3. 通过调用 XGBoost 算法的超参数设置`set_hyperparameters`估计程序的方法。有关 XGBoost 超参数的完整列表，请参阅XGBoost 超参数 (p. 1738).

```
xgb_model.set_hyperparameters(
    max_depth = 5,
    eta = 0.2,
    gamma = 4,
    min_child_weight = 6,
    subsample = 0.7,
    objective = "binary:logistic",
    num_round = 1000
)
```

  Tip

  你也可以使用 SageMaker 超参数优化功能。有关更多信息，请参阅 使用 SageMaker 执行自动模型优化 (p. 1939)。

4. 使用`TrainingInput`类来配置用于训练的数据输入流。以下示例代码说明了如何配置`TrainingInput`要使用您上传到 Amazon S3 的训练和验证数据集的对象将数据集拆分为训练、验证和测试数据集 (p. 62)部分。

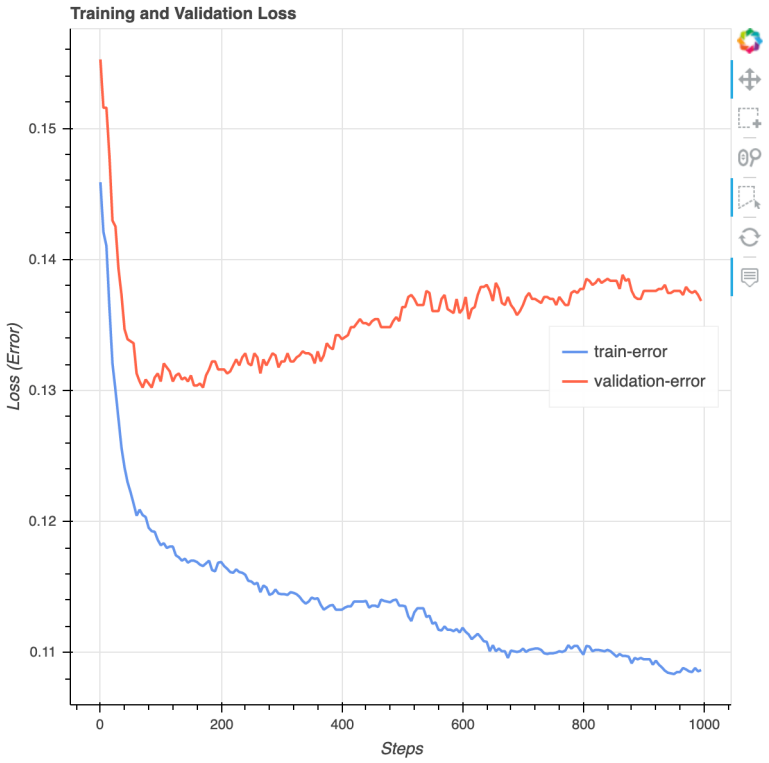
```
from sagemaker.session import TrainingInput

train_input = TrainingInput(
    "s3://{}/{}/{}".format(bucket, prefix, "data/train.csv"), content_type="csv"
)
validation_input = TrainingInput(
    "s3://{}/{}/{}".format(bucket, prefix, "data/validation.csv"), content_type="csv"
)
```

5. 要启动模型训练，请调用估算器的`fit`方法与训练和验证数据集一起使用。设置`wait=True`，`fit`方法显示进度日志，并等到训练完成。

```
xgb_model.fit({"train": train_input, "validation": validation_input}, wait=True)
```

有关模型训练的更多信息，请参阅使用 Amazon SageMaker 训练模型 (p. 7)。此教程培训作业最多可能需要 10 分钟。

培训工作完成后，您可以下载 xgBoost 培训报告和由 SageMaker 调试程序。XGBoost 训练报告为您提供了有关训练进度和结果的见解，例如与迭代相关的损失函数、功能重要性、混淆矩阵、准确度曲线以及其他训练的统计结果。例如，您可以从 xgBoost 培训报告中找到以下损失曲线，该报告清楚地表明存在过度拟合的问题。



运行以下代码以指定生成调试器培训报告的 S3 存储桶 URI，并检查报告是否存在。

```
rule_output_path = xgb_model.output_path + "/" + xgb_model.latest_training_job.job_name
 + "/rule-output"
```

```
! aws s3 ls {rule_output_path} --recursive
```

将调试器 xgBoost 训练和性能分析报告下载到当前工作空间：

```
! aws s3 cp {rule_output_path} ./ --recursive
```

运行以下 iPython 脚本以获取 xgBoost 培训报告的文件链接：

```
from IPython.display import FileLink, FileLinks
display("Click link below to view the XGBoost Training report",
 FileLink("CreateXgboostReport/xgboost_report.html"))
```

以下 iPython 脚本返回调试器性能分析报告的文件链接，该链接显示了 EC2 实例资源利用率、系统瓶颈检测结果和 python 操作分析结果的摘要和详细信息：

```
profiler_report_name = [rule["RuleConfigurationName"]
                        for rule in xgb_model.latest_training_job.rule_job_summary()
                        if "Profiler" in rule["RuleConfigurationName"]][0]
profiler_report_name
display("Click link below to view the profiler report", FileLink(profiler_report_name
+"/profiler-output/profiler-report.html"))
```

Tip

如果 HTML 报告没有在 JupyterLab 查看，您必须选择信任 HTML在报告的顶部。
要确定训练问题，例如过度拟合、消失渐变以及阻止模型收敛的其他问题，请使用 SageMaker 在设计和训练机器学习模型的同时，调试器并采取自动操作。有关更多信息，请参阅 亚马逊 SageMakerDebugger (调试程序) (p. 1783)。要查找模型参数的完整分析，请参阅亚马逊的可解释性 SageMaker Debugger (调试程序)例笔记本。

您现在有一个训练有素的 XGBoost 模型。 SageMaker 将模型项目存储在 S3 存储桶中。要查找模型对象的位置，请运行以下代码以打印`xgb_model`估算器：

```
xgb_model.model_data
```

Tip

要衡量机器学习生命周期的每个阶段（数据收集、模型训练和调整以及对部署用于预测的机器学习模型的监控）可能出现的偏见，请使用 SageMaker 澄清。有关更多信息，请参阅 亚马逊 SageMaker 澄清模型的可解释性 (p. 2072)。对于 end-to-end 例如，请参阅公平性和可解释性 SageMaker 澄清例笔记本。

第 5 步：将模型部署到 Amazon EC2

要获取预测，请使用 Amazon SageMaker 将模型部署到 Amazon EC2。

主题

- 将模型部署到 SageMaker 托管服务 (p. 69)
- （可选）使用 SageMaker 重用托管终端节点的预测器 (p. 69)
- （可选）使用 Batch 变换进行预测 (p. 69)

将模型部署到 SageMaker 托管服务

要使用 Amazon SageMaker 通过 Amazon EC2 托管模型，请部署您在其中训练的模型创建并运行训练 Job (p. 65)通过调用deploy的方法xgb_model估算器。当你打电话给deploy方法中，您必须指定要用于托管终端节点的 EC2 ML 实例的数量和类型。

```
import sagemaker
from sagemaker.serializers import CSVSerializer
xgb_predictor=xgb_model.deploy(
    initial_instance_count=1,
    instance_type='ml.t2.medium',
    serializer=CSVSerializer()
)
```

- initial_instance_count(int) — 要部署模型的实例数。
- instance_type(str) — 要操作已部署模型的实例类型。
- serializer(int) — 序列化各种格式的输入数据 (a NumPy 数组、列表、文件或缓冲区) 转换为 CSV 格式的字符串。我们使用这个是因为 xgBoost 算法接受 CSV 格式的输入文件。

这些区域有：deploy方法创建一个可部署的模型，配置 SageMaker 托管服务终端节点，并启动该终端节点以托管该模型。有关更多信息，请参阅 。SageMaker 通用估计器的部署类方法中的亚马逊 SageMaker Python 开发工具包. 检索由deploy方法中，运行以下代码：

```
xgb_predictor.endpoint_name
```

此选项应返回终端节点名称xgb_predictor. 终端节点名称的格式为"sagemaker-xgboost-YYYY-MM-DD-HH-MM-SS-SSS". 此终端节点在 ML 实例中保持活动状态，除非稍后关闭它，否则您可以随时进行即时预测。复制此端点名称并将其保存以重复使用并在其他地方进行实时预测 SageMaker Studio 或 SageMaker 笔记本实例。

> **Tip**
>
> 要了解有关编译和优化模型以便部署到 Amazon EC2 实例或边缘设备的更多信息，请参阅使用 Neo 编译并部署模型.

（可选）使用 SageMaker 重用托管终端节点的预测器

在将模型部署到终端节点后，您可以设置新的模型， SageMaker 通过对终端节点进行预测，并在任何其他笔记本电脑中持续进行实时预测。以下示例代码演示如何使用 SageMaker 预测变量类用于使用相同的终端节点设置新的预测变量对象。重复使用您用于xgb_predictor.

```
import sagemaker
xgb_predictor_reuse=sagemaker.predictor.Predictor(
    endpoint_name="sagemaker-xgboost-YYYY-MM-DD-HH-MM-SS-SSS",
    sagemaker_session=sagemaker.Session(),
    serializer=sagemaker.serializers.CSVSerializer()
)
```

这些区域有：xgb_predictor_reuse预测变量的行为与原始指标完全相同xgb_predictor. 有关更多信息，请参阅 。SageMaker 预测器中的类亚马逊 SageMaker Python 开发工具包.

（可选）使用 Batch 变换进行预测

您可以运行一次性批量推理作业，而不是在生产环境中托管终端节点，使用 SageMaker 批量转换。模型训练完成后，您可以将估计器扩展到transformer对象，它基于SageMaker 变压器类。批处理变换器从指定 S3 存储桶读取输入数据并进行预测。

运行批量转换任务

1. 运行以下代码将测试数据集的功能列转换为 CSV 文件并上传到 S3 存储桶：

```
X_test.to_csv('test.csv', index=False, header=False)

boto3.Session().resource('s3').Bucket(bucket).Object(
os.path.join(prefix, 'test/test.csv')).upload_file('test.csv')
```

2. 为批处理转换作业指定输入和输出的 S3 存储桶 URI，如下所示：

```
# The location of the test dataset
batch_input = 's3://{}/{}/test'.format(bucket, prefix)

# The location to store the results of the batch transform job
batch_output = 's3://{}/{}/batch-prediction'.format(bucket, prefix)
```

3. 创建一个变压器对象，指定最小数量的参数：`instance_count`和`instance_type`运行批处理转换作业的参数，以及`output_path`按如下所示保存预测数据：

```
transformer = xgb_model.transformer(
    instance_count=1,
    instance_type='ml.m4.xlarge',
    output_path=batch_output
)
```

4. 通过执行`transform()`的方法`transformer`对象如下所示：

```
transformer.transform(
    data=batch_input,
    data_type='S3Prefix',
    content_type='text/csv',
    split_type='Line'
)
transformer.wait()
```

5. 当批处理转换作业完成后， SageMaker 创建`test.csv.out`预测数据保存在`batch_output`path，应采用以下格式：`s3://sagemaker-<region>-111122223333/demo-sagemaker-xgboost-adult-income-prediction/batch-prediction`. 运行以下命令Amazon CLI要下载批处理转换作业的输出数据：

```
! aws s3 cp {batch_output} ./ --recursive
```

这应该创建`test.csv.out`文件位于当前工作目录下。你将能够看到基于 xgBoost 训练作业的逻辑回归预测的浮点值。

第 6 步：评估模型

现在，您已使用 Amazon SageMaker 训练并部署了一个模型，请评估该模型以确保它对新数据生成准确的预测。要进行模型评估，请使用您在中创建的测试数据集。第 3 步：下载、浏览和转换数据集 (p. 60).

评估已部署到的模型 SageMaker 托管服务

要评估模型并在生产中使用该模型，请使用测试数据集调用终端节点，并检查您获取的推理是否返回想要实现的目标准化。

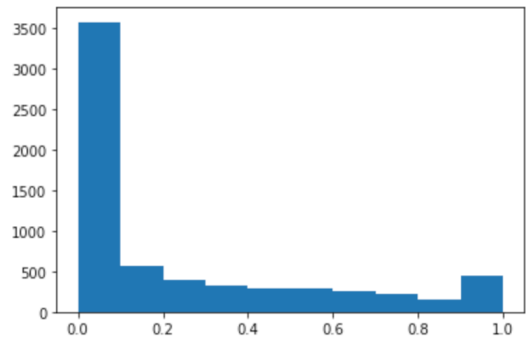
评估模型

1. 设置以下函数来预测测试集中的每一行。在以下示例代码中，rows参数是指定一次要预测的行数。您可以更改它的值，以执行充分利用实例硬件资源的批量推断。

```
import numpy as np
def predict(data, rows=1000):
    split_array = np.array_split(data, int(data.shape[0] / float(rows) + 1))
    predictions = ''
    for array in split_array:
        predictions = ','.join([predictions,
 xgb_predictor.predict(array).decode('utf-8')])
    return np.fromstring(predictions[1:], sep=',')
```

2. 运行以下代码来预测测试数据集并绘制直方图。您只需要测试数据集的要素列，不包括实际值的第 0 列。

```
import matplotlib.pyplot as plt

predictions=predict(test.to_numpy()[:,1:])
plt.hist(predictions)
plt.show()
```



3. 预测值是浮点型。确定True要么False根据浮点值，你需要设置截止值。如以下示例代码所示，使用 Scikit-Learn 库返回输出混淆指标和截止值为 0.5 的分类报告。

```
import sklearn

cutoff=0.5
print(sklearn.metrics.confusion_matrix(test.iloc[:, 0], np.where(predictions > cutoff,
 1, 0)))
print(sklearn.metrics.classification_report(test.iloc[:, 0], np.where(predictions >
 cutoff, 1, 0)))
```

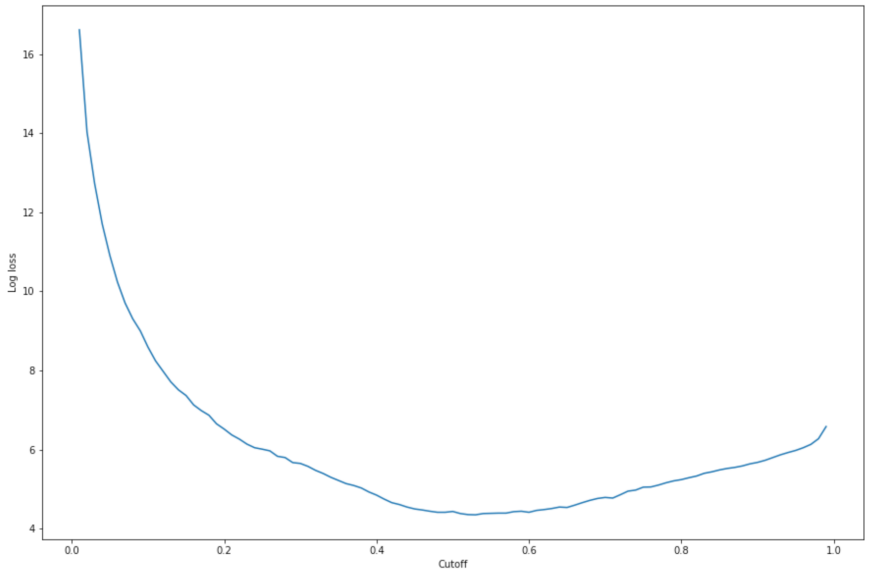
这应返回以下混淆矩阵：

```
[[4670  356]
 [ 480 1007]]
              precision    recall  f1-score   support

           0       0.91      0.93      0.92      5026
           1       0.74      0.68      0.71      1487

    accuracy                           0.87      6513
   macro avg       0.82      0.80      0.81      6513
weighted avg       0.87      0.87      0.87      6513
```

4. 要找到与给定测试集合的最佳截止值，请计算 Logistic 回归的对数损失函数。对数丢失函数被定义为逻辑模型的负对数可能性，该模型返回其地面真相标签的预测概率。下面的示例代码以数字方式和迭代方式计算对数损失值 (`-(y*log(p)+(1-y)log(1-p))`，其中`y`是真正的标签`p`是对应测试样本的概率估计。它返回了日志丢失与截止图。

```
import matplotlib.pyplot as plt

cutoffs = np.arange(0.01, 1, 0.01)
log_loss = []
for c in cutoffs:
    log_loss.append(
        sklearn.metrics.log_loss(test.iloc[:, 0], np.where(predictions > c, 1, 0))
    )

plt.figure(figsize=(15,10))
plt.plot(cutoffs, log_loss)
plt.xlabel("Cutoff")
plt.ylabel("Log loss")
plt.show()
```

这应返回以下日志损失曲线。



5. 使用 NumPy 查找错误曲线的最小点`argmin`和`min`函数：

```
print(
    'Log loss is minimized at a cutoff of ', cutoffs[np.argmin(log_loss)],
    ', and the log loss value at the minimum is ', np.min(log_loss)
```

```
)
```

应返回：`Log loss is minimized at a cutoff of 0.53, and the log loss value at the minimum is 4.348539186773897.`

您可以估计成本函数作为替代方案，而不是计算和最大限度地减少日志丢失函数。例如，如果您想训练模型来对业务问题（例如客户流失预测问题）执行二进制分类，则可以为混淆矩阵的元素设置权重并相应地计算成本函数。

现在，您在 SageMaker 中训练、部署并评估了第一个模型。

Tip

要监控模型质量、数据质量和偏差偏差，请使用亚马逊 SageMaker 模型监视器和 SageMaker 澄清。要了解更多信息，请参阅亚马逊 SageMaker 模型监控器、监控数据质量、监控模型质量、监控偏差漂移, 和监控功能归因漂移.

Tip

要获取对低置信度 ML 预测的人类审核或随机预测样本，请使用 Amazon Advanced Augmented AI 人工审核工作流程。有关更多信息，请参阅 。将 Amazon Augmented AI 用于人类评论.

步骤 7：清除

为避免产生不必要的费用，请使用Amazon Web Services Management Console删除运行练习时创建的终端节点和资源。

Note

不能删除训练作业和日志，并且会无限期保留。

Note

如果您打算探索本指南中的其他练习，您可能需要保留这些资源的一些资源，例如笔记本实例、S3 存储桶和 IAM 角色。

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/并删除以下资源：
   - 终端节点。如果删除终端节点，则还会删除支持它的一个或多个 ML 计算实例。
     1. 在推理下面，选择终端节点。
     2. 选择您在该示例中创建的终端节点，请选择操作，然后选择Delete.
   - 终端节点配置。
     1. 在推理下面，选择终端节点配置。
     2. 选择您在该示例中创建的终端节点配置，选择操作，然后选择Delete.
   - 模型。
     1. 在推理下面，选择模型。
     2. 选择您在该示例中创建的模型，选择操作，然后选择Delete.
   - 笔记本实例。在删除笔记本实例之前，请停止该实例。
     1. 在笔记本下面，选择笔记本实例。
     2. 选择在示例中创建的笔记本实例，选择操作，然后选择停止. 笔记本实例需要几分钟的时间才能停止。当状态对 的更改Stopped，请继续执行下一步。
     3. 选择 Actions，然后选择 Delete。

2. 在以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/，然后删除为存储模型构件和训练数据集创建的存储桶。
3. 打开 Amazon CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/，然后删除所有名称以开头的日志组。`/aws/sagemaker/`.

亚马逊 SageMaker Machine Learning 环境

亚马逊 SageMaker 支持以下机器学习环境。

- 亚马逊 SageMaker 工作室：您可以构建、训练、调试、部署和监控机器学习模型。
- Amazon SageMaker 上的 rStudio：rStudio 是用于的 IDER，具有控制台、支持直接代码执行的语法突出显示编辑器以及用于绘制、历史记录、调试和工作空间管理的工具。

要使用这些机器学习环境，您必须创建亚马逊 SageMaker 域。

主题

- Amazon SageMaker 域 (p. 75)
- 亚马逊 SageMaker 工作室 (p. 77)
- Amazon SageMaker 上的 rStudio (p. 131)

Amazon SageMaker 域

Amazon SageMaker 域支持 SageMaker 机器学习环境。SageMaker 域创建以下实体。有关创建域的入门步骤的信息，请参阅。注册到 Amazon SageMaker 域 (p. 32).

- 域：Amazon SageMaker 域由关联的 Amazon Elastic File System (Amazon EFS) 卷、授权用户列表以及各种安全性、应用程序、策略和 Amazon Virtual Private Cloud (Amazon VPC) 配置组成。网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon每个区域限制为一个域。域中的用户可以彼此共享笔记本文件和其他构件。
- UserProfile：用户配置文件代表域中的单个用户，也是为了共享、报告和其他面向用户的功能而引用用用户的主要方式。此实体是在用户登录 Amazon SageMaker 域时创建的。
- 应用：应用程序代表支持用户的笔记本、终端和控制台的读取和执行体验的应用程序。应用程序的类型可以是 jupyterServer、KernelGateway、rStudioServerPro 或 rSession。用户可能同时激活多个应用程序。

下表描述了域、UserProfile 和应用程序实体的状态值。如果适用，他们还提供了故障排除步骤。

域状态值

值	描述
Pending (待处理)	正在创建域。
InService	成功创建域。
正在更新	持续更新域名。
Deleting (正在删除)	持续删除域名。
失败	域的创建失败。调用`DescribeDomain`查看创建域的失败原因的 API。修复中提到的错误后，删除失败的域并重新创建域`FailureReason`.

值	描述
更新 _ 失败	域更新失败。调用`DescribeDomain`查看域更新失败原因的 API。调用`UpdateDomain`修复中提到的错误之后的 API`FailureReason`.
DELETE_FAILED	删除域名失败。调用`DescribeDomain`查看删除域名的失败原因的 API。由于删除失败，您可能有一些资源仍在运行，但是您不使用或更新域。调用`DeleteDomain`在修复中提到的错误后再次 API`FailureReason`.

UserProfile 状态值

值	描述
Pending (待处理)	正在创建用 UserProfile。
InService	成功创建用 UserProfile。
正在更新	UserProfile 的持续更新。
Deleting (正在删除)	正在删除用 UserProfile。
失败	UserProfile 的创建失败。UserProfile 的创建失败。调用`DescribeUserProfile`查看用 UserProfile 创建失败原因的 API。删除失败的 UserProfile 并在修复中提到的错误后重新创建`FailureReason`.
更新 _ 失败	UserProfile 更新失败。调用`DescribeUserProfile`查看 UserProfile 更新失败原因的 API。调用`UpdateUserProfile`在修复中提到的错误后再次 API`FailureReason`.
DELETE_FAILED	删除 UserProfile 失败。调用`DescribeUserProfile`查看删除 UserProfile 的失败原因的 API。由于删除失败，您可能有一些资源仍在运行，但是您不使用或更新 UserProfile。调用`DeleteUserProfile`在修复中提到的错误后再次 API`FailureReason`.

应用状态值

值	描述
Pending (待处理)	正在创建应用程序。
InService	成功创建应用。
Deleting (正在删除)	持续删除应用程序。
失败	应用程序的创建失败。调用`DescribeApp`查看应用创建失败原因的 API。调用`CreateApp`在修复中提到的错误后再次 API`FailureReason`.
Deleted (已删除)	成功删除应用程序。

亚马逊 SageMaker 工作室

亚马逊 SageMaker Studio 是一个用于机器学习的基于 Web 的集成开发环境 (IDE)，可让您构建、训练、调试、部署和监控机器学习模型。 SageMaker Studio 提供了您在提高生产效率的同时模型从实验到生产整个过程所需的所有工具。在单个统一的可视化界面中，客户可以执行以下任务：

- 在 Jupyter 笔记本中编写和执行代码
- 构建和训练机器学习模型
- 部署模型并监控其预测性能
- 跟踪和调试机器学习实验

有关登录的入门步骤的信息 SageMaker 工作室，请参阅注册到 Amazon SageMaker 域 (p. 32).

对于Amazon支持的区域 SageMaker 工作室，请参阅支持的区域和配额 (p. 28).

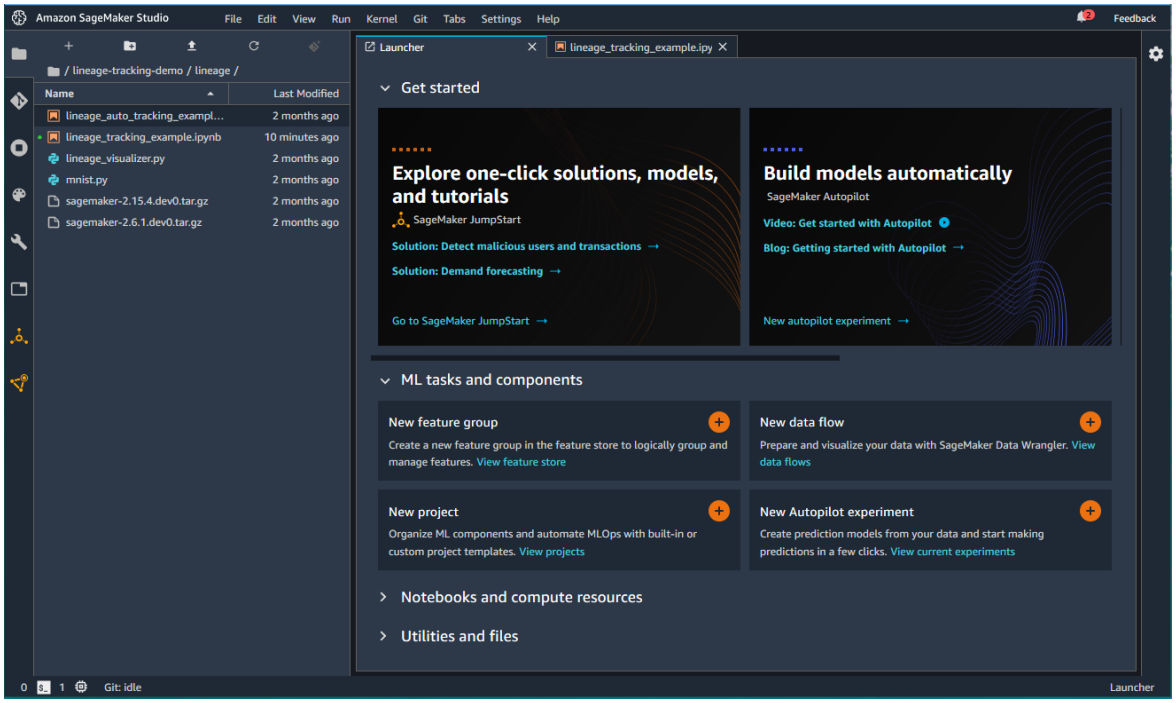
主题

- 亚马逊 SageMaker Studio UI 概述 (p. 77)
- 使用 Amazon SageMaker Studio 启动器 (p. 80)
- 使用 Amazon SageMaker Studio Notebooks (p. 82)
- 定制亚马逊 SageMaker 工作室 (p. 101)
- 在亚马逊执行常见任务 SageMaker 工作室 (p. 125)
- 亚马逊 SageMaker Studio 定价 (p. 130)
- Amazon 故障排除 SageMaker 工作室 (p. 130)

亚马逊 SageMaker Studio UI 概述

亚马逊 SageMaker Studio 扩展 JupyterLab 接口。以前的用户 JupyterLab 将注意到用户界面（包括工作区）的相似性。Studio 在界面中添加了许多附加功能。以下部分详细介绍了最突出的附加功能。有关基本的概述 JupyterLab 界面，请参阅这些区域有： JupyterLab 接口.

下图显示了 SageMaker Studio 在打开了文件浏览器并显示 Studio Launcher 的情况下。



该屏幕的顶部是菜单栏。该屏幕的左侧是左侧边栏，其中包含用来打开不同文件浏览器、资源浏览器和工具的图标。该屏幕的右侧是右侧边栏（由 Settings (设置) 图标 ( ) 表示），其中显示了当打开时的上下文属性设置。该屏幕的底部是状态栏。

在设置图标，有一个按钮可以提供有关您使用体验的反馈的按钮。 SageMaker Studio。

在左边反馈按钮里有通知图标。选择图标以查看来自 Studio 的通知，例如新的 Studio 版本和新版本 SageMaker 功能。要更新到 Studio 的新版本，请参阅关闭并更新 SageMaker 工作室和工作室应用 (p. 128).

主工作区在水平方向上分为两个窗格。左侧窗格是文件和资源浏览器。右侧窗格包含资源（例如笔记本、终端、指标和图表）的一个或多个选项卡。

主题

- 左侧边栏 (p. 78)
- 文件和资源浏览器 (p. 79)
- 主要工作区域 (p. 79)
- 设置 (p. 79)

左侧边栏

左侧边栏包含以下图标。在将鼠标指针悬停在图标上方时，工具提示将显示图标名称。在选择一个图标时，文件和资源浏览器将显示所描述的功能。对于分层条目，浏览器顶部的可选位置提示将显示您在层次结构中的位置。

图标	描述
	文件浏览器 选择上传文件图标 ( ) 以将文件添加到 Studio 中。

图标	描述
	双击一个文件以在新选项卡中打开该文件。 要打开相邻文件，请选择包含笔记本、Python 或文本文件的选项卡，然后选择新建文件视图。 在文件浏览器顶部的菜单上选择加号 (+) 以打开 Studio 启动程序。
	正在运行的终端和内核 有关更多信息，请参阅 关闭资源 (p. 96)。
	饭桶 您可以连接到一个 Git 存储库，然后访问各种 Git 工具和操作。有关更多信息，请参阅 在中克隆 Git 存储库 SageMaker 工作室 (p. 125)。
	命令 (Ctrl + Shift + C) 此处提供了大多数菜单命令。
	笔记本工具 您可以通过 Advanced Tools (高级工具) 部分访问笔记本的元数据。此图标仅在笔记本打开时才显示。
	打开的选项卡 提供打开的选项卡列表，如果您具有多个打开的选项卡，该列表是非常有用的。
	SageMaker 快速启动 提供使用创建的解决方案、模型端点或培训作业的列表 SageMaker 快速启动。
	SageMaker 组件和注册表 提供项目、数据讨论者流、管道、实验、试验、模型或端点或访问 feature store 的列表。

文件和资源浏览器

文件和资源浏览器显示了笔记本、实验、试验、试验组件和终端节点的列表。在文件浏览器的顶部菜单上，选择加号 (+) 以打开 Studio 启动程序。您可以使用启动程序创建笔记本、启动 Python 交互式 Shell 或打开终端。

主要工作区域

主要工作区域由多个选项卡组成，其中包含打开的笔记本和终端，以及有关您的实验和终端节点的详细信息。一个常用的选项卡是试验组件列出. 此列表被称为排行榜因为这是你可以比较实验和试验的地方。有关更多信息，请参阅 查看和比较亚马逊 SageMaker 查看实验、试验和试验组件 (p. 1763)。

设置

可使用设置窗格调整表格和图表属性。默认情况下，此窗格隐藏在屏幕的最右侧。要打开此窗格，请选择屏幕右上角的 Settings (设置) 图标 ( )。

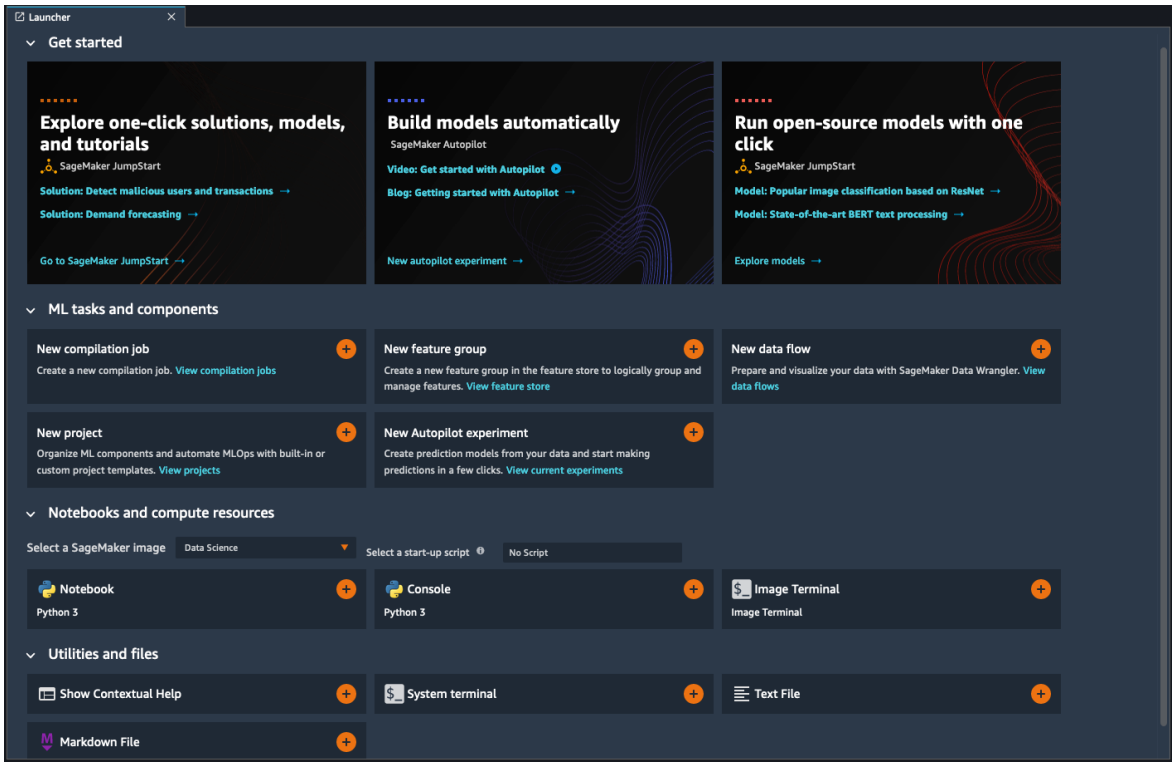
使用 Amazon SageMaker Studio 启动器

您可以使用亚马逊 SageMaker Studio Launcher 用于创建笔记本和文本文件，并启动终端和交互式 Python 外壳。

您可以通过任何以下方法打开 Studio Launcher：

- 选择亚马逊 SageMaker 工作室在工作室的左上角。
- 使用键盘快捷键 `Ctrl + Shift + L`。
- 在 Studio 菜单中，选择File (文件)选择，然后选择新的启动器.
- 如果 Studio 文件浏览器已打开，请选择加号 (+) 登录 Studio 文件浏览器菜单。

启动程序将在 Studio 中的新选项卡中打开。您的屏幕应类似于以下内容：



启动程序包含以下部分：

- 试用— 提供开始使用的材料 SageMaker Studio，例如视频和教程，以及针对机器学习问题的一键式解决方案。
- ML 任务和组件— 创建机器学习任务和组件，例如新功能组、数据流和项目。
- 笔记本和计算资源— 创建笔记本、打开图像终端或打开 Python 控制台。
- 实用程序和文件— 显示笔记本中的上下文帮助、创建文件或打开系统终端。

主题

- 笔记本和计算资源 (p. 81)
- 实用程序和文件 (p. 81)

笔记本和计算资源

要创建或启动项目，请选择 SageMaker 您希望商品在其中运行的图片SageMaker 映像下拉菜单。您还可以选择要运行的生命周期配置脚本。有关更多信息，请参阅 将生命周期配置与亚马逊 SageMaker 工作室 (p. 116)。接下来，选择项目。当您从本部分中选择物品时，可能会产生额外的使用费。有关更多信息，请参阅 使用计量 (p. 98)。

提供以下项目：

- 笔记本

  在所选的内核会话中启动笔记本 SageMaker 映像。有关更多信息，请参阅 更改内核 (p. 95)。

  在文件浏览器中当前选择的文件夹中创建笔记本。要查看文件浏览器，请在 Studio 的左侧边栏中，选择文件浏览器图标 ( )。
- 控制台

  在所选的内核会话中启动 shell SageMaker 映像。

  在文件浏览器中当前选择的文件夹中打开 shell。
- 映像终端

  在所选的终端会话中启动终端 SageMaker 映像。

  在根文件夹中为用户打开终端（如文件浏览器中的 Home 文件夹所示）。

  Note

  CPU 实例在`ml.t3.medium`实例，而 GPU 实例在`ml.g4dn.xlarge`实例。

实用程序和文件

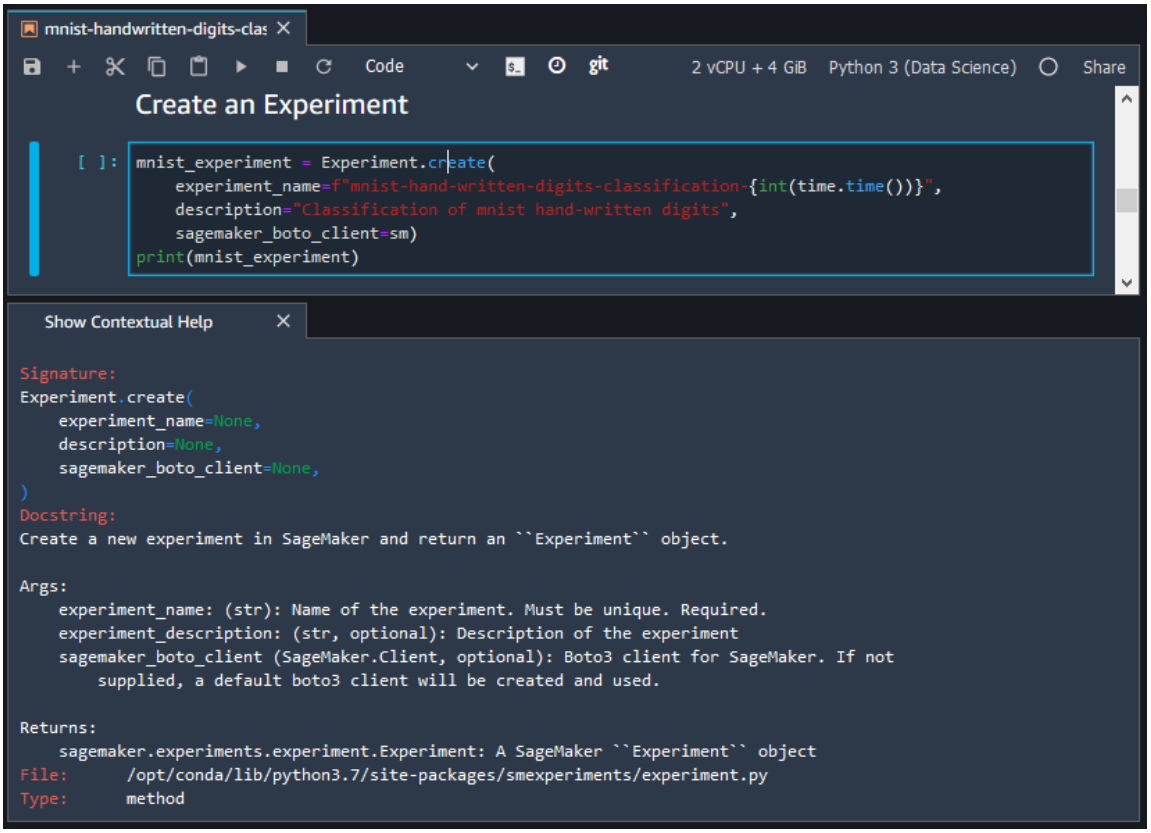
本节中的项目在下文中运行 SageMaker Studio 并且不会产生使用费。

提供以下项目：

- 显示上下文帮助

  打开一个新的选项卡，其中显示 Studio 笔记本中功能的上下文帮助。要显示帮助，请在活动笔记本中选择功能。为了更容易在上下文中查看帮助，请拖动帮助选项卡，使其与笔记本选项卡相邻。要从笔记本中打开帮助选项卡，请按`Ctrl + I`.

  以下屏幕截图显示的上下文帮助：`Experiment.create`方法。



- 系统终端

  在根文件夹中为用户打开 bash shell（如文件浏览器中的 Home 文件夹所示）。
- 文本文件和降价文件

  在您当前在文件浏览器中选择的文件夹中创建关联类型的文件。要查看文件浏览器，请在左侧边栏中，选择文件浏览器图标 ( )。

使用 Amazon SageMaker Studio Notebooks

亚马逊 SageMaker Studio 笔记本是可以快速启动的协作笔记本，因为您无需预先设置计算实例和文件存储。设计在两分钟内启动的一组实例类型（称为快速启动 类型）。 SageMaker Studio 笔记本提供持久存储，即使在运行笔记本的实例关闭时，您也可以查看和共享笔记本。

您可以与其他人共享您的笔记本，以便他们在构建模型和浏览数据时轻松重现您的结果并进行协作。您可以通过安全 URL 提供对笔记本只读副本的访问权限。笔记本的依赖项包含在笔记本的元数据中。在您的同事复制笔记本时，它将在与原始笔记本相同的环境中打开。

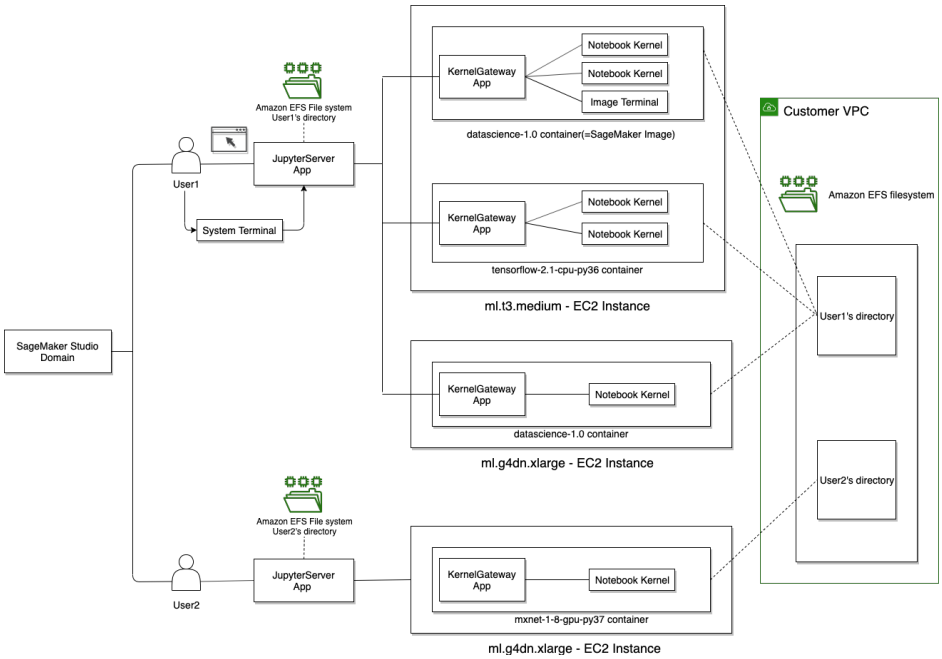
一个 SageMaker Studio 笔记本在以下内容定义的环境中运行：

- EC2 实例类型 — 运行笔记本的硬件配置。该配置包括处理器数量和类型（vCPU 和 GPU）以及内存数量和类型。实例类型决定了定价费率。
- SageMaker 映像 — 与兼容的容器映像 SageMaker Studio。映像包含在 Studio 中运行笔记本所需的内核、语言包和其他文件。一个实例中可以有多个图像。有关更多信息，请参阅 自带 SageMaker 图像 (p. 101)。
- KernelGateway 应用程序 — A SageMaker 图像作为 KernelGateway 应用程序。该应用程序提供对映像中的内核的访问。有一个 one-to-one SageMaker 图像和 SageMaker 应用程序。

- 内核 — 检查和运行笔记本中包含的代码的过程。将内核定义为内核规范在映像中。一张图像中可以有多个内核。

您可以在笔记本中更改任何这些资源。

下图概述了笔记本内核与 KernelGateway 应用程序、用户和相关的运行方式 SageMaker Studio 域。



示例 SageMaker 工作室笔记本电脑可在aws_sagemaker_studio的文件夹亚马逊 SageMaker 示例 GitHub 知识库. 每个笔记本都附带必要的 SageMaker 使用适当的内核打开笔记本的映像。

我们建议您熟悉 SageMaker 在创建或使用 Studio 笔记本之前，请先使用 Studio 界面和 Studio 笔记本工具栏。有关更多信息，请参阅 亚马逊 SageMaker Studio UI 概述 (p. 77) 和 使用 SageMaker Studio 笔记本工具栏 (p. 87)。

主题

- 亚马逊怎么样 SageMaker Studio 笔记本与笔记本实例不同？ (p. 84)
- 入门 (p. 84)
- 亚马逊 SageMaker Studio 导览 (p. 85)
- 创建或打开亚马逊 SageMaker Studio Notebook (p. 86)
- 使用 SageMaker Studio 笔记本工具栏 (p. 87)
- 在亚马逊安装外部库和内核 SageMaker 工作室 (p. 89)
- 分享和使用亚马逊 SageMaker Studio Notebook (p. 90)
- 获取笔记本和应用程序元数据 (p. 91)
- 获取笔记本差异 (p. 93)
- 管理资源 (p. 93)
- 使用计量 (p. 98)
- 可用的资源 (p. 98)

亚马逊怎么样 SageMaker Studio 笔记本与笔记本实例不同？

启动新笔记本时，我们建议您在 Amazon SageMaker Studio 中创建笔记本，而不是从 Amazon 启动笔记本实例。 SageMaker 控制台。使用具有诸多优势 SageMaker Studio 笔记本，包括以下内容：

- 启动 Studio 笔记本比启动基于实例的笔记本更快。通常情况下，它比基于实例的笔记本快 5-10 倍。
- 笔记本共享是中的一项集成功能 SageMaker Studio。用户可以生成一个可共享的链接，该链接可以重现笔记本代码，还可以 SageMaker 用户只需几次单击就可以执行映像。
- SageMaker Studio 笔记本预先安装了最新的亚马逊 SageMaker Python 开发工具包.
- SageMaker Studio 笔记本可在 Studio 中访问。这使您无需离开 Studio 即可构建、训练、调试、跟踪和监控模型。
- Studio 团队的每个成员具有自己的主目录以存储其笔记本和其他文件。目录在启动时自动挂载到所有实例和内核上，因此其笔记本和其他文件始终可用。主目录存储在 Amazon Elastic File System (Amazon EFS) 中，以便您从其他服务访问它们。
- 使用时IAM Identity Center，您可以通过唯一的 URL 使用 SSO 凭证直接访问 SageMaker Studio。您无需与 Amazon Web Services Management Console 交互即可运行笔记本。
- Studio 笔记本配备了一组预定义的 SageMaker 以使您更快地入门的映像设置。

> **Note**
>
> Studio 笔记本不支持本地模式。但是，您可以使用笔记本实例在本地训练数据集的样本，然后在 Studio 笔记本中使用相同的代码对完整数据集进行训练。

在打开笔记本时 SageMaker Studio，视图是 JupyterLab 界面的扩展。主要功能相同，因此您可以找到 Jupyter 笔记本和 JupyterLab 的典型功能。有关 Studio 接口的更多信息，请参阅 亚马逊 SageMaker Studio UI 概述 (p. 77)。

入门

要开始使用，您或您组织的管理员需要完成 Amazon SageMaker Studio 信息载入过程。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。

您可以访问 SageMaker Studio 笔记本采用以下任何方式：

- 您会收到通过组织访问 Studio 的电子邮件邀请IAM Identity Center账户，其中包含登录 Studio 的直接链接，而无需使用 Amazon SageMaker 控制台。您可以继续执行the section called “后续步骤” (p. 85)。
- 您会收到一个指向共享 Studio 笔记本的链接，其中包含登录 Studio 的直接链接，而无需使用 SageMaker 控制台。您可以继续执行the section called “后续步骤” (p. 85)。
- 登录到 Studio，然后登录到 SageMaker 控制台。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。

从亚马逊登录 SageMaker 控制台

从登录 SageMaker 控制台

1. 注册到亚马逊 SageMaker 按照上的说明操作注册到 Amazon SageMaker 域 (p. 32). 如果您已注册，请跳到下一步。
2. 打开 SageMaker 控制台.
3. 选择控制面板.
4. 这些区域有：控制面板打开。
5. 在控制面板，您将看到用户名的列表。

   在您的用户名旁边，选择 Open Studio (打开 Studio)。

后续步骤

现在，您位于 Studio 中，您可以尝试任何以下方法：

- 创建 SageMaker Studio 笔记本 — 继续执行下一部分。
- 熟悉 SageMaker 工作室界面 — 请参阅亚马逊 SageMaker Studio UI 概述 (p. 77).
- 探索 Studio end-to-end 教程笔记本 — 请参阅亚马逊 SageMaker Studio 导览 (p. 85).

亚马逊 SageMaker Studio 导览

有关带您了解亚马逊主要功能的演练 SageMaker 工作室，请参阅xgboost_customer_churn_studio.ipynb来自aws/amazon-sagemaker-examples存储库。笔记本中的代码训练多个模型并设置 SageMaker 调试程序和 SageMaker 模型监控器。本演练向您展示如何查看试验、比较生成的模型、显示调试程序结果以及使用 SageMaker UI。您无需了解代码即可遵循本演练。

先决条件

要运行本次导览的笔记本，您需要：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的IAM Identity Center或 IAM 账户以登录到 Studio。有关信息，请参阅注册到 Amazon SageMaker 域 (p. 32)。
- 基本熟悉 Studio 用户界面和 Jupyter 笔记本。有关信息，请参阅亚马逊 SageMaker Studio UI 概述 (p. 77)。
- 的副本aws/amazon-sagemaker-examplesStudio 环境中的存储库。

克隆存储库

1. 登录到 SageMaker Studio。对于 IAM Identity Center 用户，请使用邀请电子邮件中的 URL 登录。对于 IAM 用户，请按照以下步骤操作。
   a. 登录到SageMaker 控制台.
   b. 选择控制面板在左侧导航窗格中。
   c. 选择启动应用在用户名旁边的行中。
   d. 选择工作室从下拉菜单中。
2. 在顶部菜单上，选择文件、新建 ，然后选择终端。
3. 在命令提示符处，运行以下命令以下命令克隆aws/amazon-sagemaker-examples存储库。

```
git clone https://github.com/aws/amazon-sagemaker-examples.git
```

导航到样本笔记本

1. 从文件浏览器在左侧菜单中，选择亚马逊 sagemaker 示例.
2. 使用以下路径导航到示例笔记本。

   ```
   ~/amazon-sagemaker-examples/aws_sagemaker_studio/getting_started/
   xgboost_customer_churn_studio.ipynb
   ```

   Note

   如果运行示例笔记本时遇到错误，并且克隆存储库的时间已经过去了一段时间，请查看远程存储库上的笔记本以获取更新。

创建或打开亚马逊 SageMaker Studio Notebook

在亚马逊创建笔记本电脑时 SageMaker Studio 或首次在 Studio 中打开非共享笔记本，你必须选择 SageMaker 笔记本的图像和内核。 SageMaker 根据选定的类型，在默认类型的实例上启动笔记本。 SageMaker 映像。对于基于 CPU 的映像，默认实例类型为`ml.t3.medium`（ 作为的一部分提供Amazon免费套餐)。对于基于 GPU 的映像，默认实例类型为 `ml.g4dn.xlarge`。

如果您创建或打开使用相同实例类型的其他笔记本，则无论笔记本是否使用相同内核，笔记本将在该实例类型的同一实例上运行。

在启动笔记本后，您可以更改其实例类型， SageMaker 从笔记本中打开映像和内核。有关更多信息，请参阅 更改实例类型 (p. 94) 和 更改内核 (p. 95)。

每种实例类型只能有一个实例。每个实例可以在其上运行多个 SageMaker 映像。每个 SageMaker 映像可以运行多个内核或终端实例。

将按实例进行计费，并在启动给定实例类型的第一个实例时开始计费。如果您想创建或打开笔记本电脑而不会产生费用的风险，请从File (文件)菜单然后选择无内核来自 的选择内核对话框。您可以在没有正在运行的内核的情况下读取和编辑笔记本，但无法运行单元格。

账单在 SageMaker 关闭了实例的映像。有关更多信息，请参阅 使用计量 (p. 98)。

有关关闭笔记本的信息，请参阅关闭资源 (p. 96).

主题

- 打开 Studio 笔记本 (p. 86)
- 从“File (文件)”菜单创建笔记本 (p. 86)
- 从启动程序创建笔记本 (p. 86)

打开 Studio 笔记本

SageMaker Studio 只能打开 Studio 文件浏览器中列出的笔记本。有关将笔记本上传到文件浏览器的说明，请参阅将文件上传到 SageMaker 工作室 (p. 125)要么在中克隆 Git 存储库 SageMaker 工作室 (p. 125).

打开笔记本

1. 在左侧边栏中，选择 File Browser (文件浏览器) 图标 ( ) 以显示文件浏览器。
2. 浏览找到笔记本文件并双击该文件，以在新选项卡中打开笔记本。

从“File (文件)”菜单创建笔记本

从“File (文件)”菜单创建笔记本

1. 在 Studio 菜单中，选择File (文件)，选择New，然后选择笔记本.
2. 在 Select Kernel (选择内核) 对话框中，要使用默认内核 Python 3 (Data Science) (Python 3 (数据科学))，请选择 Select (选择)。否则，请使用下拉菜单选择其他内核。

有关可用内核的列表，请参阅Amazon 可用 SageMaker 内核 (p. 100)。

从启动程序创建笔记本

从启动程序创建笔记本

1. 要打开启动器，请使用键盘快捷键`Ctrl + Shift + L`.

或者，从文件浏览器中选择加号 (+) 在菜单左侧签名。

2. 在启动器上，保留默认值 SageMaker 映像，数据科学，或使用下拉菜单选择其他映像。
3. 在 Notebook (笔记本) 下，选择 Python3。

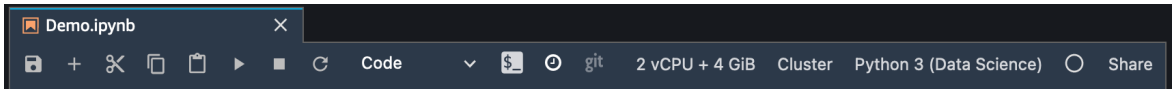
有关可用映像的列表，请参阅Amazon 可用 SageMaker 映像 (p. 100)。

选择内核或映像后，笔记本将在新 Studio 选项卡中启动并打开笔记本。要查看笔记本的内核会话，请在左侧边栏中选择 Running Terminals, Kernels, and Images (正在运行的终端、内核和映像) 图标 ( )。您可以从此视图停止笔记本的内核会话。

使用 SageMaker Studio 笔记本工具栏

亚马逊 SageMaker 将 Studio 笔记本扩展 JupyterLab 接口。有关基本 JupyterLab 界面的概述，请参阅 JupyterLab 界面。

下图显示了工具栏和一个空单元格。 SageMaker Studio 笔记本。



在暂停在工具栏图标上方时，工具提示将显示图标功能。其他笔记本命令位于 Studio 主菜单中。有关可用笔记本命令和快捷方式的列表，请在 Studio 的左侧栏中选择命令图标 ( )，然后滚动到手提电脑单元格和笔记本操作部分。该工具栏包含以下图标：

图标	描述
	保存和检查点 保存笔记本并更新检查点文件。有关更多信息，请参阅 获取最后一个检查点之间的差异 (p. 93)。
	插入单元格 在当前单元格下方插入一个代码单元格。当前单元格由左边距中的蓝色垂直标记表示。
	剪切、复制和粘贴单元格 剪切、复制和粘贴选定的单元格。
	运行单元格 运行选定单元格，然后使位于最后一个选定单元格后面的单元格成为新的选定单元格。
	中断内核 中断内核，这将取消当前正在运行的操作。内核保持活动状态。
	重新启动内核 重新启动内核。变量被重置。不保存的信息不受影响。

图标	描述
	单元格类型 显示或更改当前单元格类型。单元格类型为： • 代码 — 内核运行的代码。 • Markdown — 呈现为降价的文本。 • Raw — 显示为文本的内容，包括 Markdown 标记。
	启动终端 在中启动终端 SageMaker 托管笔记本的映像。有关示例，请参阅获取应用程序元数据 (p. 92)：
	检查点差异 打开一个新的选项卡，其中显示笔记本和检查点文件之间的差异。有关更多信息，请参阅 获取最后一个检查点之间的差异 (p. 93)。
	Git 差异 仅当从 Git 存储库打开笔记本时才启用。打开一个新的选项卡，其中显示笔记本和上次 Git 提交之间的差异。有关更多信息，请参阅 获取最后一次提交之间的差异 (p. 93)。
2 vCPU + 4 GiB	实例类型 显示或更改笔记本运行的实例类型。格式如下所示： `number of vCPUs + amount of memory + number of GPUs` `Unknown` 表示笔记本已打开而未指定内核。笔记本运行在 SageMaker Studio 实例不会产生运行时费用。您无法将笔记本分配给实例类型。您必须指定内核，然后 Studio 将笔记本分配给默认类型。 有关更多信息，请参阅 创建或打开亚马逊 SageMaker Studio Notebook (p. 86) 和 更改实例类型 (p. 94)。
Python 3（数据科学）	内核和 SageMaker Image 显示或更改用于处理笔记本中的单元格的内核。格式如下所示： `Kernel (SageMaker Image)` `No Kernel` 表示笔记本已打开而未指定内核。您可以编辑笔记本，但无法运行任何单元格。 有关更多信息，请参阅 更改内核 (p. 95)。
	内核忙状态 显示内核的忙状态。当圆的边缘及其内部颜色相同时，内核处于忙碌状态。内核在启动时和处理单元格时处于忙碌状态。其他内核状态将显示在 SageMaker Studio。
	共享笔记本 共享笔记本。有关更多信息，请参阅 分享和使用亚马逊 SageMaker Studio Notebook (p. 90)。

要选择多个单元格，请单击单元格外部的左边距。按住 Shift 键并使用 K 或 Up 键选择前面的单元格，或使用 J 或 Down 键选择后面的单元格。

在亚马逊安装外部库和内核 SageMaker 工作室

亚马逊 SageMaker Studio 笔记本电脑已经安装了多个映像。这些映像包含内核和 Python 程序包，包括 scikit、熊猫、NumPy、TensorFlow 和 MXNet。您还可以安装自己的映像，其中包含您选择的程序包和内核。有关安装映像的更多信息，请参阅自带 SageMaker 图像 (p. 101).

亚马逊中不同的 Jupyter 内核 SageMaker Studio 笔记本是独立的 conda 环境。有关 Conda 环境的更多信息，请参阅。管理环境.

Package 安装工具

从终端安装 Python 软件包的方法因图像而异。Studio 支持以下软件包安装工具：

- 笔记本— 支持以下命令。如果以下其中一项不适用于您的图片，请尝试另一个。
  - `%conda install`
  - `%pip install`
- Jupyter 终端— 你可以直接使用 pip 和 conda 安装软件包。您还可以使用`apt-get install`从终端安装系统包。

  Note

  我们建议不要使用`pip install -u`要么`pip install --user`，因为这些命令将软件包安装在用户的 Amazon EFS 卷上，并且可能会阻止 JupyterServer 应用程序重新启动。相反，使用生命周期配置在应用程序重启时重新安装所需的软件包，如中所示使用生命周期配置安装包 (p. 90).

我们建议使用`%pip`和`%conda`从笔记本中安装软件包，因为它们正确考虑了正在使用的激活环境或解释器。有关更多信息，请参阅 。添加 %pip 和 %conda 魔术函数. 你也可以使用 system 命令语法（以! 开头的行）来安装软件包。例如，`!pip install` 和 `!conda install`。

Conda

Conda 是一个开源程序包管理系统和环境管理系统，可以安装软件包及其依赖项。 SageMaker 支持将 conda 用于这两个主要频道中的任何一个：默认频道或 conda-FORGE 频道。有关更多信息，请参阅 。Conda 频道. conda-forge 频道是一个社区频道，贡献者可以在其中上传包裹。

Note

从 Cona-Forge 安装软件包最多可能需要 10 分钟。时机与 conda 如何解析依赖关系图有关。

所有 SageMaker 前提是环境正常运行。用户安装的包可能无法正常工作。

Conda 有两种激活环境的方法：`conda activate`, 和`source activate`. 有关更多信息，请参阅 。管理环境.

支持的 conda 操作

- `conda install`在单个环境中的软件包
- `conda install`在所有环境中的软件包
- 从主 conda 存储库安装软件包
- 从 conda-Forge 安装软件包
- 更改 conda 安装位置以使用亚马逊 EBS

- 支持两者`conda activate`和`source activate`

Pip

Pip 是安装和管理 Python 软件包的工具。默认情况下，Pip 在 Python 程序 Package 索引 (PyPI) 上搜索程序包。与 conda 不同，pip 没有内置的环境支持。因此，对于具有本机或系统库依赖关系的软件包而言，pip 并不像 conda 那么彻底。Pip 可以用来在 conda 环境中安装软件包。您可以将替代软件包存储库与 pip 一起使用，而不是 PyPI。

支持的点操作

- 使用 pip 在没有活动 conda 环境的情况下安装软件包
- 使用 pip 在 conda 环境中安装软件包
- 使用 pip 在所有 conda 环境中安装软件包
- 更改 pip 安装位置以使用亚马逊 EBS
- 使用替代存储库使用 pip 安装软件包

不支持

SageMaker 旨在支持尽可能多的软件包安装操作。但是，如果软件包是由 SageMaker 而且您对这些软件包使用以下操作，可能会使您的环境不稳定：

- 卸载
- 降级
- 升级

由于网络条件或配置的潜在问题，或者 Conda 或 PyPi 的可用性，软件包可能无法在固定或确定性的时间内安装。

> **Note**
>
> 尝试在具有不兼容依赖关系的环境中安装软件包可能会导致失败。如果出现问题，您可以联系库维护者了解更新软件包依赖关系。当您修改环境（例如删除或更新现有软件包）时，这可能会导致该环境的不稳定性。

使用生命周期配置安装包

在 Studio 实例的 Amazon EBS 卷上安装自定义映像和内核，以便在您停止并重新启动笔记本电脑时它们持续存在，并且 SageMaker 不会更新您安装的任何外部库。要做到这一点，请使用包含创建笔记本时运行的脚本的生命周期配置 (`on-create`)以及每次重新启动笔记本时运行的脚本（`on-start`)。有关在 Studio 中使用生命周期配置的更多信息，请参阅将生命周期配置与亚马逊 SageMaker 工作室 (p. 116). 有关生命周期配置脚本示例，请SageMaker Studio 生命周期配置示例.

分享和使用亚马逊 SageMaker Studio Notebook

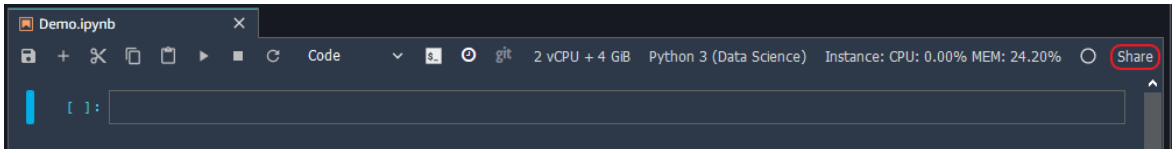
你可以分享你的亚马逊 SageMaker 与同事一起使用的 Studio 笔记本。共享笔记本是副本。共享笔记本后，对原始笔记本所做的任何更改都不会反映在共享笔记本中，而同事在笔记本的共享副本中所做的任何更改都不会反映在原始笔记本中。如果要共享最新版本，则必须创建新快照，然后共享快照。

主题

- 共享笔记本 (p. 91)
- 使用共享笔记本 (p. 91)

共享笔记本

以下屏幕截图显示 Studio 笔记本中的菜单。



共享笔记本

1. 在笔记本的右上角，选择 Share (共享)。
2. （可选）在 Create shareable snapshot (创建可共享快照) 中，选择以下任一项：
   - 包括 Git 回购信息— 包含指向包含笔记本的 Git 存储库的链接。这样，您和您的同事能够开展协作，并为同一 Git 存储库做出贡献。
   - 包含输出— 包含所有已保存的笔记本输出。

     Note

     如果您是 IAM Identity Center 用户并且看不到这些选项，则 IAM Identity Center 管理员可能已禁用该功能。请联系您的管理员。
3. 选择 Create（创建）。
4. 创建快照后，选择 Copy link (复制链接)，然后选择 Close (关闭)。
5. 与您的同事分享链接。

选择共享选项后，系统会为您提供一个 URL。您可以与有权访问 Amazon 的用户共享此链接 SageMaker Studio。当用户打开该 URL 时，系统会提示他们使用登录IAM Identity Center或 IAM 身份验证。该共享笔记本将变为副本，因此，接收人所做的更改不会在原始笔记本中重现。

使用共享笔记本

共享笔记本的使用方式与您使用自己创建的笔记本的使用方式相同。在首次单击共享笔记本的链接时，将打开该笔记本的只读版本。要编辑共享笔记本，请选择 Create a Copy (创建副本)。这会将共享笔记本复制到您的个人存储中。

复制的笔记本在实例类型的实例上启动， SageMaker 笔记本在发送人共享笔记本时使用的映像。如果您当前未运行该实例类型的实例，则会启动一个新实例。自定义 SageMaker 映像未共享。您还可以通过选择 Snapshot Details (快照详细信息) 来检查笔记本快照。

以下是有关共享和身份验证的重要注意事项：

- 如果您具有活动会话，您将看到笔记本的只读视图，直到您选择 Create a Copy (创建副本) 为止。
- 如果您没有活动会话，则需要登录。
- 如果您使用 IAM 登录，请在登录后选择用户配置文件，然后选择打开 SageMaker 工作室. 然后您需要选择您收到的链接。
- 如果您使用 SSO 登录，登录后将在 Studio 中自动打开共享笔记本。

获取笔记本和应用程序元数据

您可以使用 Amazon 访问笔记本元数据和应用程序元数据 SageMaker UI。

主题

- 获取笔记本元数据 (p. 92)

- 获取应用程序元数据 (p. 92)

获取笔记本元数据

Jupyter 笔记本包含可通过 Amazon SageMaker UI 访问的可选元数据。

查看笔记本元数据

1. 在左侧边栏中，选择 Notebook Tools (笔记本工具) 图标 ( )。仅当 Studio 中有笔记本电脑可用时，该图标才会显示。
2. 打开 Advanced Tools (高级工具) 部分。

元数据应类似于以下内容。

```
{
    "instance_type": "ml.t3.medium",
    "kernelspec": {
        "display_name": "Python 3 (Data Science)",
        "language": "python",
        "name": "python3__SAGEMAKER_INTERNAL__arn:aws:sagemaker:us-west-2:<acct-id>:image/
datascience-1.0"
    },
    "language_info": {
        "codemirror_mode": {
            "name": "ipython",
            "version": 3
        },
        "file_extension": ".py",
        "mimetype": "text/x-python",
        "name": "python",
        "nbconvert_exporter": "python",
        "pygments_lexer": "ipython3",
        "version": "3.7.6"
    }
}
```

获取应用程序元数据

在亚马逊创建笔记本电脑时 SageMaker Studio 时，应用程序元数据将写入到名为的文件中。`resource-metadata.json`在文件夹中的`/opt/ml/metadata/`. 您可以从笔记本中打开映像终端以获取应用程序元数据。该元数据为您提供以下信息，其中包括 SageMaker 运行笔记本的映像和实例类型：

- AppType–`KernelGateway`
- DomainId— 与 StudioID 相同
- UserProfileName— 当前用户的配置文件名称
- ResourceArn— 应用程序的 Amazon 资源名称 (ARN)，其中包括实例类型
- ResourceName— 名称 SageMaker 图像

Studio 可能会包含其他元数据，供 Studio 内部使用，可能会发生变化。

获取应用程序元数据

1. 在笔记本菜单的中心，选择启动终端 图标 ( )。这会在中打开一个终端。 SageMaker 笔记本运行的映像。

2. 运行以下命令以显示 `resource-metadata.json` 文件的内容。

```
cd /opt/ml/metadata/
cat resource-metadata.json
```

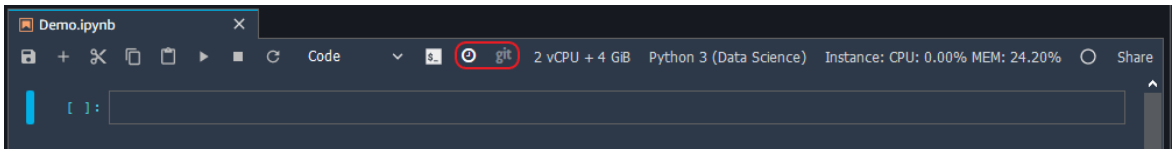
该文件应类似于以下内容。

```
{
    "AppType": "KernelGateway",
    "DomainId": "d-xxxxxxxxxxxx",
    "UserProfileName": "profile-name",
    "ResourceArn": "arn:aws:sagemaker:us-east-2:account-id:app/d-xxxxxxxxxxxx/profile-name/
KernelGateway/datascience--1-0-ml-t3-medium",
    "ResourceName": "datascience--1-0-ml"
}
```

获取笔记本差异

您可以使用 Amazon 显示当前笔记本与最后一个检查点或最后一个 Git 提交之间的差异。 SageMaker UI。

以下屏幕截图显示 Studio 笔记本中的菜单。



主题

- 获取最后一个检查点之间的差异 (p. 93)
- 获取最后一次提交之间的差异 (p. 93)

获取最后一个检查点之间的差异

创建笔记本时，将创建与笔记本匹配的隐藏检查点文件。您可以查看笔记本和检查点文件之间的更改，或者还原笔记本以匹配检查点文件。

默认情况下，笔记本每 120 秒自动保存一次，当您关闭笔记本时也会自动保存一次。但是，检查点文件不会更新为与笔记本相匹配。要保存笔记本并更新检查点文件以匹配，必须选择笔记本菜单左侧的保存笔记本并创建检查点图标 ( ) 或使用 `Ctrl + S` 键盘快捷键。

要查看笔记本和检查点文件之间的更改，请选择笔记本菜单中心的检查点差异图标 ( )。

要将笔记本还原为检查点文件，请从 Studio 主菜单中选择文件 ，然后选择将笔记本还原为检查点。

获取最后一次提交之间的差异

如果笔记本是从 Git 存储库打开的，您可以查看笔记本和上次 Git 提交之间的差异。

要查看最后一次 Git 提交的笔记本中的更改，请选择笔记本菜单中心的 Git 差异图标 ( )。

管理资源

您可以更改实例类型， SageMaker 映像和内核来自 Amazon SageMaker Studio 笔记本中。要创建自定义内核以用于笔记本，请参阅自带 SageMaker 图像 (p. 101).

主题

- 更改实例类型 (p. 94)
- 更改内核 (p. 95)
- 关闭资源 (p. 96)

更改实例类型

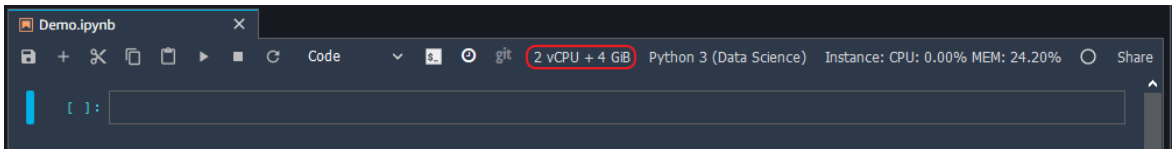
首次打开新 Studio 笔记本时，系统会为您分配默认 Amazon Elastic Compute Cloud (Amazon EC2) 实例类型来运行笔记本。在相同实例类型上打开其他笔记本时，这些笔记本将在与第一个笔记本相同的实例上运行，即使这些笔记本使用不同的内核。

您可以在笔记本中更改运行 Studio 笔记本的实例类型。

以下信息仅适用于 Studio 笔记本电脑。有关如何更改 Amazon 实例类型的信息 SageMaker 笔记本实例，请参阅更新笔记本实例 (p. 152).

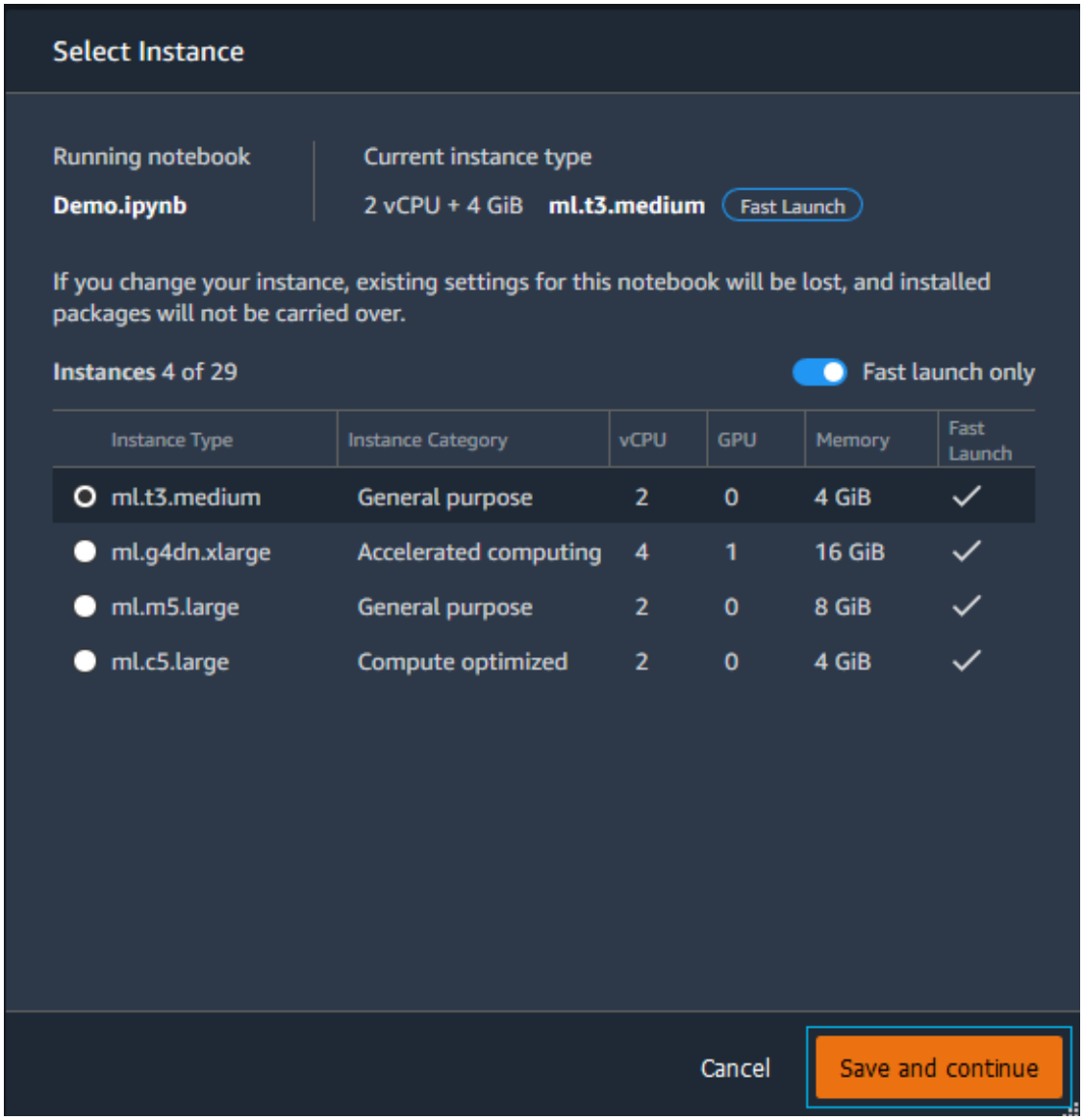
> **Important**
>
> 如果更改实例类型，笔记本的未保存信息和现有设置将会丢失，并且必须重新安装已安装的程序包。
> 即使没有内核会话或应用程序处于活动状态，之前的实例类型仍会继续运行 您必须明确停止实例才能停止累积费用。要停止实例，请参阅关闭资源 (p. 96).

以下屏幕截图显示 Studio 笔记本中的菜单。支持笔记本的实例类型的处理器和内存显示为 2 vCPU + 4 GiB。



更改实例类型

1. 选择实例类型。
2. 在 Select instance (选择实例) 中，选择列出的快速启动实例类型之一。或者，如果要查看所有实例类型，请关闭 Fast launch only (仅快速启动)。该列表可以按任何列排序。



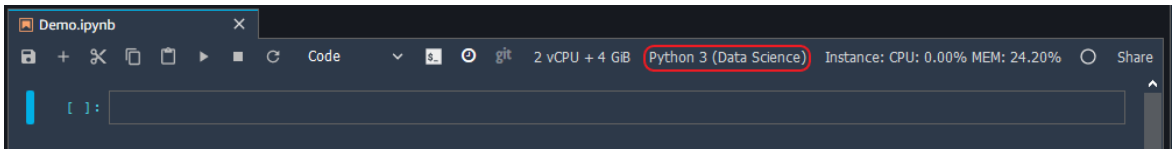
3. 选择类型后，请选择 Save and continue (保存并继续)。
4. 等待新实例启用，然后将显示新实例类型信息。

有关可用的实例类型列表，请参阅Available SageMaker Studio 实例类型 (p. 98)。

更改内核

使用 Amazon SageMaker Studio 笔记本，您可以在笔记本中更改笔记本的内核。

以下屏幕截图显示 Studio 笔记本中的菜单。当前的 SageMaker 内核和图像显示为Python 3（数据科学），其中`Python 3`表示内核和`Data Science`表示 SageMaker 包含内核的图像。右侧圆圈的颜色表示内核处于空闲状态或忙碌状态。当圆的中心和边缘颜色相同时，内核处于忙碌状态。



更改笔记本的内核

1. 选择内核名称。
2. 从下拉列表中，选择一个内核。
3. 选择内核后，选择Select.
4. 等待内核的状态显示为空闲状态，表示内核已启动。

有关可用的列表 SageMaker 内核，请参阅Amazon 可用 SageMaker 内核 (p. 100).

关闭资源

您可以关闭单个资源，包括笔记本、终端、内核、应用程序和实例。您还可以同时关闭其中一个类别中的所有资源。

> Note
>
> 亚马逊 SageMaker Studio 不支持从笔记本中关闭资源。

主题

- 关闭打开的笔记本 (p. 96)
- 关闭资源 (p. 96)

关闭打开的笔记本

您可以从亚马逊关闭打开的笔记本 SageMaker 工作室File (文件)菜单或者来自运行终端和内核窗格。

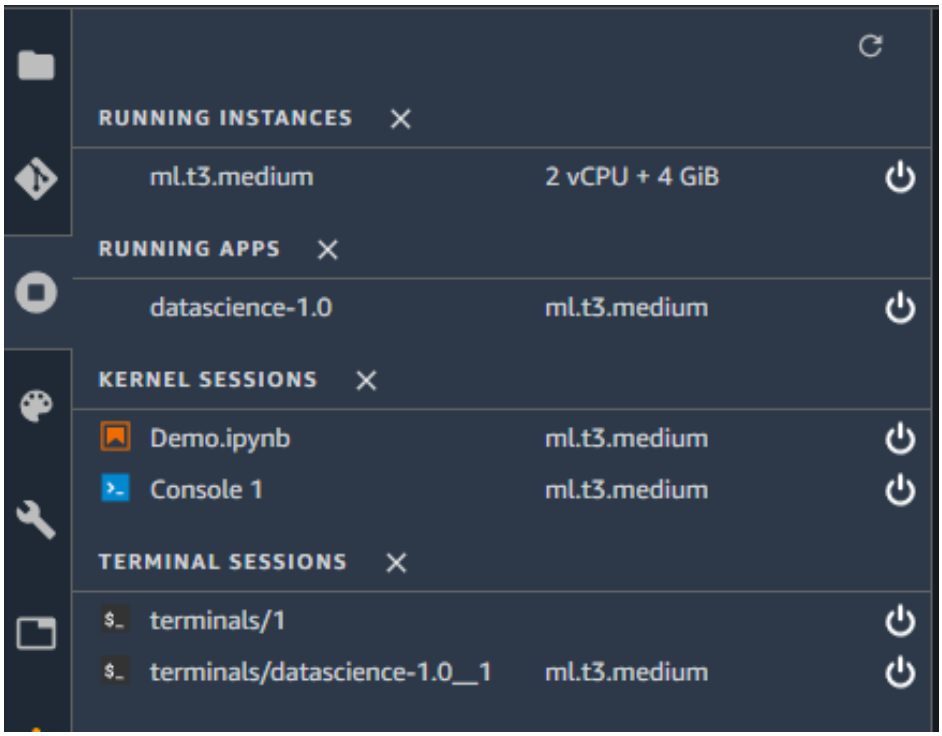
> Note
>
> 当您关闭笔记本时，笔记本中所有未保存的信息都将丢失。此操作不会删除笔记本。

从“File (文件)”菜单关闭打开的笔记本

1. 或者，通过选择Disk图标位于笔记本菜单左侧。
2. 选择 File (文件)，然后选择 Close and Shutdown Notebook (关闭并关上笔记本)。
3. 选择 OK（确定）。

关闭资源

你可以到达正在运行的终端和内核亚马逊左侧的窗格 SageMaker 带有工作室 图标。这些区域有：正在运行的终端和内核窗格由四个部分组成。每个部分列出了该类型的所有资源。您可以单独关闭某个部分中的每个资源，也可以同时关闭所有资源。



在选择关闭某个部分中的所有资源时，将发生以下情况：

- 运行实例/运行应用— 关闭所有实例、应用程序、笔记本电脑、内核会话、控制台/外壳和映像终端。不会关闭系统终端。

  Note

  关闭 Studio 笔记本实例时，任何其他资源，例如 SageMaker 不会删除从 Studio 创建的终端节点、Amazon EMR 集群和 Amazon S3 存储桶。删除这些资源以停止应计费用。
- 内核会话— 所有内核、笔记本电脑和控制台/外壳都已关闭。
- 终端会话— 所有镜像终端和系统终端都已关闭。

关闭资源

1. 在左侧边栏中，选择正在运行的终端和内核图标 ( )。
2. 请执行以下任一操作：

   - 要关闭特定资源，请选择关闭图标 ( ) 与资源相同的行上。

     对于运行的实例，将显示一个确认对话框，将列出将要关闭的所有资源。对于运行的应用程序，将显示确认对话框。选择 Shut down all (全部关闭) 以继续。

     Note

     对于内核会话或终端会话，不显示确认对话框。
   - 要关闭某个部分中的所有资源，请选择X位于章节标签的右侧。此时将显示一个确认对话框。选择 Shut down all (全部关闭) 以继续。

使用计量

使用亚马逊不收取任何额外费用。 SageMaker Studio。运营亚马逊所产生的成本 SageMaker Studio 笔记本、交互式 Shell、控制台和终端基于 Amazon EC2) 实例的使用情况。

在运行以下资源时，您必须选择 SageMaker 映像和内核：

从 Studio 启动程序中

- 笔记本
- 交互式 Shell
- 映像终端

从 File (文件) 菜单

- 笔记本
- 控制台

在启动后，资源将根据选定的实例类型的 Amazon EC2 实例运行。 SageMaker 映像和内核。如果以前启动了该类型的实例并且现在可用，资源将在该实例上运行。

对于基于 CPU 的映像，默认实例类型为 `ml.t3.medium`。对于基于 GPU 的映像，默认实例类型为 `ml.g4dn.xlarge`。

产生的费用基于实例类型。将按每个实例向您单独收取费用。

计量是在创建实例时开始的。在关闭实例上的所有应用程序或实例关闭时，计量将结束计量。有关如何关闭实例的信息，请参阅关闭资源 (p. 96).

> Important
>
> 您必须关闭实例才能停止产生费用。如果您关闭了在实例上运行的笔记本，但没有关闭实例，则仍会产生费用。关闭 Studio 笔记本实例时，任何其他资源，例如 SageMaker 不会删除从 Studio 创建的终端节点、Amazon EMR 集群和 Amazon S3 存储桶。删除这些资源以停止应计费用。

当您在同一实例类型上打开多个笔记本时，即使它们使用不同的内核，笔记本也会在同一实例上运行。仅按一个实例的运行时间向您收取费用。

您可在打开笔记本后，在笔记本中更改实例类型。有关更多信息，请参阅 更改实例类型 (p. 94)。

有关计费和定价示例的信息，请参阅亚马逊 SageMaker 定价.

可用的资源

以下部分列出了 Amazon 的可用资源。 SageMaker Studio 笔记本。

主题

- Available SageMaker Studio 实例类型 (p. 98)
- Amazon 可用 SageMaker 映像 (p. 100)
- Amazon 可用 SageMaker 内核 (p. 100)

Available SageMaker Studio 实例类型

以下 Amazon Elastic Compute Cloud (Amazon EC2) 实例类型可用于 SageMaker Studio 笔记本。

有关哪些实例类型适合您的使用案例及其性能的详细信息，请参阅Amazon Elastic Compute Cloud 实例类型.

有关可用亚马逊的信息 SageMaker 笔记本实例类型，请参阅CreateNotebookInstance.

> **Note**
>
> 对于大多数使用案例，您应使用 ml.t3.Med。这是基于 CPU 的默认实例类型： SageMaker 映像，并且作为的一部分提供Amazon免费套餐.

>> 快速启动实例类型经过优化，可以在两分钟内启动。

默认实例类型

- 基于 CPU 的映像：ml.t3.medium>> 快速启动
- 基于 GPU 的映像：ml.g4dn.xlarge>> 快速启动

通用型（无 GPU）

- ml.t3.medium>> 快速启动
- ml.t3.large
- ml.t3.xlarge
- ml.t3.2xlarge
- ml.m5.large>> 快速启动
- ml.m5.xlarge
- ml.m5.2xlarge
- ml.m5.4xlarge
- ml.m5.8xlarge
- ml.m5.12xlarge
- ml.m5.16xlarge
- ml.m5.24xlarge

计算优化（没有 GPU）

- ml.c5.large>> 快速启动
- ml.c5.xlarge
- ml.c5.2xlarge
- ml.c5.4xlarge
- ml.c5.9xlarge
- ml.c5.12xlarge
- ml.c5.18xlarge
- ml.c5.24xlarge

加速计算（1+ GPU）

- ml.p3.2xlarge
- ml.p3.8xlarge
- ml.p3.16xlarge
- ml.g4dn.xlarge>> 快速启动

- ml.g4dn.2xlarge
- ml.g4dn.4xlarge
- ml.g4dn.8xlarge
- ml.g4dn.12xlarge
- ml.g4dn.16xlarge

Amazon 可用 SageMaker 映像

以下 SageMaker 图片在亚马逊上提供 SageMaker Studio。 SageMaker 图片包含最新亚马逊 SageMaker Python 开发工具包以及最新版本的内核。括号 ([]) 中的名称是 SageMaker 映像在 Amazon 资源名称 (ARN) 中指定 SageMaker 映像。有关更多信息，请参阅 。Deep Learning Containers 映像.

- 使用 Python 3.7 的数据科学 [数据科学 1.0]

  `Data Science`是Conda带有最常用 Python 程序包和库的映像，例如 NumPy 和 SciKit 学习。
- Base Python [python-3.6]
- MXNet（针对 CPU 优化）[mxnet-1.6-cpu-py36]
- MXNet（针对 GPU 优化）[mxnet-1.6-gpu-py36]
- MXNet（针对 CPU 优化）[mxnet-1.8-cpu-py37]
- MXNet（针对 GPU 优化）[mxnet-1.8-gpu-py37]
- PyTorch（针对 CPU 优化）[pytorch-1.4-cpu-py36]
- PyTorch（针对 GPU 优化）[pytorch-1.4-gpu-py36]
- PyTorch（针对 CPU 优化）[pytorch-1.6-cpu-py36]
- PyTorch（针对 GPU 优化）[pytorch-1.6-gpu-py36]
- PyTorch 1.8（针对 CPU 优化）[pytorch-1.8-cpu-py36]
- PyTorch 1.8（针对 GPU 优化）[pytorch-1.8-gpu-py36]
- 使用 Python 3.6 的火花魔术 [火花魔法 -1.0]
- TensorFlow（针对 CPU 优化）[tensorflow-1.15-cpu-py36]
- TensorFlow（针对 GPU 优化）[tensorflow-1.15-gpu-py36]
- TensorFlow（针对 CPU 优化）[tensorflow-1.15-cpu-py37]
- TensorFlow（针对 GPU 优化）[tensorflow-1.15-gpu-py37]
- TensorFlow 2（针对 CPU 优化）[tensorflow-2.1-cpu-py36]
- TensorFlow 2（针对 GPU 优化）[tensorflow-2.1-gpu-py36]
- TensorFlow 2（针对 CPU 优化）[tensorflow-2.3-cpu-py37]
- TensorFlow 2（针对 GPU 优化）[tensorflow-2.3-gpu-py37]
- TensorFlow 2.6（针对 CPU 优化）[tensorflow-2.6-cpu-py38]
- TensorFlow 2.6（针对 GPU 优化）[tensorflow-2.6-gpu-py38]

Amazon 可用 SageMaker 内核

以下亚马逊 SageMaker 内核可在 SageMaker Studio。括号中的名称是 SageMaker 托管内核的映像。

`Data Science`是Conda带有最常用 Python 程序包和库的映像，例如 NumPy 和 scikit-learn。

- Python 3.7（数据科学）
- Python 3.6 (PySpark Magic)
- Python 3.6（基础 Python）

- Python 3（MxNet 1.6 Python 3.6 CPU 优化）
- Python 3（MxNet 1.6 Python 3.6 GPU 优化）
- Python 3（MxNet 1.8 Python 3.7 CPU 优化）
- Python 3（MxNet 1.8 Python 3.7 GPU 优化）
- Python 3（PyTorch 1.4 Python 3.6 CPU 优化）
- Python 3（PyTorch 1.4 Python 3.6 GPU 优化）
- Python 3（PyTorch 1.6 Python 3.6 CPU 优化）
- Python 3（PyTorch 1.6 Python 3.6 GPU 优化）
- Python 3（PyTorch 1.8 Python 3.6 CPU 优化）
- Python 3（PyTorch 1.8 Python 3.6 GPU 优化）
- Python 3 (SageMaker) JumpStart 使用 Python 3.7 的数据科学 1.0)
- Python 3 (SageMaker) JumpStart 使用 Python 3.7 的 MxNet 1.0)
- Python 3 (SageMaker) JumpStart PyTorch 1.0) 使用 Python 3.7
- Python 3 (SageMaker) JumpStart 使用 Python 3.7 的 TensorFlow 1.0)
- Python 3（TensorFlow 1.15 Python 3.6 CPU 优化）
- Python 3（TensorFlow 1.15 Python 3.6 GPU 优化）
- Python 3（TensorFlow 1.15 Python 3.7 CPU 优化）
- Python 3（TensorFlow 1.15 Python 3.7 GPU 优化）
- Python 3（TensorFlow 2.1 Python 3.6 CPU 优化）
- Python 3（TensorFlow 2.1 Python 3.6 GPU 优化）
- Python 3（TensorFlow 2.3 Python 3.7 CPU 优化）
- Python 3（TensorFlow 2.3 Python 3.7 GPU 优化）
- Python 3（TensorFlow 2.6 Python 3.8 CPU 优化）
- Python 3（TensorFlow 2.6 Python 3.8 GPU 优化）

定制亚马逊 SageMaker 工作室

有两种选项可用于定制您的亚马逊。 SageMaker Studio 环境。自带 SageMaker 映像或使用生命周期配置脚本。这两个选项可以单独使用或一起使用。

- 自带 SageMaker 映像：一个 SageMaker 映像是一个文件，用于标识在 Amazon 中运行 Jupyter 笔记本所需的内核、语言包和其他依赖项。 SageMaker Studio。亚马逊 SageMaker 提供了许多内置映像供你使用。如果您需要不同的功能，可以将自己的自定义图像带到 Studio。
- 将生命周期配置与亚马逊 SageMaker Studio：生命周期配置是亚马逊触发的 shell 脚本 SageMaker Studio 生命周期事件，例如启动新的 Studio 笔记本电脑。您可以使用生命周期配置为 Studio 环境自动进行自定义。例如，您可以安装自定义软件包、配置笔记本扩展、预加载数据集以及设置源代码存储库。

以下主题介绍了如何使用这两个选项来定制您的 Amazon。 SageMaker Studio 环境。

主题

- 自带 SageMaker 图像 (p. 101)
- 将生命周期配置与亚马逊 SageMaker 工作室 (p. 116)

自带 SageMaker 图像

一个 SageMaker 映像是一个文件，用于标识在 Amazon 中运行 Jupyter 笔记本所需的内核、语言包和其他依赖项。 SageMaker Studio。这些映像用于创建一个环境，然后运行 Jupyter 笔记本电脑。亚马逊

SageMaker 提供了许多内置映像供你使用。如果您需要不同的功能，可以将自己的自定义图像带到 Studio。有关内置映像的列表，请参阅Amazon 可用 SageMaker 映像 (p. 100).

一个 SageMaker 图像是一组的持有者 SageMaker 映像版本. 映像版本表示与兼容的容器映像 SageMaker Studio 并存储在 Amazon Elastic Container Registry (ECR) 存储库中。每个映像版本都是不可变的。

要进行自定义 SageMaker 映像可供域中的所有用户使用，则可以将图像附加到域。要使图像可供单个用户使用，请将图像附加到该用户的个人资料。当你附加图片时， SageMaker 默认情况下使用最新的映像版本。您也可以附加特定的映像版本。在附加版本后，您可以从 SageMaker 启动笔记本电脑时启动器或图像选择器。

您可以使用 SageMaker Studio 控制面板、Amazon SDK for Python (Boto3)，以及Amazon Command Line Interface(Amazon CLI) . 您还可以使用 SageMaker 控制台，即使你还没有登陆 Studio。

SageMaker 提供了示例 DockerFile，用作您的自定义定义的起始点。 SageMaker 中的图片SageMaker Studio 自定义图像示例存储库。其中包括以下图像的 Dockerfile：

- 朱莉娅
- R
- Scala
- TensorFlow 2

以下主题介绍如何使用自带映像： SageMaker 控制台然后在中启动映像 SageMaker Studio。此外，还有一个更全面的教程向您展示了如何从提供的 R Dockerfile 构建自定义容器映像。有关类似的博客文章，请参阅将您自己的 R 环境带入 Amazon SageMaker 工作室. 有关展示如何自带图像用于训练和推理的笔记本，请参阅亚马逊 SageMaker Studio 容器构建 CLI.

主题

- 创建自定义 SageMaker 映像（控制台）(p. 102)
- 附加自定义 SageMaker 图像（控制面板）(p. 103)
- 启动自定义 SageMaker 映像 SageMaker 工作室 (p. 104)
- 自带自带自定义 SageMaker 映像教程 (p. 107)
- Custom (自定义) SageMaker 图片规格 (p. 115)

创建自定义 SageMaker 映像（控制台）

本主题介绍如何创建自定义 SageMaker 使用映像 SageMaker 控制台。您还可以使用 SageMaker Studio 控制面板。所用的步骤是相同的。有关使用 Studio 控制面板的信息，请参阅附加自定义 SageMaker 图像（控制面板）(p. 103).

当你创建图像时， SageMaker 还会创建初始映像版本。镜像版本表示中的容器镜像Amazon Elastic Container Registry (ECR). 容器映像必须满足在 Amazon SageMaker Studio 中使用的要求。有关更多信息，请参阅 Custom (自定义) SageMaker 图片规格 (p. 115)。

有关在本地测试映像和解决常见问题的信息，请参阅SageMaker Studio 自定义图像示例回购.

要创建映像

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在左侧导航窗格中，选择映像.
3. 在存储库的自定义映像页面，选择创建映像.
4. 适用于源镜像中，在 Amazon ECR 中输入容器映像的注册表路径。路径采用以下格式：

acct-id.dkr.ecr.*region*.amazonaws.com/*repo-name[:tag] or [@digest]*

5. 选择 Next（下一步）。
6. UDER映像属性中，输入以下内容：
   - 映像名称 — 该名称必须是当前账户的唯一名称Amazon区域。
   - （可选）显示名称 — Studio 用户界面中显示的名称。如果没有提供，`Image name`此时将显示。
   - （可选）描述 — 映像的描述。
   - IAM 角色 — 该角色必须具有AmazonSageMakerFullAccess附加策略。使用下拉菜单选择以下选项之一：
     - 创建新角色 — 指定您希望笔记本用户能够访问的任何额外 Amazon Simple Storage Service (Amazon S3) 存储桶。如果您不想允许访问其他存储桶，请选择None（无）.

       将 SageMaker 附加到`AmazonSageMakerFullAccess`对角色的策略。该角色允许笔记本电脑的用户访问复选标记旁边列出的 S3 存储桶。
     - 输入自定义 IAM 角色 ARN (IAM 角色 ARN) — 输入您的 IAM 角色的 Amazon 资源名称 (ARN)。
     - 使用现有角色 — 从列表中选择一个现有角色。
   - （可选）图片标签 — 选择添加新标签. 您最多可以添加 50 个标签。可以使用 Studio 用户界面、SageMaker 或控制台，或 SageMaker `Search`API。
7. 选择 Submit（提交）。

新图像将显示在自定义映像列出并简要突出显示。成功创建映像后，您可以选择映像名称以查看其属性或选择创建版本以创建另一个版本。

创建另一个映像版本

1. 选择创建版本在映像同一行上。
2. 适用于源镜像中，输入 ECR 容器镜像的注册表路径。容器镜像不应与以前版本的 SageMaker 映像。

要在 Studio 中使用自定义映像，必须将其附加到域。有关更多信息，请参阅 附加自定义 SageMaker 图像（控制面板）(p. 103)。

附加自定义 SageMaker 图像（控制面板）

使用自定义 SageMaker 图片，您必须将图片版本附加到域名。当你附加图像版本时，它会出现在 SageMaker 提供了 Studio 启动器，并在选择映像下拉列表，用户用于启动活动或更改笔记本电脑使用的图像。

在任何给定时间，可以附加的映像版本数有限制。达到限制后，必须分离版本才能附加另一个版本的映像。

将现有图片版本附加到您的域

本主题介绍如何附加现有自定义 SageMaker 使用图片版本到你的域 SageMaker Studio 控制面板。您还可以创建自定义 SageMaker 映像和图像版本，然后将该版本附加到您的域。

无论您使用 Studio 控制面板还是使用 Studio 控制面板还是 SageMaker 控制台。有关创建映像和映像版本的过程，请参阅创建自定义 SageMaker 映像（控制台）(p. 102).

要附加现有映像

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在左侧导航窗格中，选择控制面板.
3. 在存储库的控制面板，在Custom (自定义) SageMaker 附加到域的工作室图片，选择附加映像.

4. 适用于源镜像，选择现有图片.
5. 从列表中选择一个现有映像。
6. 从列表中选择映像的版本。
7. 选择 Next（下一步）。
8. 输入值映像名称、映像显示名称, 和说明.
9. 选择 IAM 角色。有关更多信息，请参阅 创建自定义 SageMaker 映像（控制台） (p. 102)。
10. （可选）为映像添加标签。
11. 选择 Next（下一步）。
12. UDERStudio 配置，输入或更改以下设置。有关如何从映像获取内核信息的信息，请参阅发展中的 SageMaker Studio 自定义图像示例存储库。

    - EFS 挂载路径 — 映像中用于挂载用户 Amazon Elastic File System EFS (EFS) 主目录的路径。
    - 内核：
      - 适用于内核名称在映像中输入现有内核的名称。
      - （可选）对于内核显示名称中，输入内核的显示名称。
      - 选择添加内核.
    - （可选）配置标签 — 选择添加新标签然后添加配置标签。

      Note

      有关更多信息，请参阅 。内核发现和用户数据的部分Custom (自定义) SageMaker 图片规格 (p. 115).
13. 选择提交

等待镜像版本附加到域。附加后，版本将显示在自定义映像列出并简要突出显示。

分离自定义 SageMaker 图像

当您从域中分离图像时，该映像的所有版本都将被分离。分离映像后，域的所有用户都将失去对映像版本的访问权限。

在分离版本时，映像版本上有内核会话的正在运行的笔记本将继续运行。当笔记本停止或内核关闭时，映像版本将变得不可用。

分离映像

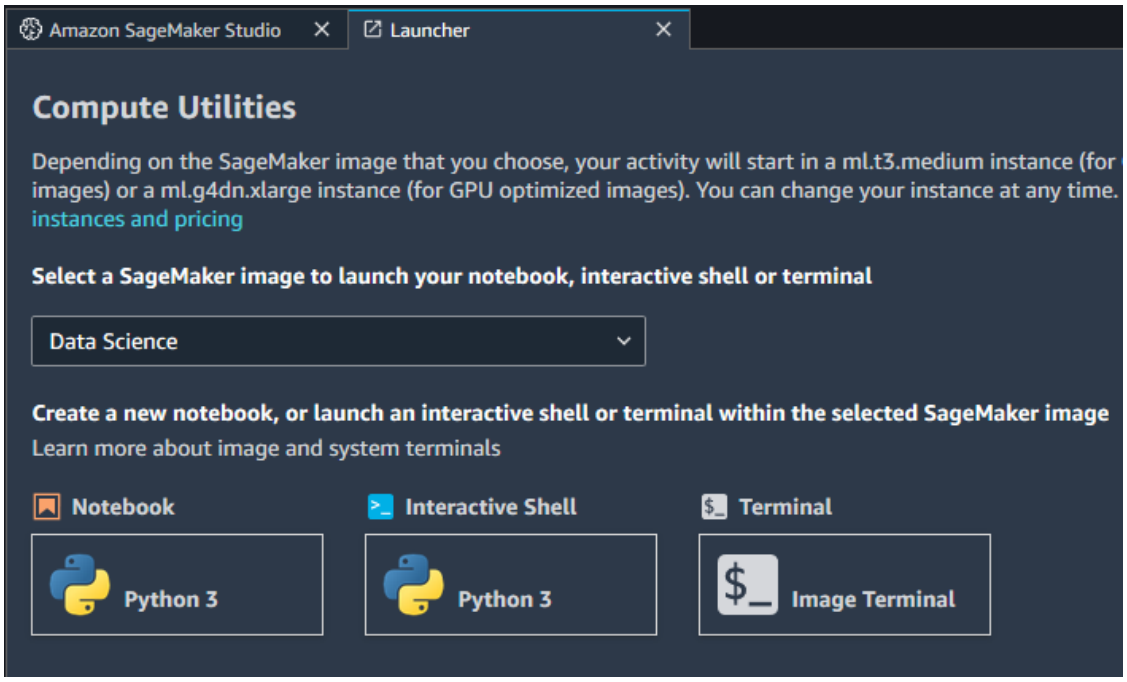
1. 在控制面板，在Custom (自定义) SageMaker 附加到域的工作室图片，选择图片然后选择Detach.
2. （可选）要从 SageMaker 中删除图像和所有版本，请选择同时删除选定的图像.... 这不会从 Amazon ECR 中删除相关容器映像。
3. 选择分离。

启动自定义 SageMaker 映像 SageMaker 工作室

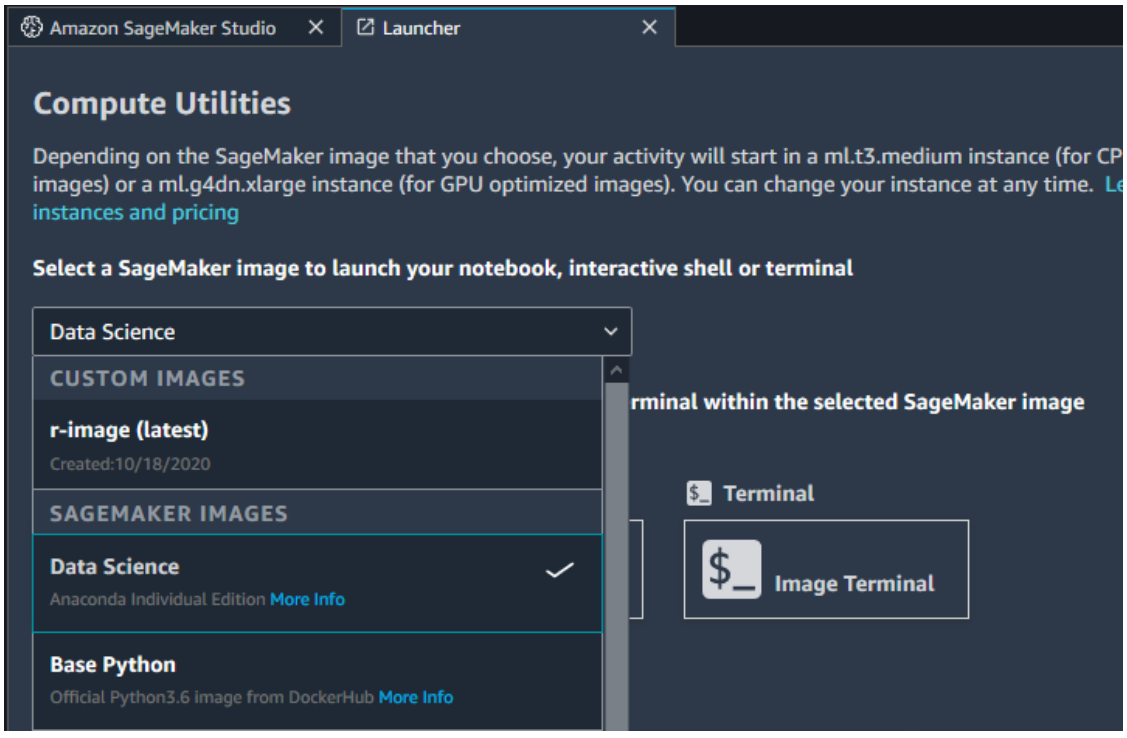
创建自定义之后 SageMaker 图片并将其附加到您的域名，图片将显示在 SageMaker Studio Launcher，内核将显示在内核选择器对话框中。

启动并选择自定义映像

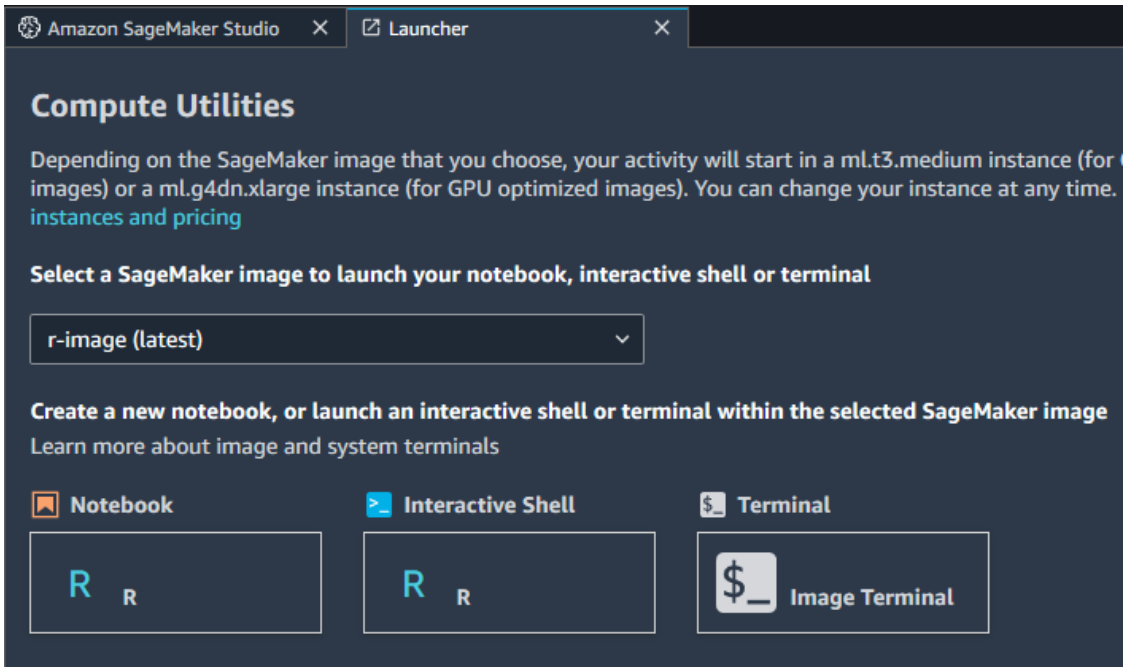
1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 使用键盘快捷键`Ctrl + Shift + L`打开 Studio 启动器。



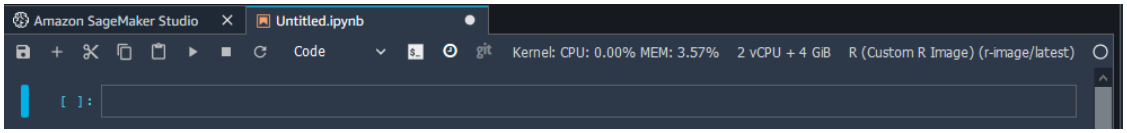
3. 打开在下选择 SageMaker 图像下拉菜单。



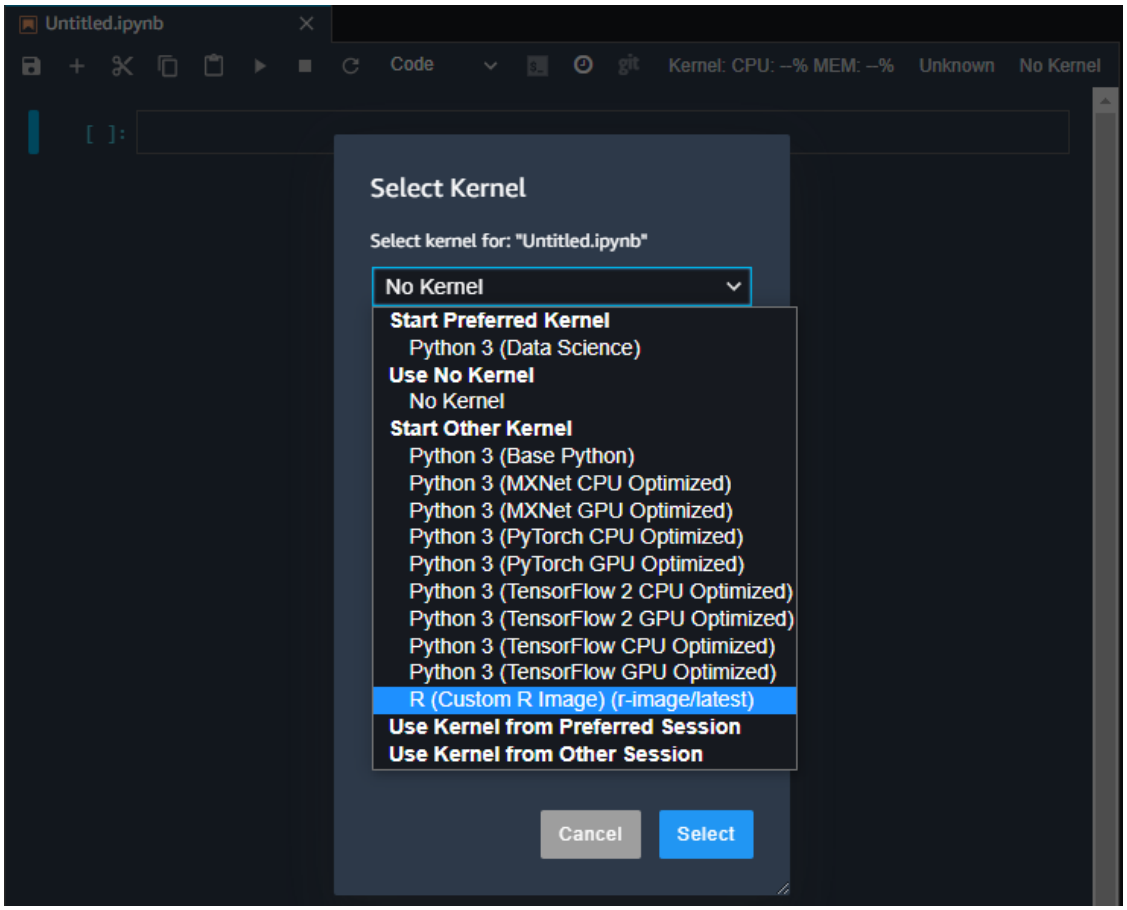
4. 选择自定义映像。



5. 在自定义映像中启动笔记本或交互式外壳。



6. 在打开的笔记本中，您可以通过在选择内核对话框。



Note

如果您在启动图片时遇到错误，请检查您的亚马逊 CloudWatch 日志。日志组的名称是`/aws/sagemaker/studio`. 日志流的名称是`$domainID/$userProfileName/KernelGateway/$appName`.

自带自带自定义 SageMaker 映像教程

在本教程中，您将创建自定义 SageMaker 映像并将图片版本附加到您的域名以便在亚马逊中使用 SageMaker Studio。映像版本包含精选的 R 软件包以及Amazon SDK for Python (Boto3)和亚马逊 SageMaker Python 开发工具包. 完成本教程后，您可以在 Studio 中选择版本，然后使用 R 使用 rStudio 网状程序包访问 SDK。有关网状软件包的更多信息，请参阅R 到 Python 的接口. 有关与本教程类似的博客文章，请参阅将您自己的 R 环境带入 Amazon SageMaker 工作室.

有两种方法将图像版本附加到您的域名。在第一种方法中，您创建一个附加了版本的新域。此方法更简单，但是您需要指定创建域所需的 Amazon Virtual Private Cloud (VPC) 信息和执行角色。

如果您已经加入 Studio，则可以使用第二种方法将映像版本附加到当前域。在这种情况下，不需要指定 VPC 信息和执行角色。附加版本后，必须删除域中的所有应用程序并重新打开 Studio。

您无法从 Studio 运行本教程，因为您无法从 Studio 内创建或更新域。

先决条件

- Docker 应用程序。有关设置 Docker 的信息，请参阅。定向和设置.
- 用于创建兼容 Studio 的 Dockerfile 的本地副本R 映像来自 的SageMaker Studio 自定义图像样本存储库。

Note

从 Dockerfile 构建 R 映像会安装依赖项，这些依赖项可以使用 copyleft 许可证（例如 gPLv3）进行许可。在继续构建此映像之前，您应查看许可条款并确保它们适合您的使用案例。

- 访问 Amazon Elastic Container Registry (Amazon ECR) 服务的权限。有关更多信息，请参阅 。Amazon ECR 托管策略.
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon Identity and Access Management具有AmazonSageMakerFullAccess附加策略。如果你已经加入亚马逊 SageMaker Studio，你可以从Studio 摘要的 部分 SageMaker Studio 控制面板。

链接

- Amazon Elastic Container Registry API 参考
- Docker 引擎 API
- Dockerfile 参考

主题

- 将兼容 Studio 的容器镜像添加到亚马逊 ECR (p. 108)
- 创建 SageMaker ECR 容器镜像中的映像 (p. 109)
- 将 SageMaker 映像到新域 (p. 111)
- 将 SageMaker 映像到你当前域名 (p. 112)
- 在 Studio 控制面板中查看附加的图像 (p. 114)
- 清理 资源 (p. 114)

将兼容 Studio 的容器镜像添加到亚马逊 ECR

您可以执行以下步骤将容器镜像添加到 Amazon ECR：

- 创建 Amazon ECR 存储库。
- 向 Amazon ECR 进行身份验证。
- 构建兼容 Studio 的容器镜像。
- 将映像推送到 Amazon ECR 存储库。

Note

Amazon ECR 存储库必须在相同的位置Amazon地区为 SageMaker Studio。

构建容器映像并将其添加到 Amazon ECR

1. 使用Amazon CLI. 要使用 Amazon ECR 控制台创建存储库，请参阅创建存储库.

```
aws ecr create-repository \
    --repository-name smstudio-custom \
    --image-scanning-configuration scanOnPush=true
```

响应:

```
{
    "repository": {
```

```
        "repositoryArn": "arn:aws:ecr:us-east-2:acct-id:repository/smstudio-custom",
        "registryId": "acct-id",
        "repositoryName": "smstudio-custom",
        "repositoryUri": "acct-id.dkr.ecr.us-east-2.amazonaws.com/smstudio-custom",
        ...
    }
}
```

2. 安装 SageMaker 请按照中的步骤构建 Studio 映像构建 CLISageMaker Docker 构建. 使用此 CLI，您可以使用Amazon CodeBuild.
3. 构建 R 映像`Dockerfile`使用 Studio 映像构建 CLI。句点 (.) 指定 Dockerfile 应该在构建命令的上下文中。此命令构建映像，创建 ECR 存储库，然后将构建的映像上传到 ECR 存储库。然后，它会输出映像 URI。

```
sm-docker build . -t smstudio-r -t <acct-id>.dkr.ecr.<region>.amazonaws.com/smstudio-
custom:r
```

响应:

```
Image URI: <acct-id>.dkr.ecr.<region>.amazonaws.com/<image_name>
```

创建 SageMaker ECR 容器镜像中的映像

请执行下列步骤来创建 SageMaker 从容器映像中：

- 创建 `Image`。
- 创建 `ImageVersion`。
- 创建配置文件。
- 创建 `AppImageConfig`。

创建 SageMaker 映像实体

1. 创建 SageMaker 映像。

```
aws sagemaker create-image \
    --image-name r-image \
    --role-arn arn:aws:iam::<acct-id>:role/service-role/<execution-role>
```

响应:

```
{
    "ImageArn": "arn:aws:sagemaker:us-east-2:acct-id:image/r-image"
}
```

2. 创建 SageMaker 来自容器映像的映像版本。

```
aws sagemaker create-image-version \
    --image-name r-image \
    --base-image <acct-id>.dkr.ecr.<region>.amazonaws.com/smstudio-custom:r
```

响应:

```
{
    "ImageVersionArn": "arn:aws:sagemaker:us-east-2:acct-id:image-version/r-image/1"
```

```
}
```

3. 检查映像版本是否已成功创建。

```
aws sagemaker describe-image-version \
    --image-name r-image \
    --version 1
```

响应:

```
{
    "ImageVersionArn": "arn:aws:sagemaker:us-east-2:acct-id:image-version/r-image/1",
    "ImageVersionStatus": "CREATED"
}
```

Note

如果响应如此`"ImageVersionStatus": "CREATED_FAILED"`，答复还包括失败原因。权限问题是导致失败的常见原因。你也可以查看你的亚马逊 CloudWatch 日志。日志组的名称是`/aws/sagemaker/studio`. 日志流的名称是`$domainID/$userProfileName/KernelGateway/$appName`.

4. 创建配置文件，名为`app-image-config-input.json`. 这些区域有：`Name`的值`KernelSpecs`必须与此相关的映像中可用的 KernelSpec 名称匹配。`AppImageConfig`. 此值区分大小写。您可以通过运行以下方法在映像中找到可用的 KernelSpeces`jupyter-kernelspec list`来自容器内的外壳。`MountPath`是映像中挂载 Amazon Elastic File System (Amazon EFS) 主目录的路径。它需要与您在容器内使用的路径不同，因为挂载 Amazon EFS 主目录时，该路径将被覆盖。有关在 Studio 中使用之前在本地测试图像的信息，请参阅发展中的 SageMaker Studio 自定义图像示例存储库。

Note

以下`DefaultUID`和`DefaultGID`组合是唯一可接受的值：

- defaultuID：1000 和 defaultGID：100
- defaultuID：0 和 defaultGID：0

```
{
    "AppImageConfigName": "r-image-config",
    "KernelGatewayImageConfig": {
        "KernelSpecs": [
            {
                "Name": "ir",
                "DisplayName": "R (Custom R Image)"
            }
        ],
        "FileSystemConfig": {
            "MountPath": "/home/sagemaker-user",
            "DefaultUid": 1000,
            "DefaultGid": 100
        }
    }
}
```

5. 创建 AppImageConfig 使用在上一步中创建的文件。

```
aws sagemaker create-app-image-config \
    --cli-input-json file://app-image-config-input.json
```

响应:

```
{
    "AppImageConfigArn": "arn:aws:sagemaker:us-east-2:acct-id:app-image-config/r-image-config"
}
```

将 SageMaker 映像到新域

要使用此方法，您需要指定一个执行角色，该角色具有AmazonSageMakerFullAccess附加策略。

> **Note**
>
> 您只能有一个域。如果你已经登上 SageMaker Studio，您必须先删除当前域名，然后才能使用此方法。有关更多信息，请参阅 删除 Amazon SageMaker 域 (p. 36)。

您可以执行以下步骤来创建域并附加自定义。 SageMaker 映像：

- 获取默认 VPC ID 和子网 ID。
- 为域创建配置文件，用于指定映像。
- 使用配置文件创建域。

添加自定义 SageMaker 映像到您的域

1. 获取默认 VPC ID。

```
aws ec2 describe-vpcs \
    --filters Name=isDefault,Values=true \
    --query "Vpcs[0].VpcId" --output text
```

响应:

```
vpc-xxxxxxxx
```

2. 使用上一步中的 VPC ID 获取默认子网 ID。

```
aws ec2 describe-subnets \
    --filters Name=vpc-id,Values=<vpc-id> \
    --query "Subnets[*].SubnetId" --output json
```

响应:

```
[
    "subnet-b55171dd",
    "subnet-8a5f99c6",
    "subnet-e88d1392"
]
```

3. 创建一个名为的配置文件`create-domain-input.json`. 插入 VPC ID、子网 ID、`ImageName`, 和`AppImageConfigName`从前面的步骤中。由于`ImageVersionNumber`未指定，则使用最新版本的图像，这是这种情况下唯一的版本。

```
{
    "DomainName": "domain-with-custom-r-image",
    "VpcId": "<vpc-id>",
    "SubnetIds": [
```

```
        "<subnet-ids>"
    ],
    "DefaultUserSettings": {
        "ExecutionRole": "<execution-role>",
        "KernelGatewayAppSettings": {
            "CustomImages": [
                {
                    "ImageName": "r-image",
                    "AppImageConfigName": "r-image-config"
                }
            ]
        }
    },
    "AuthMode": "IAM"
}
```

4. 使用附加的自定义创建域 SageMaker 映像。

```
aws sagemaker create-domain \
    --cli-input-json file://create-domain-input.json
```

响应:

```
{
    "DomainArn": "arn:aws:sagemaker:us-east-2:acct-id:domain/d-xxxxxxxxxxxx",
    "Url": "https://d-xxxxxxxxxxxx.studio.us-east-2.sagemaker.aws/..."
}
```

将 SageMaker 映像到你当前域名

此方法假定您已经加入亚马逊 SageMaker Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。

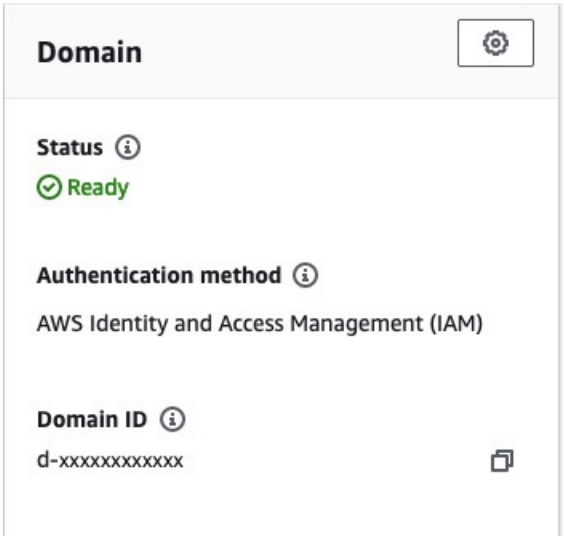
> **Note**
>
> 在使用新映像版本更新域名之前，必须先删除域中的所有应用程序。有关删除应用程序的信息，请参阅。删除 Amazon SageMaker 域 (p. 36).

请执行下列步骤来添加 SageMaker 映像到你当前域名。

- 获取您的`DomainID`从 SageMaker Studio。
- 使用`DomainID`获取`DefaultUserSettings`对于域。
- 添加`ImageName`和`AppImageConfig`作为`CustomImage`到`DefaultUserSettings`.
- 更新域名以包含自定义映像。

添加自定义 SageMaker 映像到您的域

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 从导航窗格的左上角选择控制面板.
3. 从控制面板，在域，查找`DomainId`. ID 采用以下格式：`d-xxxxxxxxxxxx`.



4. 使用域 ID 获取域的描述。

```
aws sagemaker describe-domain \
    --domain-id <d-xxxxxxxxxxxx>
```

响应:

```
{
    "DomainId": "d-xxxxxxxxxxxx",
    "DefaultUserSettings": {
      "KernelGatewayAppSettings": {
        "CustomImages": [
        ],
        ...
      }
    }
}
```

5. 将响应的默认用户设置部分保存到名为的文件`default-user-settings.json`.
6. 插入`ImageName`和`AppImageConfigName`从前面的步骤中将其作为自定义映像。由于`ImageVersionNumber`未指定，则使用最新版本的图像，这是这种情况下唯一的版本。

```
{
    "DefaultUserSettings": {
        "KernelGatewayAppSettings": {
           "CustomImages": [
              {
                 "ImageName": "string",
                 "AppImageConfigName": "string"
              }
           ],
           ...
        }
    }
}
```

7. 使用域 ID 和默认用户设置文件更新您的域。

```
aws sagemaker update-domain \
    --domain-id <d-xxxxxxxxxxxx> \
```

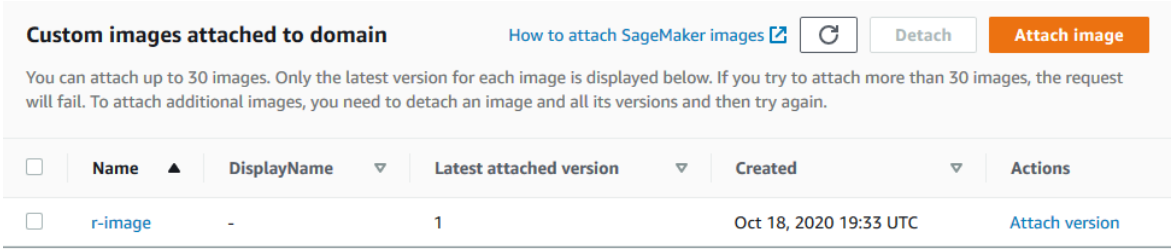
```
    --cli-input-json file://default-user-settings.json
```

响应:

```
{
    "DomainArn": "arn:aws:sagemaker:us-east-2:acct-id:domain/d-xxxxxxxxxxxx"
}
```

在 Studio 控制面板中查看附加的图像

在创建自定义后 SageMaker 图像并将其附加到您的域中，图像将显示在控制面板的自定义图像列表中。



有关如何在亚马逊启动映像的信息 SageMaker 工作室，请参阅启动自定义 SageMaker 映像 SageMaker 工作室 (p. 104).

清理 资源

您可以执行以下步骤来清理在前面部分中创建的资源：

- 从域中分离图像和图像版本。
- 删除映像、映像版本和应用程序映像配置。
- 从 Amazon ECR 中删除容器映像和存储库。有关更多信息，请参阅 。删除存储库.

清除资源

1. 通过将空的自定义映像列表传递给域名，从域中分离图像和映像版本。打开`default-user-settings.json`你在中创建的文件将图像附加到当前域 (p. 112).
2. 删除自定义映像，然后保存文件。

```
"DefaultUserSettings": {
  "KernelGatewayAppSettings": {
     "CustomImages": [
     ],
     ...
  },
  ...
}
```

3. 使用域 ID 和默认用户设置文件更新您的域。

```
aws sagemaker update-domain \
    --domain-id <d-xxxxxxxxxxxx> \
    --cli-input-json file://default-user-settings.json
```

响应:

```
{
```

```
    "DomainArn": "arn:aws:sagemaker:us-east-2:acct-id:domain/d-xxxxxxxxxxxx"
}
```

4. 删除应用程序映像配置。

```
aws sagemaker delete-app-image-config \
    --app-image-config-name r-image-config
```

5. 删除存储在设备上的 SageMaker 映像，它还会删除所有映像版本。不会删除由映像版本表示的 ECR 中的容器镜像。

```
aws sagemaker delete-image \
    --image-name r-image
```

Custom (自定义) SageMaker 图片规格

以下规范适用于由 SageMaker 映像版本表示的容器映像。

运行图像

ENTRYPOINT和CMD指令被覆盖以允许映像作为 KernelGateway 应用程序。

镜像中的端口 8888 保留用于运行 KernelGateway Web 服务器的任务。

停止映像

这些区域有：DeleteAppAPI 发出相当于docker stop命令。容器中的其他进程不会获得 SIGKILL/SIGTERM 信号。

内核发现

SageMaker 识别 Jupyter 定义的内核内核规范.

您可以在运行映像之前指定要显示的内核列表。如果未指定，则会显示 python3。使用描述AppImageConfig查看内核列表的 API。

默认情况下，Conda 环境被识别为内核规范。

文件系统

这些区域有：/opt/.sagemakerinternal和/opt/ml保留了目录。这些目录中的任何数据在运行时都可能不可见。

用户数据

Studio 域中的每个用户都会在映像中的共享 Amazon Elastic File System 卷上获取一个用户目录。当前用户目录在 Amazon EFS 卷上的位置是可配置的。默认情况下，目录的位置为/home/sagemaker-user.

SageMaker 在映像和主机之间配置 POSIX UID/GID 映射。默认情况下，将 root 用户的 UID/GID (0/0) 映射到主机上的 UID/GID。

您可以使用创建应用程序图像配置API。

GID/UID 限制

SageMaker Studio 仅支持介于 0 到 65535 之间的 UID 和 GID 值。此限制适用于图像每个图层中的文件。

Metadata

元数据文件位于/opt/ml/metadata/resource-metadata.json. 不会向图像中定义的变量添加任何其他环境变量。有关更多信息，请参阅 获取应用程序元数据 (p. 92)。

GPU

在 GPU 实例上，映像使用`--gpus`选项。映像中应包含 CUDA 工具包，而不是 NVIDIA 驱动程序。有关更多信息，请参阅 。NVIDIA 用户指南.

指标和日志记录

中的日志 KernelGateway 将流程发送到 Amazon CloudWatch 在客户的账户中。日志组的名称是`/aws/sagemaker/studio`. 日志流的名称是`$domainID/$userProfileName/KernelGateway/$appName`.

图片大小

限制为 25 GB。要查看映像的大小，请运行`docker image ls`.

将生命周期配置与亚马逊 SageMaker 工作室

生命周期配置是亚马逊触发的 shell 脚本 SageMaker Studio 生命周期事件，例如启动新的 Studio 笔记本电脑。您可以使用生命周期配置为 Studio 环境自动进行自定义。此自定义包括安装自定义软件包、配置笔记本扩展、预加载数据集以及设置源代码存储库。

使用生命周期配置可让您灵活地控制配置 Studio 以满足您的特定需求。例如，您可以使用最常用的软件包和库创建一组最少的基本容器映像，然后使用生命周期配置为数据科学和机器学习团队中的特定用例安装其他软件包。

例如生命周期配置脚本，请参阅Studio 生命周期配置示例回购. 有关实施生命周期配置的博客，请参阅定制亚马逊 SageMaker Studio 使用生命周期配置.

主题

- 创建和关联生命周期配置 (p. 116)
- 设置默认周期配置 (p. 122)
- 对生命周期配置 (p. 123)
- 更新和删除生命周期配置 (p. 125)

创建和关联生命周期配置

亚马逊 SageMaker Studio 应用程序是交互式应用程序，可实现 Studio 的可视化界面、代码创作和执行体验。应用程序类型可以是 jupyterServer 或 KernelGateway。

- jupyterServer 应用程序：此应用程序类型允许访问 Studio 的可视化界面。Studio 中的每个用户都有自己的 JupyterServer 应用程序。
- KernelGateway 应用程序：此应用程序类型允许访问 Studio 笔记本电脑和终端的代码执行环境和内核。有关更多信息，请参阅 。Jupyter 内核网关.

有关 Studio 体系结构和 Studio 应用程序的更多信息，请参阅使用 Amazon SageMaker Studio Notebooks.

主题

- 先决条件 (p. 117)
- 第 1 步：创建新的生命周期配置 (p. 117)
- 第 2 步：将生命周期配置附加到你的 Studio 域或 UserProfile (p. 117)
- 第 3 步：启动新应用程序时选择生命周期配置 (p. 118)
- 第 4 步：查看生命周期配置的日志 (p. 120)

先决条件

- 确保你的Amazon CLI使用中的步骤处于最新状态安装最新的AmazonCLI 版本.
- 从本地计算机上运行`aws configure`提供您的Amazon凭证。有关Amazon凭据，请参阅了解并获取您的Amazon证书.
- 注册到亚马逊 SageMaker Studio。有关更多信息，请参阅 。注册到亚马逊 SageMaker 工作室.

第 1 步：创建新的生命周期配置

以下过程介绍如何创建打印的生命周期配置脚本。`Hello World`.

1. 从本地计算机上，创建一个名为`my-script.sh`包含以下内容。

```
#!/bin/bash
set -eux
echo 'Hello World!'
```

2. 转换您的`my-script.sh`将文件转换为 Base64 格式。此要求可防止由于间距和换行符的编码而导致的错误。

```
LCC_CONTENT=`openssl base64 -A -in my-script.sh`
```

3. 创建 Studio 生命周期配置。以下命令创建一个生命周期配置，该配置在启动关联的 KernelGateway 应用程序。

```
aws sagemaker create-studio-lifecycle-config \
--region <your-region> \
--studio-lifecycle-config-name my-studio-lcc \
--studio-lifecycle-config-content $LCC_CONTENT \
--studio-lifecycle-config-app-type KernelGateway
```

   记下返回的新创建的生命周期配置的 ARN。将生命周期配置附加到您的应用程序需要使用此 ARN。

第 2 步：将生命周期配置附加到你的 Studio 域或 UserProfile

您需要更新 UserSettings 对于你的 Studio 域或个人 UserProfile 以附加生命周期配置。在域级别关联的生命周期配置脚本由所有用户继承，而在 UserProfile 级别的范围限定为特定用户。

以下示例显示了如何创建一个新的 UserProfile 附加生命周期配置。如果要更新现有的 UserProfile (用户配置文件)，请使用`update-user-profile`命令。

将上一步中的生命周期配置 ARN 添加到相应 AppType 的设置中。例如，将其放在`JupyterServerAppSettings`用户的。您可以使用生命周期配置列表一次添加多个生命周期配置。

```
# Create a new UserProfile
aws sagemaker create-user-profile --domain-id <DOMAIN-ID> \
--user-profile-name <USER-PROFILE-NAME> \
--region <REGION> \
--user-settings '{
"JupyterServerAppSettings": {
  "LifecycleConfigArns":
    ["<LIFECYCLE-CONFIGURATION-ARN-LIST>"]
  }
}'
```

第 3 步：启动新应用程序时选择生命周期配置

将生命周期配置附加到 UserProfile 后，用户可以在启动应用程序时选择它。启动应用程序的两种方法是使用Amazon CLI并通过 Studio 启动程序。以下几节介绍了如何使用这两种方法启动应用程序。

使用启动应用Amazon CLI

启动应用程序并 ARN`ResourceSpec`参数 CreateApp API。

- 以下示例显示了如何创建 JupyterServer 应用程序。创建 JupyterServer 应用程序，应用程序名称必须是`default`.

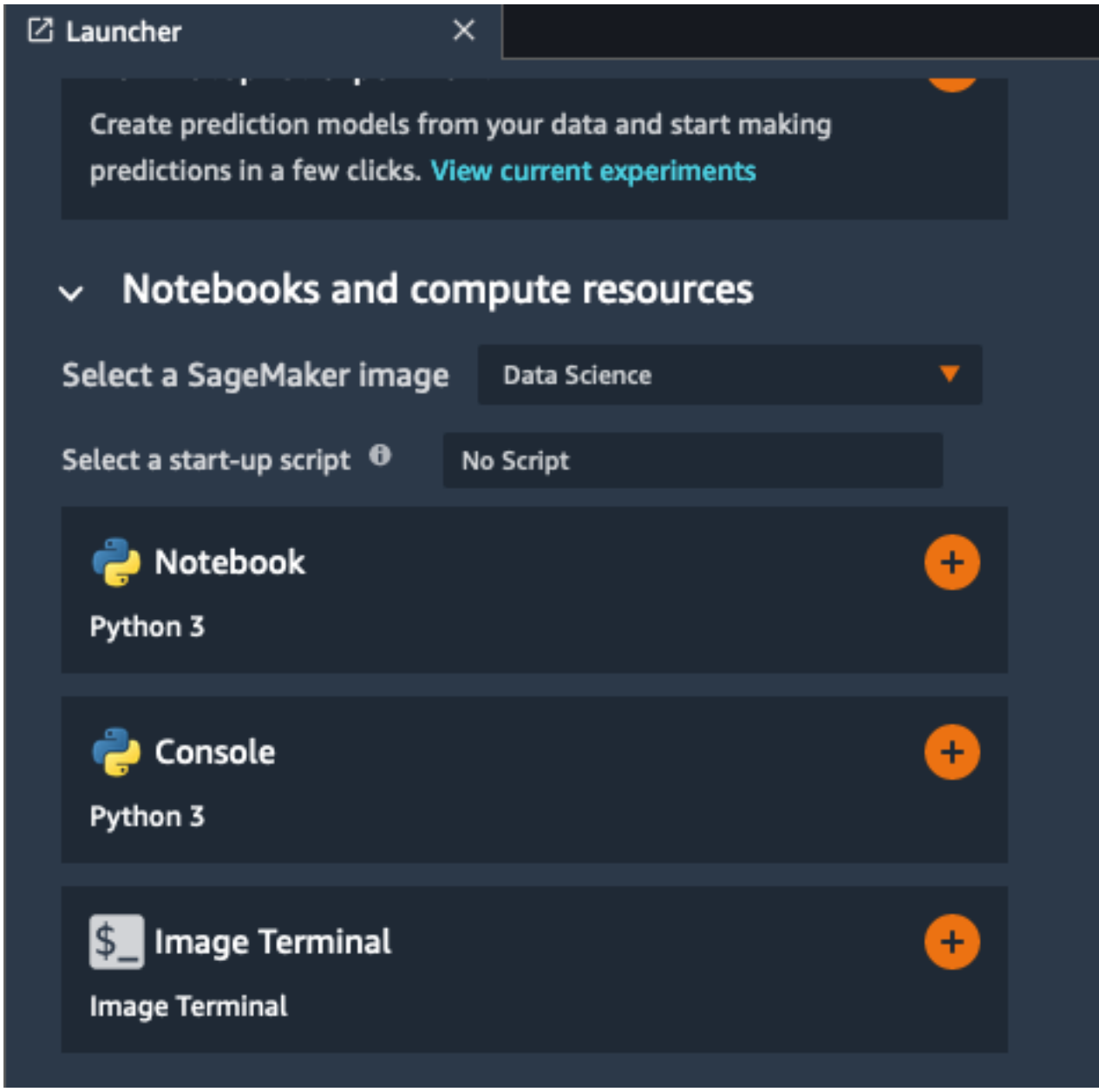
```
aws sagemaker create-app --domain-id <DOMAIN-ID> \
--region <YOUR-REGION> \
--user-profile-name <USERPROFILE-NAME> \
--app-type JupyterServer \
--resource-spec LifecycleConfigArn=<LIFECYCLE-CONFIGURATION-ARN> \
--app-name default
```

- 以下示例显示了如何创建 KernelGateway 应用程序。

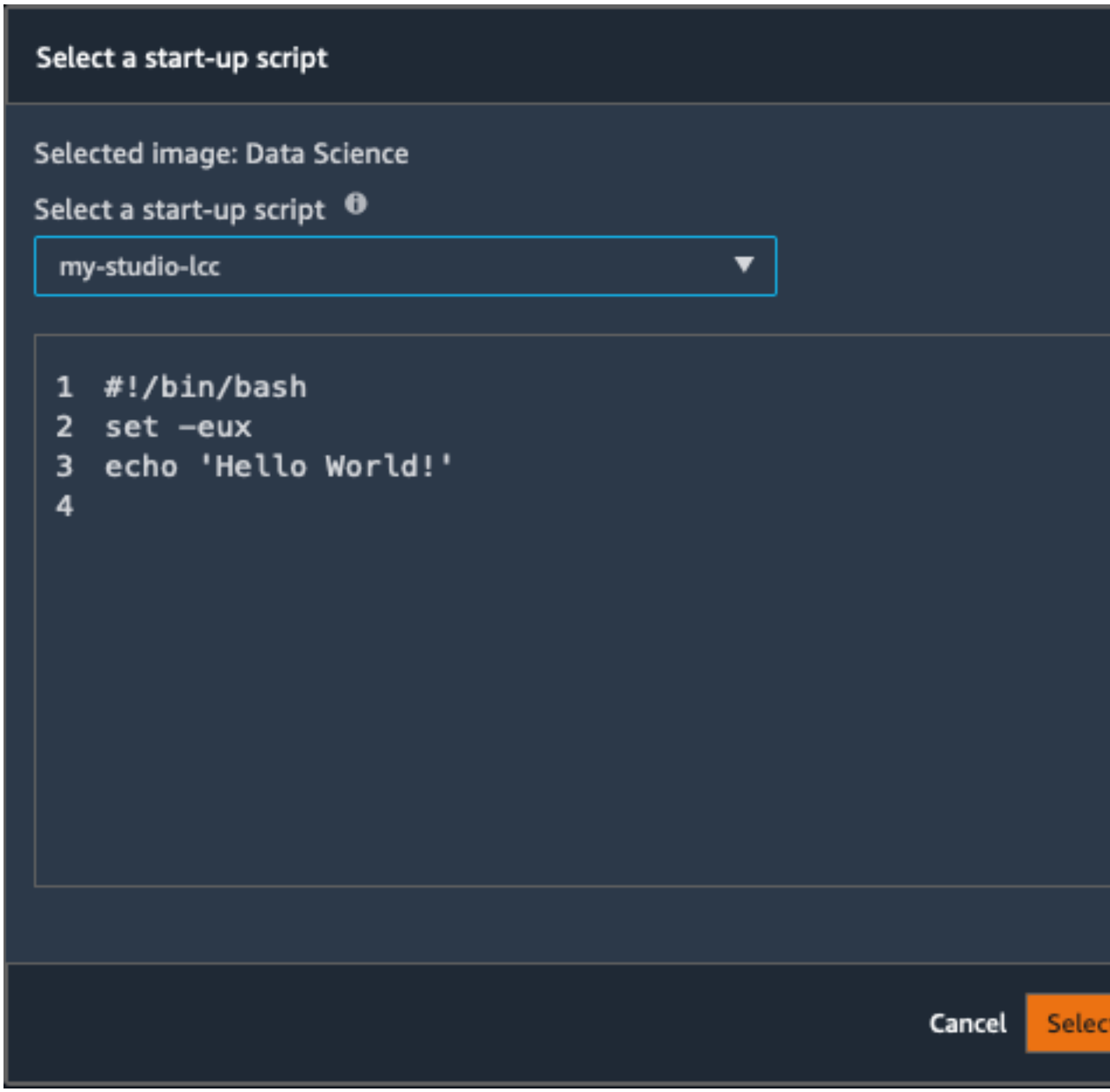
```
aws sagemaker create-app --domain-id <DOMAIN-ID> \
--region <YOUR-REGION> \
--user-profile-name <USERPROFILE-NAME> \
--resource-spec LifecycleConfigArn=<LIFECYCLE-CONFIGURATION-
ARN>,SageMakerImageArn=<SAGEMAKER-IMAGE-ARN>,InstanceType=<INSTANCE-TYPE> \
--app-type KernelGateway \
--app-name <APP-NAME>
```

启动 KernelGateway 使用 Studio 启动程序的应用程序

1. 启动 Studio 域。有关更多信息，请参阅 使用 Amazon SageMaker Studio 启动器 (p. 80)。
2. 在启动器中，导航到`Notebooks and compute resources`部分。



3. 选择您的 SageMaker 映像。
4. 选择启动脚本。如果没有默认的生命周期配置，则此值默认为`No script`. 否则，此值等于默认的生命周期配置。选择生命周期配置后，您可以查看整个脚本。

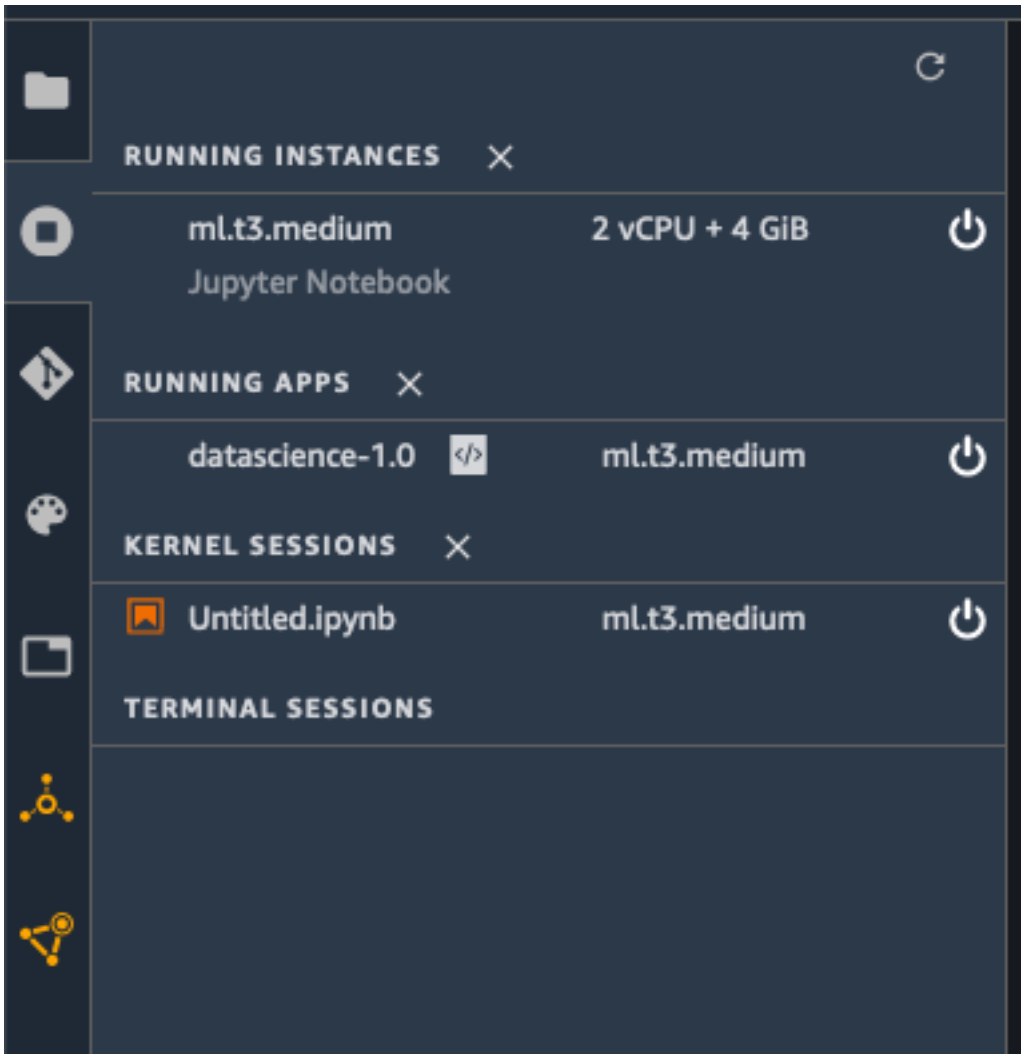


5. Select`Notebook`使用所选映像和生命周期配置启动新的笔记本内核。

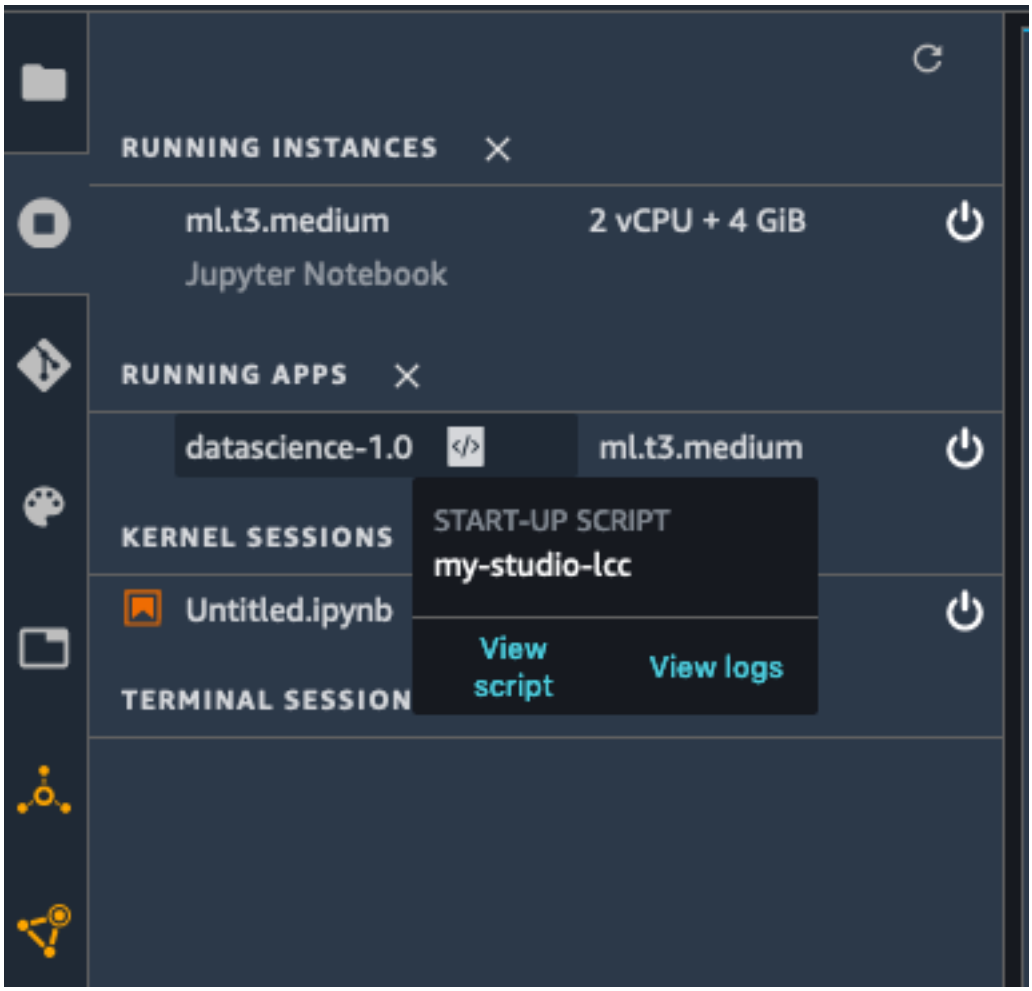
第 4 步：查看生命周期配置的日志

在将生命周期配置附加到 Studio 域或 UserProfile 后，您可以查看其日志。

1. 查看 CloudWatch 生命周期配置的日志，您必须首先提供访问 CloudWatch 对于您的 IAM 角色。您需要以下日志组的读取权限`/aws/sagemaker/studio`以及以下日志流`{Domain}/{UserProfile}/{AppType}/{AppName}/LifecycleConfigOnStart`. 有关添加权限的信息，请参阅。启用特定日志记录Amazon服务.
2. 要监控生命周期配置，请导航到`Running instances`选项卡。



3. 从正在运行的应用程序列表中选择应用程序。附加了生命周期配置的应用程序有附加的指标 .
4. 单击应用程序的指标图标。这将打开一个新面板，其中列出了生命周期配置。



5. 在新面板中，选择`View logs`. 这将打开一个显示日志的新选项卡。

设置默认周期配置

要以编程方式将生命周期配置设置为域或 UserProfile 的默认配置，您可以创建新资源或更新现有资源。要将生命周期配置关联为默认值，首先需要按照中的步骤创建生命周期配置创建和关联生命周期配置 (p. 116). 在域级别设置的默认生命周期配置由所有用户继承，而在用户级别设置的默认生命周期配置则限定为特定用户。

> Note
>
> 用户级别默认值将覆盖在域级别设置的默认值。

要设置默认的生命周期配置，必须将其添加到`DefaultResourceSpec`适当的应用程序类型。生命周期配置的行为取决于是否将其添加到`DefaultResourceSpec`的 JupyterServer 要么 KernelGateway 应用程序。

- jupyterServer 应用程序：当添加到`DefaultResourceSpec`的 JupyterServer 应用程序，当用户首次登录 Studio 或重新启动 Studio 时，默认的生命周期配置脚本将自动运行。这可用于为 Studio 开发人员环境自动执行一次性设置操作，例如安装笔记本扩展程序或设置 GitHub 存储库。有关此示例，请参阅定制亚马逊 SageMaker Studio 使用生命周期配置.
- KernelGateway 应用程序：当添加到`DefaultResourceSpec`的 KernelGateway app，Studio 默认为从 Studio 启动器中选择生命周期配置脚本。用户可以在选择默认脚本的情况下启动笔记本或终端，也可以从生命周期配置列表中选择其他脚本。

Note

一个默认 KernelGateway 在中指定的生命周期`DefaultResourceSpec`适用于所有 KernelGateway 除非用户从 Studio 启动器中显示的列表中选择了不同的脚本，否则 Studio Domain 中的映像。默认脚本也会运行`No Script`是由用户选择的。有关如何选择脚本的更多信息，请参阅第 3 步：启动新应用程序时选择生命周期配置 (p. 118).

在创建新域或 UserProfile 时关联默认的生命周期配置

要在创建新的 Studio 域或 UserProfile 时关联生命周期配置，您需要创建的生命周期配置的 ARN。此 ARN 传递给以下 API 调用之一：

- 创建用户配置文件
- 创建域

例如，以下 API 调用创建了一个新的 UserProfile 使用关联的生命周期配置。

```
aws sagemaker create-user-profile --domain-id <DOMAIN-ID> \
--user-profile-name <USER-PROFILE-NAME> \
--region <REGION> \
--user-settings '{
"KernelGatewayAppSettings": {
    "DefaultResourceSpec": {
            "InstanceType": "ml.t3.medium",
            "LifecycleConfigArn": "<LIFECYCLE-CONFIGURATION-ARN>"
        }
  }
}'
```

更新域或用户配置文件时关联默认的生命周期配置

要在更新现有 Studio 域或 UserProfile 时关联生命周期配置，您需要创建的生命周期配置的 ARN。此 ARN 传递给以下 API 调用之一：

- 更新用户配置文件
- 更新域

生命周期配置 ARN 应放在两个位置，`DefaultResourceSpec`和`LifecycleConfigArns`在中列出`KernelGatewayAppSettings`. 例如，以下 API 调用更新了 UserProfile 使用关联的生命周期配置。

```
aws sagemaker update-user-profile --domain-id <DOMAIN-ID> \
--user-profile-name <USER-PROFILE-NAME> \
--region <REGION> \
--user-settings '{
"KernelGatewayAppSettings": {
    "DefaultResourceSpec": {
            "InstanceType": "ml.t3.medium",
            "LifecycleConfigArn": "<LIFECYCLE-CONFIGURATION-ARN>"
        }
  }
}'
```

对生命周期配置

以下主题介绍了如何获取有关生命周期配置的信息和调试。

主题

- 验证亚马逊的生命周期配置流 CloudWatch 日志 (p. 124)

- JupyterServer 应用程序失败 (p. 124)
- KernelGateway 应用程序故 (p. 124)
- 周期 Config 超时 (p. 125)

验证亚马逊的生命周期配置流 CloudWatch 日志

仅记录生命周期配置`STDOUT`和`STDERR`.`STDOUT`是 bash 脚本的默认输出，而`STDERR`可以通过附加写入`>&2`到 bash 命令的末尾。例如，`echo 'hello'>&2`。生命周期配置的日志将发布到Amazon通过 CloudWatch 账户。这些日志可以在`/aws/sagemaker/studio`从日志流AmazonCloudWatch 控制台。

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. Select`Logs`从左侧。在下拉菜单中，选择`Log Groups`.
3. 在存储库的`Log Groups`屏幕，搜索`aws/sagemaker/studio`. 选择 日志组。
4. 在存储库的`aws/sagemaker/studio Log Group`屏幕上，导航到`Log Streams`选项卡。
5. 要查找特定应用程序的日志，请搜索`Log Streams`使用以下格式：

   ```
   <DomainId>/<UserProfileName>/<AppType>/<AppName>
   ```

   例如，要查找域的生命周期配置日志`d-m85lcu8vbqmz`、 UserProfile `i-sonic-js`，AppType`JupyterServer`和 AppName`test-lcc-echo`中，使用以下搜索字符串：

   ```
   d-m85lcu8vbqmz/i-sonic-js/JupyterServer/test-lcc-echo
   ```

6. 选择附加到的日志流`LifecycleConfigOnStart`以查看脚本执行日志。

JupyterServer 应用程序失败

如果您的 JupyterServer 由于附加的生命周期配置出现问题，应用程序崩溃，Studio 在 Studio 启动屏幕上显示以下错误消息。

```
Failed to create SageMaker Studio due to start-up script failure
```

单击`View script logs`链接以查看 CloudWatch 的日志 JupyterServer 应用程序。

如果在`DefaultResourceSpec`在你的 Studio 域或 UserProfile 中，即使在重新启动 Studio 之后，Studio 仍会继续使用生命周期配置。

要解决此错误，请按照中的步骤进行操作设置默认周期配置 (p. 122)从中删除生命周期配置脚本`DefaultResourceSpec`或者使用Amazon CLI. 然后启动新的 JupyterServer 应用程序。

KernelGateway 应用程序故

如果您的 KernelGateway 由于附加的生命周期配置存在问题，应用程序崩溃，Studio 会在您的 Studio Notebook 中显示错误消息。

单击`View script logs`链接以查看 CloudWatch 的日志 KernelGateway 应用程序。

在这种情况下，启动新的 Studio 笔记本电脑时，您的生命周期配置是在 Studio Launcher 中指定的。

要解决此错误，请使用 Studio 启动器选择不同的生命周期配置或选择`No script`.

> Note
>
> 一个默认 KernelGateway 在中指定的生命周期`DefaultResourceSpec`适用于所有 KernelGateway 除非用户从 Studio 启动器中显示的列表中选择了不同的脚本，否则 Studio Domain

中的映像。默认脚本也会运行`No Script`是由用户选择的。有关如何选择脚本的更多信息，请参阅第 3 步：启动新应用程序时选择生命周期配置 (p. 118).

周期 Config 超时

生命周期配置超时限制为 5 分钟。如果生命周期配置脚本运行时间超过 5 分钟，Studio 会抛出错误。

要解决此错误，请确保生命周期配置脚本在 5 分钟内完成。

更新和删除生命周期配置

生命周期配置脚本在创建后不可更改。要更新脚本，必须创建一个新的生命周期配置脚本并使用更新域和更新用户配置文件用于将生命周期配置脚本附加到相应的域或 UserProfile 的 API。有关更多信息，请参阅 创建和关联生命周期配置 (p. 116)。

要删除现有的生命周期配置，请使用删除 Studio LifeCycle ConfigAPI。要成功删除生命周期配置，没有正在运行的应用程序可以使用它。

在亚马逊执行常见任务 SageMaker 工作室

以下部分介绍如何在 Amazon 中执行常见任务： SageMaker Studio。有关 Studio 界面的概述，请参阅 亚马逊 SageMaker Studio UI 概述 (p. 77)。

主题

- 将文件上传到 SageMaker 工作室 (p. 125)
- 在中克隆 Git 存储库 SageMaker 工作室 (p. 125)
- 在中停止训练 Job SageMaker 工作室 (p. 126)
- 使用 TensorBoard 在 Amazon SageMaker 工作室 (p. 126)
- 在中管理 EFS 存储卷 SageMaker 工作室 (p. 128)
- 提供反馈 SageMaker 工作室 (p. 128)
- 关闭并更新 SageMaker 工作室和工作室应用 (p. 128)

将文件上传到 SageMaker 工作室

当你加入亚马逊时 SageMaker Studio 中，系统会在为您的团队创建的 Amazon Elastic File System (Amazon EFS) 卷中为您创建主目录。Studio 只能打开已上传到目录的文件。Studio 文件浏览器映像到您的主目录。

要将文件上传到您的主目录

1. 在左侧边栏中，选择 File Browser (文件浏览器) 图标 ( )。
2. 在文件浏览器中，选择上传文件图标 ( )。
3. 选择要上传的文件，然后选择打开.
4. 双击某个文件以在 Studio 中的新选项卡中打开该文件。

在中克隆 Git 存储库 SageMaker 工作室

亚马逊 SageMaker Studio 只能连接到本地存储库。在此示例中，您克隆aws/amazon-sagemaker-examples存储库（回购）。

克隆存储库

1. 在左侧边栏中，选择 File Browser (文件浏览器) 图标 ( )。
2. 选择根文件夹或要克隆回购的文件夹。
3. 在左侧边栏中，选择 Git 图标 ( )。
4. 选择克隆存储库.
5. 输入 URI SageMaker 存储库示例`https://github.com/aws/amazon-sagemaker-examples.git`.
6. 选择 CLONE (克隆)。
7. 如果回购需要凭据，系统会提示您输入用户名和个人访问令牌。
8. 请等待下载完成。在回购被克隆之后，文件浏览器打开以显示克隆的回购库。
9. 双击存储库将其打开。
10. 选择饭桶图标来查看 Git 用户界面，该界面现在跟踪示例存储库。
11. 要跟踪不同的回购，请在文件浏览器中打开存储库，然后选择饭桶图标。

在中停止训练 Job SageMaker 工作室

您可以停止在 Amazon 的训练任务 SageMaker Studio UI。当您停止训练作业时，其状态将更改为`Stopping`届时结算停止。算法可以延迟终止，以便保存模型构件，之后作业状态更改为`Stopped`. 有关更多信息，请参阅 。stop_训练_job中的方法Amazon SDK for Python (Boto3).

停止训练作业

1. 按照此页面上的查看和比较实验、试验和试验组件 (p. 1763)过程进行操作，直至打开 Describe Trial Component (描述试验组件) 选项卡。
2. 在选项卡的右上方，选择 Stop training job (停止训练作业)。该选项卡左上方的 Status (状态) 将更改为 Stopped (已停止)。
3. 要查看训练时间和计费时间，请选择Amazon设置.

使用 TensorBoard 在 Amazon SageMaker 工作室

以下文档概述如何安装和运行。 TensorBoard 在 Amazon SageMaker Studio。

先决条件

本教程要求拥有亚马逊 SageMaker Studio 域。

设置 `TensorBoardCallback`

1. 启动 Studio。
2. 在 Amazon 中 SageMaker 下 Studio 启动器`Notebooks and compute resources`，选择`TensorFlow 2.3 Python 3.7(optimized for CPU)`Studio 映像。
3. 启动笔记本。
4. 导入所需的包裹。

```
import os
import datetime
import tensorflow as tf
```

5. 创建你的 Keras 模型。

```
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

def create_model():
  return tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
  ])
```

6. 为您的创建一个目录 TensorBoard 圆木

```
LOG_DIR = os.path.join(os.getcwd(), "logs/fit/" + datetime.datetime.now().strftime("%Y
%m%d-%H%M%S"))
```

7. 使用 TensorBoard 进行训练。

```
model = create_model()
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=LOG_DIR,
 histogram_freq=1)

model.fit(x=x_train,
          y=y_train,
          epochs=5,
          validation_data=(x_test, y_test),
          callbacks=[tensorboard_callback])
```

8. 生成的 EFS 路径 TensorBoard 日志。您可以使用此路径从终端设置日志。

```
EFS_PATH_LOG_DIR = "/".join(LOG_DIR.strip("/").split('/')[1:-1])
print (EFS_PATH_LOG_DIR)
```

安装 TensorBoard

1. 点击Amazon SageMaker StudioStudio 左上角的按钮以打开亚马逊 SageMaker Studio 启动程序。必须从根目录中打开此启动器。
2. 在启动程序下Utilities and files，单击System terminal.
3. 从终端运行以下命令。CopyEFS_PATH_LOG_DIR从 Jupyter 笔记本中。您必须从/home/sagemaker-user根目录。

```
pip install tensorboard
tensorboard --logdir <EFS_PATH_LOG_DIR>
```

启动 TensorBoard

1. 要启动 TensorBoard，请复制你的 Studio URL 并替换lab?和proxy/6006/如下所示。您必须包含尾随/character.

```
https://<YOUR_URL>.studio.region.sagemaker.aws/jupyter/default/proxy/6006/
```

2. 导航到 URL 以检查结果。

在中管理 EFS 存储卷 SageMaker 工作室

团队中的用户首次加入亚马逊 SageMaker Amazon Studio SageMaker 为团队创建 Amazon Elastic File System (Amazon EFS) 卷。在卷中为作为团队成员加入 Studio 的每个用户创建一个主目录。笔记本文件和数据文件存储在这些目录中。用户无权访问其他团队成员的主目录。

> **Important**
>
> 请勿删除 Amazon EFS 卷。如果您删除它，域将无法正常工作，所有用户都将失去其工作。

查找 Amazon EFS 卷

1. 从控制面板，在域，查找域名 ID. ID 的格式如下：`d-xxxxxxxxxxxx`。
2. 传递`Domain ID`，因为`DomainId`，转到describe_域名方法。
3. 在从回复`describe_domain`，记下的值`HomeEfsFileSystemId`键。这是 Amazon EFS 文件系统 ID。
4. 打开Amazon EFS 控制台. 确保Amazon地区与 Studio 使用的区域相同。
5. UDER文件系统中，请从上一步中选择文件系统 ID。
6. 要验证您选择了正确的文件系统，请选择标签标题。对应于`ManagedByAmazonSageMakerResource`密钥应与`Studio ID`.

有关如何访问 Amazon EFS 卷的信息，请参阅在 Amazon EFS 中使用文件系统.

要删除 Amazon EFS 卷，请参阅删除 Amazon EFS 文件系统.

提供反馈 SageMaker 工作室

亚马逊 SageMaker 认真对待你的反馈。我们鼓励你提供反馈。

提供反馈

1. 在右上部分 SageMaker 选择 Studio反馈.
2. 选择笑脸表情符号，让我们了解您对多满意 SageMaker 并添加您想与我们分享的任何反馈。
3. 决定是否与我们共享您的身份，然后选择 Submit (提交)。

关闭并更新 SageMaker 工作室和工作室应用

以下主题介绍如何关闭和更新。 SageMaker 工作室和 Studio 应用程序。

亚马逊 SageMaker 不更新亚马逊 SageMaker Studio 应用程序在使用时。

Studio 提供了一个通知图标 ( )。此通知图标显示未读通知的数量。要阅读通知，请选择图标。

Studio 提供了两种类型的通知：

- 升级 — 当 Studio 或其中一个 Studio 应用程序发布了新版本时显示。要更新，请参阅关闭并更新 SageMaker 工作室和工作室应用 (p. 128).

- 信息 — 显示新功能和其他信息。

要重置通知图标，您必须在每个通知中选择链接。已读通知可能仍会显示在图标中。这并不表示更新 Studio 和 Studio 应用程序后仍需要更新。

了解如何更新亚马逊 SageMaker DATA Wrangler请参阅关闭并更新 Studio 应用 (p. 130).

要确保您拥有最新的软件更新，请更新 Amazon SageMaker Studio 和 Studio 应用程序使用以下主题概述的方法。

主题
- 关闭并更新 SageMaker 工作室 (p. 129)
- 关闭并更新 Studio 应用 (p. 130)

关闭并更新 SageMaker 工作室

更新 Amazon SageMaker Studio 到最新版本时，您必须关闭 JupyterServer 应用程序。你可以关闭 JupyterServer 应用程序来自 SageMaker 控制台或从 Studio 中。在 JupyterServer 应用程序已关闭，您必须通过 SageMaker 创建一个新版本的控制台 JupyterServer 应用程序。

在该过程中，任何未保存的笔记本信息将会丢失。Amazon EFS 卷中的用户数据不会受到影响。

Studio 中的一些服务（如 Data Wrangler）在自己的应用程序上运行。要更新这些服务，您必须删除该服务的应用程序。要了解更多信息，请参阅 关闭并更新 Studio 应用 (p. 130)。

> Note
>
> 一个 JupyterServer 应用程序与单个 Studio 用户关联。当你为一个用户更新应用程序时，它不会影响其他用户。

以下主题显示了如何更新 JupyterServer 来自 SageMaker 控制台或来自 Studio 内部。

更新 JupyterServer 应用程序来自 SageMaker 控制台

1. 导航到https://console.aws.amazon.com/sagemaker/.
2. 选择控制面板.
3. UDER用户，选择您的用户名。
4. UDER应用程序，在显示的行中JupyterServer，选择删除 App.
5. 选择是，删除应用.
6. 类型**delete**在确认框中。
7. 请选择 Delete（删除）。
8. 删除应用程序后，启动一个新的 Studio 应用程序以获取最新版本。

更新 JupyterServer 来自 Studio 内部的应用

1. 启动 Studio。
2. 在顶部菜单上，选择文件，然后选择关闭。
3. 请选择以下任一选项：
   - 关闭服务器— 关闭 JupyterServer 应用程序。终端会话、内核会话、 SageMaker 映像和实例不会关闭。这些资源继续产生费用。
   - 全部关闭— 关闭所有应用程序、终端会话、内核会话、 SageMaker 映像和实例。这些资源不再产生费用。
4. 关闭 窗口。

5. 删除应用程序后，启动一个新的 Studio 应用程序以使用最新版本。

关闭并更新 Studio 应用

更新亚马逊 SageMaker Studio 应用程序到最新版本，你必须先关闭相应的 KernelGateway 应用程序来自 SageMaker 控制台。KernelGateway 应用程序关闭后，您必须通过重新打开它 SageMaker Studio 通过运行新内核。内核会自动更新。在该过程中，任何未保存的笔记本信息将会丢失。Amazon EFS 卷中的用户数据不会受到影响。

> Note
>
> 一个 KernelGateway 应用程序与单个 Studio 用户关联。当你为一个用户更新应用程序时，它不会影响其他用户。

更新 KernelGateway 应用

1. 导航到https://console.aws.amazon.com/sagemaker/.
2. 选择控制面板.
3. UDER用户，选择您的用户名。
4. UDER应用程序，在显示App name，选择删除 App.

   要更新 Data Wrangler，请删除以开头的应用程序sagemaker-数据响起.
5. 选择是，删除应用.
6. 类型**delete**在确认框中。
7. 请选择 Delete（删除）。
8. 删除应用程序后，从 Studio 中启动新内核以使用最新版本。

亚马逊 SageMaker Studio 定价

当你团队的第一名成员加入亚马逊时 SageMaker Amazon Studio SageMaker 为团队创建 Amazon Elastic File System (Amazon EFS) 卷。在 SageMaker Studio 时的控制面板状态显示为`Ready`，Amazon EFS 卷已创建。

当此成员或团队的任何成员打开 Studio 时，将在卷中为该成员创建主目录。此目录将产生存储费。随后，存储在会员主目录中的笔记本和数据文件将产生额外的存储费用。有关 Amazon EFS 的定价信息，请参阅。Amazon EFS 定价.

在 Studio 中运行其他操作（例如，运行笔记本、运行训练作业和托管模型）时，将产生额外费用。

有关使用 Studio 笔记本的相关费用的信息，请参阅 使用计量 (p. 98)。

有关计费和定价示例的信息，请参阅亚马逊 SageMaker 定价.

Amazon 故障排除 SageMaker 工作室

以下是您在使用 Amazon 时可能会遇到的常见错误。 SageMaker Studio。每个错误之后都是错误的解决方案。

- SageMaker Studio 的核心功能不可用。

  如果你在打开 Studio 时收到此错误消息，可能是由于 Python 软件包版本冲突造成的。如果您在笔记本或终端中使用以下命令来安装版本冲突的 Python 软件包，则会发生这种情况： SageMaker 软件包依赖项。

```
!pip install
```

```
pip install --user
```

要解决此问题，请完成以下步骤：

1. 卸载最近安装的 Python 软件包。如果你不确定要卸载哪个软件包，请使用左下角的反馈按钮联系 Amazon Web Services Management Console.
2. 重新启动 Studio:
   a. 从中关闭 StudioFile (文件)菜单。
   b. 等 1 分钟。
   c. 重新打开 Studio，方法是刷新页面或从Amazon Web Services Management Console.

如果你卸载了导致冲突的软件包，问题应该得到解决。要在不再造成此问题的情况下安装包，请使用`%pip install`没有`--user`标志。

如果问题仍然存在，请创建新的用户配置文件并使用该用户配置文件设置环境。

如果这些解决方案不能解决问题，请使用左下角的反馈按钮联系Amazon Web Services Management Console.

Amazon SageMaker 上的 rStudio

RStudio 是 R 的集成开发环境，具有控制台、支持直接代码执行的语法突出显示编辑器以及绘图、历史记录、调试和工作空间管理工具。 Amazon SageMaker 支持 rStudio 作为与 Amazon SageMaker 域集成的完全托管的集成开发环境 (IDE)。

rStudio 允许客户使用 R 环境创建数据科学见解。借助 rStudio 集成，您可以在域中启动 rStudio 环境，以便在 SageMaker 资源上运行 rStudio 工作流程。有关 RStudio 的更多信息，请参阅rStudio 网站.

主题

- rStudio 组件 (p. 132)
- 与 rStudio 工作台的区别 (p. 132)
- Amazon SageMaker 上的 (p. 132)
- 在 Amazon SageMaker 上使用 RStudio (p. 143)

SageMaker 通过创建 rStudioServerPro 应用程序集成了 rStudio。

SageMaker 上的 rStudio 支持以下内容。

- R 开发人员将 rStudio IDE 界面与 R 生态系统中的流行开发人员工具结合使用。用户可以启动新的 RStudio 会话、编写 R 代码、从 rStudio 软件包管理器安装依赖项以及使用 rStudio Connect 发布闪亮应用程序。
- R 开发人员可以快速扩展底层计算资源以运行大规模数据处理和统计分析。
- 平台管理员可以通过以下方式为其数据科学团队设置用户身份、授权、网络、存储和安全性Amazon IAM Identity Center (successor to Amazon Single Sign-On)和Amazon Identity and Access Management 集成。这包括连接到私有 Amazon Virtual Private Cloud (Amazon VPC) 资源和使用的免费互联网模式 Amazon PrivateLink.
- 集成Amazon License Manager.

有关启用 RStudio 的域的入门步骤的信息，请参阅。注册到 Amazon SageMaker 域 (p. 32).

有关的信息Amazon Web Services 区域SageMaker 上的 rStudio 受支持，请参阅支持的区域和配额 (p. 28).

rStudio 组件

- rStudioServerPRO：rStudioServerPro 应用程序是多用户应用程序，是域中所有用户配置文件之间的共享资源。在域中创建 rStudio 应用程序后，管理员可以向域中的用户授予权限。
- rStudio 用户：rStudio 用户是域中有权使用 rStudio 许可证的用户。
- RStudio 管理员：Amazon SageMaker 上的 rStudio 管理员可以访问 rStudio 管理控制面板。Amazon SageMaker 上的 rStudio 管理员与 “股票” rStudio Workbench 管理员不同，因为他们没有对运行 rStudioServerPro 应用程序的实例的根访问权限，也无法修改 rStudio 配置文件。
- RStudio 服务器：rStudio 服务器实例负责向所有授权用户提供 rStudio UI。此实例在 Amazon SageMaker 实例上启动。
- 会话：RSession 是 Amazon SageMaker 实例上运行的 RStudio IDE 的基于浏览器的接口。用户可以通过 rSession 创建 RStudio 项目并与之交互。
- rSessionGateway：ressionGateway 应用程序用于支持 rSession。
- RStudio 管理控制面板：此仪表板提供了有关 Amazon SageMaker 域中的 rStudio 用户及其会话的信息。只有拥有 rStudio 管理员授权的用户才能访问此仪表板。

与 rStudio 工作台的区别

Amazon SageMaker 上的 rStudio 与rStudio Workbench.

- 在 SageMaker 上使用 RStudio 时，用户无权访问 RStudio 配置文件。Amazon SageMaker 管理配置文件并设置默认值。创建启用了 RSTUDIO 的 Amazon SageMaker 域时，您可以修改 rStudio Connect 和 rStudio 软件包管理器 URL。
- 在 Amazon SageMaker 上使用 rStudio 时，目前不支持项目共享、实时协作和作 Job 启动器。
- 在 SageMaker 上使用 rStudio 时，RStudio IDE 将在 Amazon SageMaker 实例上运行，以获取按需容器化计算资源。
- SageMaker 上的 rStudio 仅支持 rStudio IDE，不支持 rStudio 工作台安装支持的其他 IDE。

Amazon SageMaker 上的

以下主题提供了有关在 Amazon SageMaker 上管理 rStudio 的信息。这包括有关 RStudio 环境配置、用户会话和必要资源的信息。有关如何在 SageMaker 上的使用 RStudio 的信息，请参阅。在 Amazon SageMaker 上使用 RStudio (p. 143).

有关在启用 RStudio 的情况下创建 Amazon SageMaker 域的信息，请参阅。注册到 Amazon SageMaker 域 (p. 32).

有关的信息AmazonSageMaker 上的 RStudio 支持的区域，请参阅。支持的区域和配额 (p. 28).

主题

- rStudio 许可证 (p. 133)
- 联网和存储 (p. 133)
- RStudioServerPro 实例类型 (p. 134)
- rStudio Connect 网址 (p. 134)
- rStudio Package 管理器 (p. 135)
- 使用 RStudio 创建 Amazon SageMaker 域Amazon CLI (p. 135)
- 管理用户 (p. 139)
- rStudio 管理控制面板 (p. 141)
- 管理账单和成本 (p. 142)

- 诊断问题并获得支持 (p. 142)

rStudio 许可证

Amazon SageMaker 上的 rStudio 是付费产品，要求每个用户都获得适当的许可。Amazon SageMaker 不销售或提供 rStudio 许可证。Amazon SageMaker 上的 rStudio 需要与 rStudio 单独的产品许可证。对于 rStudio Workbench 企业版的客户，许可证发放不额外费用。

要将 RStudio 许可与 Amazon SageMaker 一起使用，您必须首先拥有注册的有效 rStudio 许可证Amazon License Manager. 有关将许可证注册的更多信息Amazon License Manager，请参阅卖家在Amazon License Manager. 以下主题介绍了如何获取和验证 RStudio 许可证。

获取 rStudio 许可证

1. 如果您没有 RStudio 许可证，您可以在此处购买一个rStudio 定价或者通过联系sales@rstudio.com. 当购买或更新 RStudio 许可证时，必须提供Amazon将托管您的 Amazon SageMaker 域名的账户。

   如果你有现有的 rStudio 许可证，请联系你的 rStudio 销售代表或sales@rstudio.com将 Amazon SageMaker 上的 rStudio 添加到现有的 rStudio Workbench 企业版许可证中，或者转换您的 rStudio Workbench 标准许可证。rStudio 销售代表将向您发送相应的电子订单。
2. rStudio 将 rStudio 工作台许可证授予您的Amazon账户通过Amazon License Manager在美国东部 (弗吉尼亚北部) 区域。尽管 RStudio 许可证是在美国东部 (弗吉尼亚北部) 区域授予的，但您的许可证可可以在任何一个区域中使用AmazonAmazon SageMaker 上的 rStudio 受支持的区域。您可以预计许可证授予流程将在共享您的Amazon使用 rStudio 的账户 ID。
3. 授予此许可证后，您会收到 RStudio Sales 代表发送的电子邮件，其中包含接受许可证授予的说明。

验证您的 rStudio 许可证以与 Amazon SageMaker 一起使用

1. 登录Amazon License Manager控制台位于 Amazon SageMaker 域位于相同区域中。如果您正在使用Amazon License Manager首次，您需要授予使用许可Amazon License Manager.
2. Select创建客户托管许可证.
3. SelectI grant Amazon License Manager the required permissions然后选择授予权限.
4. 导航到已授予许可在左侧面板上。
5. 使用选择许可证授予RSW-SageMaker作为Product name然后选择查看.
6. 在许可证详细信息页面中，选择接受并激活许可证.

获取有关 RStudio 许可证使用量的信息

要分享有多少用户在使用 RStudio 许可证，您必须从 Amazon SageMaker 获得有关此使用情况的信息。您可以导出使用情况信息，然后将其报告给 RStudio PBC。您可以从中获取有关许可证使用量的信息Amazon License Manager或者 rStudio 管理仪表板。以下是如何收集和导出此信息的演示。

Amazon License Manager

Amazon License Manager跟踪授权许可证的使用情况，并可以生成该使用情况的报告。有关如何生成的信息Amazon License Manager请参阅使用量报告许可证管理器中的许可证.

rStudio 管理控制面板

您可以按照中的步骤使用 rStudio 管理仪表板查看许可证上的用户数量rStudio 管理控制面板 (p. 141).

联网和存储

以下主题介绍了 RStudio 实例的网络访问和数据存储注意事项。有关使用 Amazon SageMaker 时的网络访问和数据存储的一般信息，请参阅Amazon SageMaker 中的数据保护 (p. 2686).

加密

Amazon SageMaker 上的 rStudio 支持静态加密。

在仅限 vPC 模式下使用 rStudio

Amazon SageMaker 中的 rStudio 支持Amazon PrivateLink集成。通过此集成，您可以在 SageMaker 上以仅 VPC 模式使用 RStudio，而无需直接访问互联网。当您在仅限 VPC 模式下使用 RStudio 时，您的安全组将由服务自动管理。这包括 rServer 和 rSessions 之间的连接。

在仅 VPC 模式下使用 RStudio 需要以下条件。有关选择 VPC 的更多信息，请参阅选择 VPC。 (p. 35).

- 可以访问互联网以致电 Amazon SageMaker 和许可证管理器的私有子网，或者可以访问 Amazon SageMaker 和许可证管理器的 Amazon Virtual Private Cloud (Amazon VPC) 终端节点。
- 域设置中用于域设置中的域的安全组 ID。这必须允许所有出站访问。
- 与 Amazon VPC 终端节点配合使用的安全组 ID。此安全组必须允许来自域安全组 ID 的入站流量。
- 的 Amazon VPC 终端节点`sagemaker.api`和Amazon License Manager. 它必须与私有子网位于同一 Amazon VPC 中。

RStudioServerPro 实例类型

在决定将哪种 Amazon EC2 实例类型用于您的 rStudioServerPro 应用程序时，需要考虑的主要因素是带宽。带宽很重要，因为 rStudioServerPro 实例负责向所有用户提供 rStudio UI。这包括 UI 繁重的工作流程，例如生成数字、动画和显示许多数据行。因此，根据所有用户的工作负载，可能会出现一些 UI 性能下降。有以下实例类型可供使用：有关这些实例的定价信息，请参阅Amazon SageMaker 定价.

- `ml.t3.medium`：建议使用此实例类型适用于 UI 使用率低的域，且可以免费使用。
- `ml.c5.4xlarge`：建议使用适度 UI 的域使用此实例类型。
- `ml.c5.9xlarge`：建议使用大量用户界面的域使用此实例类型。

更改 rStudio 实例类型

要更改 rStudioServerPro 的实例类型，请将新的实例类型作为调用的一部分传递给`update-domain`CLI 命令。然后你需要使用删除现有的 RSTUDIOServerPRO 应用程序`delete-app`CLI 命令，然后使用`create-app`CLI 命令。

rStudio Connect 网址

rStudio Connect 是 Shiny 应用程序、R Markdown 报告、仪表板、绘图等的发布平台。通过使托管内容变得简单且可扩展，RStudio Connect 可以轻松地显示机器学习和数据科学见解。如果您有 RStudio Connect 服务器，则可以将服务器设置为发布应用程序的默认位置。 有关 RStudio Connect 的更多信息，请参阅RStudio Connect.

当你登机到亚马逊上的 rStudio 时 SageMaker 域，未创建 rStudio Connect 服务器。您可以在 Amazon EC2 实例上创建 RStudio Connect 服务器以使用 “与亚马逊 Connect” SageMaker 域。有关如何设置 RStudio Connect 服务器的信息，请参阅在 Amazon SageMaker 上的 rStudio 中进行机器学习开发的 RStudio Connect 和 Package 管理器.

添加 rStudio Connect 网址

如果您有 rStudio Connect URL，则可以更新默认 URL，以便 RStudio 用户可以向其发布。

1. 导航到控制面板.
2. 在域下，选择编辑设置. 这将打开一个新页面。
3. 在新页面中，选择rStudio 设置在左侧。
4. 在下rStudio Connect 网址中，输入要添加的 rStudio Connect URL。

5. 选择 Submit (提交)。

CLI

创建亚马逊时，您可以设置默认的 rStudio Connect URL SageMaker 域。从中更新 rStudio Connect URL 的唯一方法Amazon CLI要删除域并使用更新的 RStudio Connect URL 创建一个新域。

rStudio Package 管理器

rStudio Package 管理器是一个存储库管理服务器，用于在整个组织中组织和集中管理软件包。有关 RStudio Package 管理器的更多信息，请参阅rStudio Package 管理器. 如果您未提供自己的 Package 管理器 URL，则亚马逊 SageMaker 当您加载 rStudio 时，域将使用默认的 Package 管理器存储库，按照中的步骤注册到 Amazon SageMaker 域 (p. 32). 有关更多信息，请参阅 。在 Amazon SageMaker 上的 rStudio 中进行机器学习开发的 RStudio Connect 和 Package 管理器.

更新 Package 管理器 URL

您可以按如下方式更新用于启用了 RSTUDIO 的域的 Package 管理器 URL。

1. 导航到控制面板.
2. 在域名下，选择编辑设置. 这将打开新页面。
3. 在新页面中，选择rStudio 设置在左侧。
4. 下rStudio Package 管理器中，输入您的 rStudio Package 管理器 URL。
5. 选择 Submit (提交)。

CLI

从中更新 Package 管理器 URL 的唯一方法Amazon CLI是删除您的域并使用更新的 Package 管理器 URL 创建一个新域名。

使用 RStudio 创建 Amazon SageMaker 域Amazon CLI

以下主题演示了如何在启用了 rStudio 的情况下使用Amazon CLI. 要使用Amazon Web Services Management Console请参阅,注册到 Amazon SageMaker 域 (p. 32).

先决条件

- 安装和配置Amazon CLI版本 2
- 配置Amazon CLI使用 IAM 凭证

创建 `DomainExecution` 角色

要启动 rStudio 应用程序，您必须提供`DomainExecution`角色。此角色用于确定是否需要在创建 Amazon SageMaker 域时启动 rStudio。Amazon SageMaker 还使用此角色访问 rStudio 许可证并推送 rStudio 日志。

> **Note**
>
> 这些区域有：`DomainExecution`角色至少应该有Amazon License Manager访问 rStudio 许可证的权限以及在账户中推送日志的 CloudWatch 权限。

以下过程演示如何创建`DomainExecution`角色与Amazon CLI.

1. 使用以下内容创建名为 `assume-role-policy.json` 的文件。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "sts:AssumeRole",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "sagemaker.amazonaws.com"
                ]
            }
        }
    ]
}
```

2. 创建DomainExecution角色。<REGION>应为Amazon要在其中启动域名的区域。

```
aws iam create-role --region <REGION> --role-name DomainExecution --assume-role-policy-
document file://assume-role-policy.json
```

3. 使用以下内容创建名为 domain-setting-policy.json 的文件。此政策允许 rStudioServerPRO 应用程序访问必要的资源，并允许 Amazon SageMaker 在现有的 rStudioServerPRO 应用程序位于Deleted要么Failed状态。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "license-manager:ExtendLicenseConsumption",
                "license-manager:ListReceivedLicenses",
                "license-manager:GetLicense",
                "license-manager:CheckoutLicense",
                "license-manager:CheckInLicense",
                "logs:CreateLogDelivery",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DeleteLogDelivery",
                "logs:Describe*",
                "logs:GetLogDelivery",
                "logs:GetLogEvents",
                "logs:ListLogDeliveries",
                "logs:PutLogEvents",
                "logs:PutResourcePolicy",
                "logs:UpdateLogDelivery",
                "sagemaker:CreateApp"
            ],
            "Resource": "*"
        }
    ]
}
```

4. 创建附加到的域设置策略DomainExecution角色。请注意PolicyArn在响应中，您将需要在接下来的步骤中输入该 ARN。

```
aws iam create-policy --region <REGION> --policy-name domain-setting-policy --policy-
document file://domain-setting-policy.json
```

5. Attachdomain-setting-policy到DomainExecution角色。使用PolicyArn在上一步中返回。

```
aws iam attach-role-policy --role-name DomainExecution --policy-arn <POLICY_ARN>
```

使用 rStudio 应用程序创建 Amazon SageMaker 域

当 Amazon SageMaker 用`create-domain`使用 CLI 命令`RStudioServerProDomainSettings`已指定参数。启动 rStudioServerPro 应用程序时，Amazon SageMaker 会检查账户中是否有效的 rStudio 许可证，如果找不到许可证，则域名创建失败。

Amazon SageMaker 域的创建取决于身份验证方法和网络类型。必须同时使用这些选项，同时选择一种身份验证方法和一种网络连接类型。有关创建新域的要求的更多信息，请参阅CreateDomain.

支持以下身份验证方法。

- `IAM Auth`
- `SSO Auth`

支持以下网络连接类型：

- `PublicInternet`
- `VPCOnly`

验证方法

IAM 身份验证模式

下面介绍了如何在启用了 rStudio 的情况下创建 Amazon SageMaker 域名和`IAM Auth`网络类型。有关 的更多信息Amazon Identity and Access Management请参阅,什么是 IAM？.

- `DomainExecutionRoleArn`应为上一步中创建的 ARN。
- `ExecutionRole`是在 Amazon SageMaker 域中分配给用户的角色的 ARN。
- `vpc-id`应为 Amazon Virtual Private Cloud 的 ID。`subnet-ids`应为子网 ID 的空格分隔的列表。有关的信息`vpc-id`和`subnet-ids`请参阅,VPC 和子网.
- `RStudioPackageManagerUrl`和`RStudioConnectUrl`是可选的，应分别设置为 rStudio 软件包管理器和 rStudio Connect 服务器的 URL。
- `app-network-access-type`应为`PublicInternetOnly`要么`VPCOnly`.

```
aws sagemaker create-domain --region <REGION> --domain-name <DOMAIN_NAME> \
    --auth-mode IAM \
    --default-user-settings ExecutionRole=<DEFAULT_USER_EXECUTIONROLE> \
    --domain-settings
 RStudioServerProDomainSettings={RStudioPackageManagerUrl=<<PACKAGE_MANAGER_URL>,RStudioConnectUrl=<<CO
 \
    --vpc-id <VPC_ID> \
    --subnet-ids <SUBNET_IDS> \
    --app-network-access-type <NETWORK_ACCESS_TYPE>
```

IAM Identity Center身份验证模式

下面介绍了如何在启用了 rStudio 的情况下创建 Amazon SageMaker 域名和`SSO Auth`网络类型。Amazon IAM Identity Center (successor to Amazon Single Sign-On)必须为启动域的区域启用。有关 IAM Identity Center 的更多信息，请参阅什么是 Amazon IAM Identity Center (successor to Amazon Single Sign-On)？。

- `DomainExecutionRoleArn`应为上一步中创建的 ARN。
- `ExecutionRole`是在 Amazon SageMaker 域中分配给用户的角色的 ARN。
- `vpc-id`应为 Amazon Virtual Private Cloud 的 ID。`subnet-ids`应为子网 ID 的空格分隔的列表。有关的信息`vpc-id`和`subnet-ids`请参阅,VPC 和子网.

- `RStudioPackageManagerUrl`和`RStudioConnectUrl`是可选的，应分别设置为 rStudio 软件包管理器和 rStudio Connect 服务器的 URL。
- `app-network-access-type`应为`PublicInternetOnly`要么`VPCOnly`.

```
aws sagemaker create-domain --region <REGION> --domain-name <DOMAIN_NAME> \
    --auth-mode SSO \
    --default-user-settings ExecutionRole=<DEFAULT_USER_EXECUTIONROLE> \
    --domain-settings
 RStudioServerProDomainSettings={RStudioPackageManagerUrl=<<PACKAGE_MANAGER_URL>,RStudioConnectUrl=<<CO
 \
    --vpc-id <VPC_ID> \
    --subnet-ids <SUBNET_IDS> \
    --app-network-access-type <NETWORK_ACCESS_TYPE>
```

连接类型

公共互联网/直接互联网络类型

以下内容介绍了如何创建启用 rStudio 的 Amazon SageMaker 域，以及`PublicInternet`网络类型。

- `DomainExecutionRoleArn`应为上一步中创建的 ARN。
- `ExecutionRole`是在 Amazon SageMaker 域中分配给用户的角色的 ARN。
- `vpc-id`应为 Amazon Virtual Private Cloud 的 ID。`subnet-ids`应为子网 ID 的空格分隔的列表。有关的信息`vpc-id`和`subnet-ids`请参阅,VPC 和子网.
- `RStudioPackageManagerUrl`和`RStudioConnectUrl`是可选的，应分别设置为 rStudio 软件包管理器和 rStudio Connect 服务器的 URL。
- `auth-mode`应为`SSO`要么`IAM`.

```
aws sagemaker create-domain --region <REGION> --domain-name <DOMAIN_NAME> \
    --auth-mode <AUTH_MODE> \
    --default-user-settings ExecutionRole=<DEFAULT_USER_EXECUTIONROLE> \
    --domain-settings
 RStudioServerProDomainSettings={RStudioPackageManagerUrl=<<PACKAGE_MANAGER_URL>,RStudioConnectUrl=<<CO
 \
    --vpc-id <VPC_ID> \
    --subnet-ids <SUBNET_IDS> \
    --app-network-access-type PublicInternetOnly
```

vPconly 模式

下面介绍了如何启动启用了 rStudio 的 Amazon SageMaker 域名和`VPCOnly`网络类型。有关使用`VPCOnly`网络访问类型，请参阅Connect (连接) SageMaker VPC 中的 Studio 笔记本电脑到外部资源 (p. 2781).

- `DomainExecutionRoleArn`应为上一步中创建的 ARN。
- `ExecutionRole`是在 Amazon SageMaker 域中分配给用户的角色的 ARN。
- `vpc-id`应为 Amazon Virtual Private Cloud 的 ID。`subnet-ids`应为子网 ID 的空格分隔的列表。您的私有子网必须能够访问互联网才能拨打 Amazon SageMaker，以及Amazon License Manager或者为 Amazon SageMaker 和拥有亚马逊 VPC 终端节点Amazon License Manager. 有关 Amazon VPC 终端节点的信息，请参阅接口 Amazon VPC 终端节点有关的信息`vpc-id`和`subnet-ids`请参阅,VPC 和子网.
- `SecurityGroups`必须允许对 Amazon SageMaker 进行出站访问权限Amazon License Manager终端节点。
- `auth-mode`应为`SSO`要么`IAM`.

Note

使用 Amazon Virtual Private Cloud 终端节点时，附加到您的 Amazon Virtual Private Cloud 终端节点的安全组必须允许来自您作为`domain-setting`的参数`create-domain`CLI 调用。

借助 rStudio，Amazon SageMaker 为您管理安全组。这意味着 Amazon SageMaker 管理安全组规则，以确保 rSessions 可以访问 rStudioServerPro 应用程序。Amazon SageMaker 为每个用户配置文件创建一个安全组规则。

```
aws sagemaker create-domain --region <REGION> --domain-name <DOMAIN_NAME> \
    --auth-mode <AUTH_MODE> \
    --default-user-settings
 SecurityGroups=<USER_SECURITY_GROUP>,ExecutionRole=<DEFAULT_USER_EXECUTIONROLE> \
    --domain-settings
 SecurityGroupIds=<DOMAIN_SECURITY_GROUP>,RStudioServerProDomainSettings={DomainExecutionRoleArn=<DOMAI
 \
    --vpc-id <VPC_ID> \
    --subnet-ids "<SUBNET_IDS>" \
    --app-network-access-type VPCOnly --app-security-group-management Service
```

注意：rStudioServerPro 应用程序由名为的特殊用户配置文件启动`domain-shared`. 因此，此应用程序不会作为`list-app`任何其他用户配置文件的 API 调用。

您可能必须增加账户中的 Amazon VPC 配额才能增加用户数量。有关更多信息，请参阅 Amazon VPC 配额。

验证域的创建

使用以下命令验证您的域是否已使用`Status`的`InService`. 您的`domain-id`附加到域 ARN 中。例如：`arn:aws:sagemaker:`*<REGION>*`:`*<ACCOUNT_ID>*`:domain/`*<DOMAIN_ID>*。

```
aws sagemaker describe-domain --domain-id <DOMAIN_ID> --region <REGION>
```

管理用户

在启用 RSTUDIO 之后的亚马逊 SageMaker 域正在运行，您可以将用户配置文件（用 UserProfiles）添加到域。以下主题演示了如何创建有权使用 rStudio 的用户配置文件，以及如何更新现有用户配置文件。有关如何删除 RStudio 应用程序、UserProfile 或域的信息，请按照中的步骤操作删除 Amazon SageMaker 域.

Note

总数的限制 UserProfiles 在 Amazon 中 SageMaker 域为 60。

有两种类型的用户：

- 未授权：此用户无法访问 rStudio 应用程序。
- 授权：此用户可以访问 rStudio 应用程序并使用其中一个 rStudio 许可证席位。默认情况下，新用户是`Authorized`是否为 rStudio 启用了域。

如果用户获得授权，则可以向他们授予以下访问 RStudio 级别之一的权限。

- rStudio 用户：这是标准的 rStudio 用户，可以访问 rStudio。
- RStudio 管理员：亚马逊的管理员 SageMaker 域可以创建用户、添加现有用户和更新现有用户的权限。管理员还可以访问 rStudio 管理仪表板。但是，该管理员无法更新由 Amazon SageMaker 管理的参数。

创建用户的方法

以下主题介绍如何在启用 RSTUDIOU 的亚马逊中创建用户。 SageMaker 域。

创建用户 IAM

以下过程演示了如何将用户添加到亚马逊。 SageMaker 使用 IAM 创建的域。有关将 IAM 与 Amazon SageMaker 结合使用的更多信息，请参阅Amazon 的方式 SageMaker 与 IAM 协同工作.

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 导航到控制面板.
3. Select添加用户. 这打开了一个新的用户设置页.
4. UDER用户资料，为您的用户输入一个名称并选择一个 IAM 角色。您可以使用现有角色，也可以创建一个 IAM 角色。IAM 角色必须具有`AmazonSageMakerFullAccess`附加策略。
5. 选择 Next (下一步) 。
6. UDERSageMaker 项目和快速启动中，选择是否启用 Amazon SageMaker 项目模板和亚马逊 SageMaker Studio 用户的 JumpStart。
7. 选择 Next (下一步) 。
8. UDERRStudio Workbench中，验证是否检测到 rStudio Workbench 许可证。
9. UDER许可证授权，选择是否要使用以下授权之一创建用户。
   - Unauthorized
   - RStudio 管理员
   - rStudio 用户
10. 选择 Submit (提交)。

创建用户 SSO

以下过程演示了如何将用户添加到亚马逊。 SageMaker 使用创建域Amazon IAM Identity Center (successor to Amazon Single Sign-On)。 有关的信息Amazon IAM Identity Center (successor to Amazon Single Sign-On)，请参阅是什么Amazon单点登录？.

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 导航到控制面板.
3. Select分配用户和组. 这将打开一个新的 “分配用户和组” 页面。
4. 从列表中选择一个用户或组。有关添加用户和组的信息，请参阅管理中的身份AmazonIAM Identity Center.
5. Select分配用户和组.

创建用户 CLI

以下命令说明如何将用户添加到亚马逊。 SageMaker 具有 IAM 身份验证的域。用户可以属于`R_STUDIO_USER`要么`R_STUDIO_ADMIN`用户组。

```
aws sagemaker create-user-profile --region <REGION> \
    --domain-id <DOMAIN-ID> \
    --user-profile-name <USER_PROFILE_NAME-ID> \
    --user-settings RStudioServerProAppSettings={UserGroup=<USER-GROUP>}
```

以下命令说明如何将用户添加到亚马逊。 SageMaker 带有域IAM Identity Center身份验证。用户可归属于`R_STUDIO_USER`要么`R_STUDIO_ADMIN`用户组。

```
aws sagemaker create-user-profile --region <REGION> \
    --domain-id <DOMAIN-ID> \
    --user-profile-name <USER_PROFILE_NAME-ID> \
    --user-settings RStudioServerProAppSettings={UserGroup=<USER-GROUP>} \
    --single-sign-on-user-identifier UserName \
```

```
    --single-sign-on-user-value <USER-NAME>
```

更新现有用户

您不能更新现有用户。您必须删除现有用户并使用更新授权创建一个新用户。

以另一个用户身份登录 rStudio

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 导航到控制面板.
3. 从用户列表中选择用户名。这将打开一个新页面，其中包含有关用户配置文件和正在运行的应用的详细信息
4. 选择启动应用.
5. 从下拉菜单中，选择rStudio以启动 rStudio 实例。

终止另一个用户的会话

1. 从正在运行的应用程序列表中，确定要删除的应用程序。
2. 点击相应的删除 App您要删除的应用程序的按钮。

删除另一个用户

如果用户正在运行任何应用程序，则无法删除该用户。在尝试删除用户之前删除所有应用程序。

1. 从用户资料页面，选择编辑. 这打开了一个新的常规设置页.
2. UDER删除用户，选择删除用户.

rStudio 管理控制面板

本主题介绍了如何访问和使用 RStudio 管理控制面板。使用 rStudio 管理控制面板，管理员可以管理用户和 RSession，以及查看有关 rStudio Server 实例利用率和 Amazon CloudWatch Logs 的信息。

启动 rStudio 管理仪表板

这些区域有：`R_STUDIO_ADMIN`授权允许用户访问 rStudio 管理仪表板。网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`R_STUDIO_ADMIN`用户可以通过替换来访问 rStudio 管理仪表板`workspaces`和`admin`手动在他们的 rStudio 网址中。下面介绍了如何修改 URL 以访问 rStudio 管理仪表板。

例如，以下 rStudio URL：

```
https://<DOMAIN-ID>.studio.us-east-2.sagemaker.aws/rstudio/default/s/<SESSION-ID>/
workspaces
```

可以转换为：

```
https://<DOMAIN-ID>.studio.us-east-2.sagemaker.aws/rstudio/default/s/<SESSION-ID>/admin
```

控制面板

此选项卡概述了您的 rStudio Server 实例利用率，以及有关活动的 rSession 数量的信息。

“Sessions”

此选项卡提供有关活动的 rSession 的信息，例如启动 rSession 的用户、rSession 的运行时间以及它们的资源利用率。

Users 选项卡

此选项卡提供了有关域中 RStudio 授权用户的信息，例如上次启动 rSession 的时间及其资源利用率。以下过程显示了如何获取有关用户历史资源利用率的信息。

1. 从用户列表中，选择要查看其信息的用户。这将打开一个特定于用户的新页面。
2. 要查看用户的历史资源利用率，请选择统计信息选项卡。此选项卡提供有关历史 CPU 和内存使用情况以及活动的 RSession 数量的信息。
3. 要查看特定于用户的 Amazon CloudWatch Logs，请选择日志选项卡。

Stats 选项卡

此选项卡提供了有关 rStudio Server 实例历史使用率的信息。

日志选项卡

此选项卡显示 rStudio 服务器实例的 Amazon CloudWatch Logs。有关使用 Amazon CloudWatch Logs 记录事件的更多信息，请参阅什么是 Amazon CloudWatch Logs？.

管理账单和成本

要跟踪与 rStudio 环境相关的成本，您可以使用Amazon Billing and Cost Management服务。Amazon Billing and Cost ManagementBilling 提供了多种有用的工具，以帮助您收集有关成本和使用情况的信息、分析成本驱动因素和使用趋势以编制支出预算。有关更多信息，请参阅 。是什么AmazonBilling and Cost Management？.

以下介绍了在 Amazon SageMaker 上运行 RStudio 所需的组件，以及每个组件如何将每个组件纳入 RStudio 实例的计费。

- rStudio 许可证 — 在 Amazon SageMaker 中使用您的 RStudio 许可证不会产生额外的费用。有关您的 RStudio 许可证的更多信息，请参阅rStudio 许可证 (p. 133).
- rSession-这些是最终用户启动的 rStudio 工作会话。在 rSession 运行时，您需要向您收费。
- rStudio 服务器-多租户服务器管理所有 rSession。您可以选择在其上运行 RStudio Server 的实例类型，并支付相关费用。默认实例 “系统” 是免费的，但您可以选择支付更高级别的费用。有关 RStudio Server 可用实例类型的更多信息，请参阅RStudioServerPro 实例类型 (p. 134).

在用户级别跟踪账单

要使用成本分配标签在用户级别跟踪账单，请参阅使用成本分配标签.

诊断问题并获得支持

下面几节介绍如何诊断 Amazon SageMaker 上的 RStudio 的问题。 要在 Amazon SageMaker 上获得对 rStudio 的支持，请联系 Amazon SageMaker 支持。有关购买 RStudio 许可证或修改许可证席位数的帮助，请联系sales@rstudio.com.

查看指标和日志

您可以在 Amazon SageMaker 上使用 rStudio 时监控工作流程性能。使用 rStudio 管理控制面板或 Amazon CloudWatch 查看有关指标的数据日志和信息。

从 rStudio 管理控制面板查看您的 rStudio 日志

您可以直接从 rStudio 管理仪表板查看指标和日志。

1. 登录您的Amazon SageMaker 域名.
2. 按照中的步骤导航到 rStudio 管理仪表板rStudio 管理控制面板 (p. 141).
3. 选择日志选项卡。

从 Amazon CloudWatch Logs 中查看您的 rStudio 日志

Amazon CloudWatch 实时监控您的 Amazon 资源以及在 Amazon 上运行的应用程序。您可以使用 Amazon CloudWatch 收集和跟踪指标，这些指标是您可衡量的相关资源和应用程序的变量。为确保 RStudio 应用程序具有 Amazon CloudWatch 的权限，您必须包含中所述的权限注册到 Amazon SageMaker 域 (p. 32). 收集 Amazon CloudWatch Logs 不需要进行任何设置。

下面的步骤说明了如何查看您的 rSession 的 Amazon CloudWatch Logs。

这些日志可在`/aws/sagemaker/studio`来自AmazonCloudWatch 控制台。

1. 访问 https://console.aws.amazon.com/cloudwatch/，打开 CloudWatch 控制台。
2. Select`Logs`从左侧。从下拉菜单中，选择`Log groups`.
3. 在存储库的`Log groups`屏幕，搜索`aws/sagemaker/studio`. 选择日志组。
4. 在存储库的`aws/sagemaker/studio Log group`屏幕上，导航到`Log streams`选项卡。
5. 要查找您的域的日志，请搜索`Log streams`使用下列格式：

```
<DomainId>/domain-shared/rstudioserverpro/default
```

在 Amazon SageMaker 上使用 RStudio

借助 Amazon SageMaker 中的 RStudio 支持，您可以使用制作工作流程并充分利用 SageMaker 功能。以下主题介绍了如何启动 RStudio 会话并完成关键工作流。有关在 SageMaker 上管理 RStudio 的信息，请参阅Amazon SageMaker 上的 (p. 132).

有关在启用了 rStudio 的情况下创建 Amazon SageMaker 域的入门步骤的信息，请参阅注册到 Amazon SageMaker 域 (p. 32).

有关的信息AmazonSageMaker 上的 RStudio 受支持的区域，请参阅支持的区域和配额 (p. 28).

主题

- 在 rStudio 中进行协作 (p. 143)
- Base R 图像 (p. 144)
- 打开 rStudio 启动器并启动 rSession (p. 144)
- 发布到 rStudio Connect (p. 145)
- 使用 Amazon SageMaker 上的 rStudio 访问 Amazon SageMaker 功能 (p. 145)

在 rStudio 中进行协作

要共享你的 rStudio 项目，你可以将 rStudio 连接到 Git 仓库。有关设置此设置的信息，请参阅。使用 Git 和 SVN 进行版本控制.

注意：在 Amazon SageMaker 上使用 rStudio 时，目前不支持项目共享和实时协作。

Base R 图像

启动 rStudio 实例时，Base R 映像作为实例的基础。此图像扩展了r-会话完成Docker 镜像。

此 Base R 图片包括以下内容：

- R v4.0 或更高版本
- `awscli`、`sagemaker`, 和`boto3`Python 软件包
- 网纹用于 R SDK 集成的软件包

打开 rStudio 启动器并启动 rSession

以下主题介绍了如何使用 RStudio 启动器启动 RSession。

打开 RStudio 启动器

从 Amazon SageMaker 控制台打开 rStudio 启动器

1. 打开亚马逊 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 导航到控制面板.
3. 创建用户后，选择启动应用.
4. 选择任意工作室要么rStudio启动该类型的新应用程序。

从中打开 rStudio 启动器Amazon CLI

使用打开 rStudio 启动器的过程Amazon CLI根据用于管理您的用户的方法而有所不同。

SSO

1. 使用IAM Identity Center打开亚马逊的门户 SageMaker 域。
2. 如下所示将 URL 路径修改为 “/rstudio /default”。

```
#Studio URL
https://<domain-id>.studio.<region>.sagemaker.aws/jupyter/default/lab

#modified URL
https://<domain-id>.studio.<region>.sagemaker.aws/rstudio/default
```

IAM

要从Amazon CLI在 IAM 模式下，请完成以下过程。

1. 使用以下命令创建预签名 URL。

```
aws sagemaker create-presigned-domain-url --region <REGION> \
    --domain-id <DOMAIN-ID> \
    --user-profile-name <USER-PROFILE-NAME>
```

2. 附加&Rerectt = RSTUDIO ServerPro转到生成的 URL。
3. 导航到更新后的 URL。

启动 Session

启动完 RStudio 启动器后，您可以创建新的 rSession。

1. Select新会话.
2. 输入会话名.
3. 选择您的 rSession 运行的实例类型。默认值为 `ml.t3.medium`。
4. 选择您的 rSession 用作内核的映像。
5. 选择启动会话。
6. 创建完会话后，您可以通过选择名称启动它。

暂停您的 Ressession

1. 在 rStudio 启动器中，确定要暂停的 rSession。
2. Select暂停对于会话。

删除您的 Ressession

1. 从 rStudio 启动器中，确定要删除的 rSession。
2. Select退出对于会话。这打开了一个新的退出会话窗口。
3. 从退出会话窗口中，选择退出，以结束会话中的所有子进程。
4. Select退出会话以确认删除会话。

发布到 rStudio Connect

rStudio Connect 使数据科学家能够发布来自 Amazon SageMaker 上的 RStudio 的见解、仪表板和 Web 应用程序。有关更多信息，请参阅 。在 Amazon SageMaker 上的 rStudio 中进行机器学习开发的 RStudio Connect 和软件包管理器.

有关 rStudio Connect 的更多信息，请参阅rStudio Connect 用户指南.

使用 Amazon SageMaker 上的 rStudio 访问 Amazon SageMaker 功能

在 Amazon SageMaker 上使用 rStudio 的好处之一是集成了 Amazon SageMaker 功能。这包括与 Amazon SageMaker Studio 和网状图集成。

在 Amazon SageMaker 上使用 Amazon SageMaker 工作室 JupyterLab 和 rStudio

您的 Amazon SageMaker Studio JupyterLab 和 rStudio 实例共享相同的 Amazon EFS 文件系统。这意味着您使用 JupyterLab 导入和创建的文件可以使用 rStudio 访问，反之亦然。这使您可以使用 jupyterLab 和 rStudio 处理相同的文件，而无需在两者之间移动文件。有关此工作流的更多信息，请参阅宣布在 Amazon SageMaker 上为数据科学家推出完全托管的 rStudio博客。

使用 Amazon SageMaker 开发工具包和网格

这些区域有：网纹包用作 R 到的接口Amazon SageMaker Python 开发工具包以便对 Amazon SageMaker 进行 API 调用。网状软件包可在 R 和 Python 对象之间进行转换，Amazon SageMaker 提供了一个无服务器数据科学环境，用于大规模训练和部署 Machine Learning (ML) 模型。有关网格包的一般信息，请参阅R 到 Python 的接口.

有关概述如何在 Amazon SageMaker 上使用网格模拟包的博客，请参阅在 Amazon SageMaker 中使用 R.

以下各示例介绍如何在特定使用案例中使用网状。

- 有关描述如何使用网状进行批量转换进行预测的笔记本，请参阅在 Amazon SageMaker 中使用 R 进行 Batch 转换.

- 有关描述如何使用 griculate 进行超参数调整和生成预测的笔记本，请参阅将 R 与 Amazon SageMaker 结合使用超参数优化.

使用 Amazon SageMaker 笔记本实例

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的亚马逊 SageMaker 笔记本实例是一个运行 Jupyter 笔记本应用程序的机器学习 (ML) 计算实例。 SageMaker 管理实例和相关资源的创建。在您的笔记本实例中使用 Jupyter 笔记本来准备和处理数据，编写代码以训练模型，部署模型到 SageMaker 托管、测试或验证模型。

SageMaker 还提供了包含完整代码演练的示例笔记本。这些演练介绍了如何使用 SageMaker 以执行常见的机器学习任务。有关更多信息，请参阅 示例笔记本 (p. 158)。

该视频介绍了如何设置和使用 SageMaker 笔记本实例。（长度：26:04）

本视频深入了解如何使用 SageMaker 笔记本实例。（长度：16:44）

主题

- 亚马逊 Linux 2 vs 亚马逊 Linux 笔记本实例 (p. 147)
- 创建笔记本实例 (p. 149)
- 访问笔记本实例 (p. 151)
- 更新笔记本实例 (p. 152)
- 使用生命周期配置脚本自定义笔记本实例 (p. 152)
- 示例笔记本 (p. 158)
- 设置笔记本内核 (p. 160)
- 将 Git 存储库与关联 SageMaker 笔记本实例 (p. 161)
- 笔记本实例元数 (p. 168)
- 监控 Amazon Jupyter 日志 CloudWatch 日志 (p. 168)

亚马逊 Linux 2 vs 亚马逊 Linux 笔记本实例

亚马逊 SageMaker 笔记本实例目前支持亚马逊 Linux 2 (AL2) 和亚马逊 Linux (AL1) 操作系统。创建笔记本实例时，您可以选择笔记本实例所基于的操作系统。2021 年 8 月 18 日之前创建的笔记本实例自动在 AL1 上运行。基于 AL1 的笔记本实例将输入维护阶段截至 2022 年 4 月 18 日。要替换 AL1，买家现在可以选择创建亚马逊 SageMaker 带有 AL2 的笔记本实例。AL1 维护阶段也与 Python 2 和 Chainer 的弃用相吻合。基于 AL2 的笔记本电脑没有托管 Python 2 和 Chainer 内核。

AL1 维护阶段计划

下表概述了 AL1 进入延长维护阶段的时间表。

日期	描述
08/18/2021	启动基于 AL2 的笔记本实例。新启动的笔记本实例仍默认为 AL1。AL1 受安全补丁和更新支持，但没有新功能。在启动新的笔记本实例时，客户可以在两个操作系统中进行选择。
04/18/2022	安全补丁和更新不再支持 AL1。笔记本实例默认为 AL2。客户仍然可以在 AL1 上启动实例，但承担与使用不受支持的操作系统相关的风险。

可用内核

`notebook-al1-v1`：基于 Amazon Linux 平台的笔记本实例中可以使用以下内核。

内核名称
R
Sparkmagic（PySpark）
Spark Magic（Spark）
Sparkmagic（SparkR）
conda_amazonei_mxnet_p27
conda_amazonei_mxnet_p36
conda_amazonei_pytorch_latest_p36
conda_amazonei_tensorflow2_p27
conda_amazonei_tensorflow2_p36
conda_amazonei_tensorflow_p27
conda_amazonei_tensorflow_p36
conda_chainer_p27
conda_chainer_p36
conda_mxnet_latest_p37
conda_mxnet_p27
conda_mxnet_p36
conda_python2
conda_python3
conda_pytorch_latest_p36
conda_pytorch_p27
conda_pytorch_p36
conda_tensorflow2_p36
conda_tensorflow_p27
conda_tensorflow_p36

`notebook-al2-v1`：基于 Amazon Linux 2 平台的笔记本电脑实例中可用以下内核。

内核名称
R

内核名称
Sparkmagic（PySpark）
Spark Magic（Spark）
Sparkmagic（SparkR）
conda_amazonei_mxnet_p36
conda_amazonei_pytorch_latest_p37
conda_amazonei_tensorflow2_p36
conda_mxnet_p37
conda_python3
conda_pytorch_p38
conda_tensorflow2_p38

迁移到 Amazon Linux 2

您现有的笔记本实例不会自动迁移到 Amazon Linux 2。要将笔记本实例升级到 Amazon Linux 2，您必须创建新的笔记本实例、复制代码和环境，然后删除旧的笔记本实例。有关更多信息，请参阅 。Amazon Linux 2 迁移博客.

创建笔记本实例

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的亚马逊 SageMaker 笔记本实例是一个运行 Jupyter 笔记本应用程序的 ML 计算实例。 SageMaker 管理实例和相关资源的创建。在您的笔记本实例中使用 Jupyter 笔记本来准备和处理数据，编写代码以训练模型，部署模型到 SageMaker 托管、测试或验证模型。

要创建笔记本实例，请使用 SageMaker 控制台或
`CreateNotebookInstance`API。

您选择的笔记本实例类型取决于您使用笔记本实例的方式。您希望确保笔记本实例不受内存、CPU 或 IO 的约束。如果您计划将数据集加载到笔记本实例上的内存中以进行探索或预处理，我们建议您为数据集选择具有足够 RAM 内存的实例类型。这将需要具有至少 16 GB 内存的实例（超大型或大型）。如果您计划使用笔记本进行计算密集型预处理，我们建议您选择计算优化的实例，例如 c4 或 c5。

使用 SageMaker 笔记本实例是要使用笔记本实例协调其他Amazon服务。例如，您可以通过调用以下方式使用笔记本实例管理大型数据集处理。Amazon用于 ETL（提取、转换和加载）服务的 Glue 或 Amazon EMR，用于使用 Hadoop 进行映射和减少数据。您可以使用Amazon服务作为数据计算或存储的临时形式。

您可以使用 Amazon S3 存储桶存储和检索您的训练和测试数据。然后您可以使用 SageMaker 要训练和构建模型，所以笔记本的实例类型不会影响模型训练和测试的速度。

收到请求后， SageMaker 执行以下操作：

- 创建网络接口— 如果您选择可选的 VPC 配置， SageMaker 在 VPC 中创建网络接口。它使用您在请求中提供的子网 ID 来确定在哪个可用区中创建子网。 SageMaker 将您在请求中提供的安全组与该子网关联。有关更多信息，请参阅 将 VPC 中的笔记本实例 Connect 到外部资源 (p. 2783)。
- 启动 ML 计算实例—SageMaker 在中启动 ML 计算实例 SageMaker VPC。 SageMaker 要执行配置任务以便能够管理您的笔记本实例，并且如果您指定了 VPC，则会启用 VPC 与笔记本实例之间的流量。

- 安装 Anaconda 程序包和库用于常见的深度学习平台。—SageMaker 安装随安装程序提供的所有 Anaconda 程序包。有关更多信息，请参阅 Anaconda 程序包列表。此外， SageMaker 安装 TensorFlow 和 Apache MXNet 深度学习库。
- 附加 ML 存储卷—SageMaker 将 ML 存储卷附加到 ML 计算实例。您可以将该卷作为工作区域，用于清理训练数据集或者临时存储验证、测试或其他数据。以 1 GB 为增量，为卷选择 5 GB 到 16384 GB 之间的任意大小。默认值为 5 GB。所以 ML 存储卷是加密的，因此 SageMaker 无法确定该卷上的可用空间量。因此，您可以在更新笔记本实例时增加卷大小，但无法减小卷大小。如果要减小正在使用的 ML 存储卷的大小，请创建一个具有所需大小的新笔记本实例。

  只有保存在 `/home/ec2-user/SageMaker` 文件夹中的文件和数据才会在笔记本实例会话之间保留。在笔记本实例停止并重新启动时，保存在此目录之外的文件和数据将被覆盖。每个笔记本实例的 /tmp 目录在实例存储中提供至少 10 GB 的存储。实例存储是非持久的临时性块级存储。实例停止或重新启动时，SageMaker 删除目录的内容。此临时存储是笔记本实例实例的根卷的一部分。
- 复制 Jupyter 笔记本— 这些 Python 代码示例使用各种算法和训练数据集演示了模型训练和托管练习。

创建 SageMaker 笔记本实例：

1. 打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 选择 Notebook instances (笔记本实例)，然后选择 Create notebook instance (创建笔记本实例)。
3. 在 Create notebook instance (创建笔记本实例) 页面上提供以下信息：

   a. 对于 Notebook instance name (笔记本实例名称)，请键入您的笔记本实例的名称。

   b. 适用于笔记本实例类型中，请选择适合您的使用案例的实例大小。有关受支持实例类型和配额的列表，请参阅。Amazon SageMaker Service Quotas.

   c. 对于 Elastic Inference，如果您计划从笔记本实例开展推理，请选择一种与笔记本实例关联的推理加速器类型，或选择无。有关 Elastic Inference 的信息，请参阅 使用 Amazon SageMaker Elastic Inference (EI) (p. 2387)。

   d. 适用于平台标识符中，选择要在其上创建笔记本实例的平台类型。此平台类型决定了创建笔记本实例的操作系统。有关平台标识符类型的信息，请参阅亚马逊 Linux 2 vs 亚马逊 Linux 笔记本实例 (p. 147).

   e. （可选）其他配置允许高级用户创建可在您创建或启动实例时运行的 shell 脚本。此脚本称为生命周期配置脚本，可用于设置笔记本的环境或执行其他功能。有关信息，请参阅使用生命周期配置脚本自定义笔记本实例 (p. 152)。

   f. （可选）其他配置还允许您指定附加到笔记本实例的 ML 存储卷的大小，以 GB 为单位。您可以选择介于 5 GB 到 16384 GB 之间的大小，以 1 GB 为增量。您可以使用该卷来清理训练数据集，或临时存储验证或其他数据。

   g. 适用于IAM 角色中，选择您的账户中具有访问 SageMaker 资源所需的权限的现有 IAM 角色，或者选择创建新角色. 如果选择创建新角色、 SageMaker 创建命名为的 IAM 角色`AmazonSageMaker-ExecutionRole-`*`YYYYMMDD`*`T`*`HHmmSS`*. 这些区域有：Amazon管理的策略`AmazonSageMakerFullAccess`附加到角色。该角色提供允许笔记本实例调用的权限。SageMaker 和 Amazon S3。

   h. 对于根访问，要为所有笔记本实例用户启用根访问，请选择启用。要为用户禁用根访问，请选择禁用。如果启用根访问，则所有笔记本实例用户具有管理员权限，并且可以访问和编辑实例上的所有文件。

   i. （可选）加密密钥允许您使用 Amazon Key Management Service (Amazon KMS) 密钥，对连接到笔记本实例的 ML 存储卷上的数据进行加密。如果您计划在 ML 存储卷上存储敏感信息，请考虑加密信息。

   j. （可选）网络允许您将笔记本实例放在 Virtual Private Cloud (VPC) 中。VPC 提供额外的安全性，限制从 VPC 外部的源访问 VPC 中的资源。有关 VPC 的更多信息，请参阅Amazon VPC User Guide.

      要将笔记本实例添加到 VPC，请执行以下操作：

      i. 选择 VPC 和 SubnetId。

ii. 对于 Security groups (安全组)，请选择您的 VPC 的默认安全组。

iii. 如果您希望笔记本实例能够访问 Internet，请启用直接 Internet 访问。对于 Direct internet access (直接 internet 访问)，请选择 Enable (启用)。Internet 访问可能会降低笔记本实例的安全性。有关更多信息，请参阅 将 VPC 中的笔记本实例 Connect 到外部资源 (p. 2783)。

k. （可选）要将 Git 存储库与笔记本实例相关联，请选择一个默认存储库以及最多 3 个额外的存储库。有关更多信息，请参阅 将 Git 存储库与关联 SageMaker 笔记本实例 (p. 161)。

l. 选择创建笔记本实例。

亚马逊在几分钟后 SageMaker 启动 ML 计算实例 — 在本例中是笔记本实例 — 并将 ML 存储卷附加到该实例。笔记本实例有一个预配置的 Jupyter 笔记本服务器和一组 Anaconda 库。有关更多信息，请参阅 `CreateNotebookInstance` API。

4. 在控制台中，当笔记本实例的状态为 `InService` 时，笔记本实例即可使用。选择笔记本名称旁边的打开 Jupyter 以打开经典 Jupyter 控制面板。

您可以选择打开 JupyterLab 以打开 JupyterLab 控制面板。控制面板提供对笔记本实例和包含完整代码演练的示例 SageMaker 笔记本的访问权限。这些演练介绍了如何使用 SageMaker 执行常见的机器学习任务。有关更多信息，请参阅 示例笔记本 (p. 158)。有关更多信息，请参阅 控制对 SageMaker 笔记本实例 (p. 2685)。

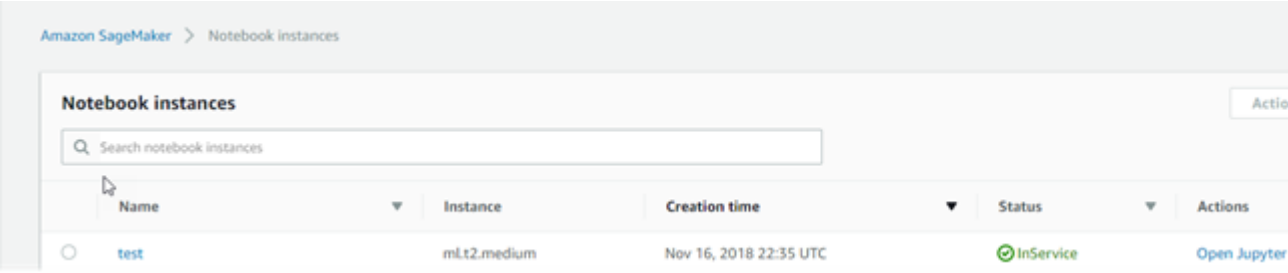
有关 Jupyter 笔记本的更多信息，请参阅 Jupyter 笔记本。

访问笔记本实例

访问您的亚马逊 SageMaker 选择笔记本实例之一，请选择以下选项之一：

- 使用 控制台。

选择 Notebook instance (笔记本实例)。该控制台显示了您账户中的笔记本实例列表。要打开具有标准 Jupyter 界面的笔记本实例，请为该实例选择 Open Jupyter (打开 Jupyter)。使用打开笔记本实例 JupyterLab 界面，选择打开 JupyterLab对于那个例子。



该控制台使用您的登录凭证发送 `CreatePresignedNotebookInstanceUrl`向 SageMaker 发出的 API 请求。 SageMaker 会返回笔记本实例的 URL，并且控制台会在另一个浏览器选项卡中打开该 URL 并显示 Jupyter 笔记本控制面板。

Note

您从调用 `CreatePresignedNotebookInstanceUrl` 获得的 URL 仅在 5 分钟内有效。如果在 5 分钟期限过后使用该 URL，您将被定向到 Amazon Web Services Management Console登录页面。

- 使用 API。

要获取笔记本实例的 URL，请调用 `CreatePresignedNotebookInstanceUrl` API 并使用 API 返回的 URL 打开笔记本实例。

使用 Jupyter 笔记本控制面板来创建和管理笔记本并编写代码。有关 Jupyter 笔记本的更多信息，请参阅 http://jupyter.org/documentation.html。

更新笔记本实例

创建笔记本实例后，您可以使用 SageMaker 和控制台`UpdateNotebookInstance`API 操作。

您可以更新笔记本实例的标签，即`InService`. 要更新笔记本实例的任何其他属性，其状态必须为`Stopped`.

在中更新笔记本实例 SageMaker 控制台：

1. 打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 选择 Notebook instance (笔记本实例)。
3. 选择笔记本实例，选择您要更新的笔记本实例。名称从列表中。
4. 如果你的笔记本状态不是`Stopped`，选择停止按钮来停止笔记本实例。

   执行此操作时，笔记本实例状态更改为`Stopping`. 等到状态变为`Stopped`完成以下步骤：
5. 选择编辑按钮以打开编辑笔记本实例页.
6. 更新笔记本实例并选择更新笔记本实例完成返回笔记本实例页面时，会显示在页面底部的按钮。您的笔记本实例状态更改为正在更新.

   在笔记本实例更新完成后，状态将更改为`Stopped`.

使用生命周期配置脚本自定义笔记本实例

要在笔记本实例上安装程序包或示例笔记本，请为其配置网络和安全性，或者使用 Shell 脚本对其进行自定义以使用生命周期配置。生命周期配置 提供了 Shell 脚本，仅在您创建笔记本实例或者随时启动笔记本实例时运行。当您创建笔记本实例时，您可以创建新的生命周期配置以及它使用的脚本，或者应用您已有的实例。

您还可以使用生命周期配置脚本访问Amazon笔记本电脑提供的服务。例如，您可以创建一个脚本来让您使用笔记本控制其他Amazon资源，例如 Amazon EMR 实例。

我们维护一个笔记本生命周期配置脚本的公有存储库，用于解决自定义笔记本实例的常见使用案例，网址为：https://github.com/aws-samples/amazon-sagemaker-notebook-instance-lifecycle-configuration-samples。

Note

每个脚本只能有 16384 个字符。
可用于这两个脚本的 `$PATH` 环境变量的值是 `/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/sbin:/sbin:/bin`。工作目录（这是 `$PWD` 环境变量的值）是 `/`。
查看 CloudWatch 日志组中笔记本实例生命周期配置的日志`/aws/sagemaker/NotebookInstances`在日志流中`[notebook-instance-name]/[LifecycleConfigHook]`.
脚本运行时间不能超过 5 分钟。如果脚本运行时间超过 5 分钟，它会失败，并且不创建或启动笔记本实例。要帮助减少脚本的运行时间，请尝试采取以下措施：

- 减少所需的步骤。例如，限制在其中安装大型包的 conda 环境。
- 在并行进程中运行任务。
- 在脚本中使用 `nohup` 命令。

您可以通过选择以前创建的笔记本实例生命周期配置列表。生命周期配置中的 SageMaker 控制台。这些笔记本实例生命周期配置在您创建新笔记本实例时可用。

创建生命周期配置

1. 打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在左侧，选择笔记本.
3. 从下拉菜单中，选择生命期配置.
4. 从生命期配置页面，选择创建配置.
5. 对于名称，请键入使用字母数字字符和 - (但不能含空格) 的名称。名称最多可包含 63 个字符。
6. (可选) 要创建在您创建笔记本和每次启动笔记本时运行的脚本，请选择 Start notebook (启动笔记本)。
7. 在 Start notebook (启动笔记本) 编辑器中，键入脚本。
8. （可选）要创建仅运行一次的脚本，则在创建笔记本时，选择 Create notebook (创建笔记本)。
9. 在 Create notebook (创建笔记本) 编辑器中，键入脚本配置网络。
10. 选择 Create configuration (创建配置) 。

生命周期配置最佳实践

以下是使用生命周期配置的最佳实践：

> **Important**
>
> 我们不建议在生命周期配置脚本中存储敏感信息。

- 以 root 用户身份运行生命周期配置。如果您的脚本在 /home/ec2-user/SageMaker 目录中进行任何更改 (例如，通过 pip 安装程序包) ，请使用命令 sudo -u ec2-user 以 ec2-user 用户的身份运行。这是 Amazon 的同一个用户 SageMaker 运行方式。
- SageMaker 笔记本实例使用conda为 Jupyter 笔记本实施不同的内核的环境。如果您要安装对一个或多个笔记本内核可用的程序包，请将用于安装程序包的命令与 conda 环境命令配合使用，以便激活包含您要在其中安装程序包的内核的 conda 环境。

  例如，如果您要仅为 python3 环境安装程序包，请使用以下代码：

```
#!/bin/bash
sudo -u ec2-user -i <<'EOF'

# This will affect only the Jupyter kernel called "conda_python3".
source activate python3

# Replace myPackage with the name of the package you want to install.
pip install myPackage
# You can also perform "conda install" here as well.

source deactivate

EOF
```

  如果您要在笔记本实例的所有 conda 环境中安装程序包，请使用以下代码：

```
#!/bin/bash
sudo -u ec2-user -i <<'EOF'

# Note that "base" is special environment name, include it there as well.
for env in base /home/ec2-user/anaconda3/envs/*; do
    source /home/ec2-user/anaconda3/bin/activate $(basename "$env")

    # Installing packages in the Jupyter system environment can affect stability of your
 SageMaker
    # Notebook Instance.  You can remove this check if you'd like to install Jupyter
 extensions, etc.
```

```
    if [ $env = 'JupyterSystemEnv' ]; then
      continue
    fi

    # Replace myPackage with the name of the package you want to install.
    pip install --upgrade --quiet myPackage
    # You can also perform "conda install" here as well.

    source /home/ec2-user/anaconda3/bin/deactivate
done

EOF
```

- 您必须将所有 conda 环境存储在默认环境文件夹 (/home/user/anaconda3/envs) 中。

  Important

  当您创建或更改脚本时，我们建议您使用提供 Unix 样式换行符的文本编辑器，例如，创建笔记本时可在控制台中使用的文本编辑器。从非 Linux 操作系统复制文本可能会引入不兼容的换行符，并导致意外错误。

在笔记本实例中安装外部库和内核

亚马逊 SageMaker 笔记本实例附带多个已安装的环境。这些环境包含 Jupyter 内核和 Python 程序包，包括：scikit、Pandas、NumPy、TensorFlow 和 MXNet。当您停止和启动笔记本实例时，这些环境以及 `sample-notebooks` 文件夹中的所有文件均会刷新。您也可以安装自己的包含所选程序包和内核的环境。

亚马逊中不同的 Jupyter 内核 SageMaker 笔记本实例是独立的 conda 环境。有关 conda 环境的信息，请参阅管理环境中的Conda文档中）。

在笔记本实例的 Amazon EBS 卷上安装自定义环境和内核。这可以确保在停止和重新启动笔记本实例时，它们会持续存在，并且 SageMaker 不会更新您安装的任何外部库。要做到这一点，请使用生命周期配置，其中包含在您创建笔记本实例时运行的脚本 (`on-create`)以及每次重新启动笔记本实例时运行的脚本 (`on-start`)。有关使用笔记本实例生命周期配置的更多信息，请使用生命周期配置脚本自定义笔记本实例 (p. 152). 有一个 GitHub 存储库，其中包含示例生命周期配置脚本SageMaker 笔记本实例生命周期 Config.

上的示例https://github.com/aws-samples/amazon-sagemaker-notebook-instance-lifecycle-config-samples/blob/master/scripts/persistent-conda-ebs/on-create.sh和https://github.com/aws-samples/amazon-sagemaker-notebook-instance-lifecycle-config-samples/blob/master/scripts/persistent-conda-ebs/on-start.sh展示在笔记本实例上安装环境和内核的最佳实践。这些区域有：`on-create`脚本将安装`ipykernel`库来创建自定义环境作为 Jupyter 内核，然后使用`pip install`和`conda install`安装库。您可以调整脚本以创建自定义环境并安装所需的库。 SageMaker 停止并重新启动 notebook 实例时不会更新这些库，因此您可以确保自定义环境具有所需的特定版本的库。这些区域有：`on-start`脚本会安装您创建为 Jupyter 内核的所有自定义环境，以便它们显示在 Jupyter 的下拉列表中New菜单。

Package 安装工具

SageMaker 笔记本支持以下软件包安装工具：

- Conda 安装
- 安装 pip

您可以使用以下方法安装软件包：

- 生命周期配置脚本。

  有关示例脚本，请参阅SageMaker 笔记本实例生命周期 Config. 有关周期配置的更多信息，请参阅使用生命周期配置脚本自定义笔记本实例.

- 笔记本 — 以下命令受支持。
  - `%conda install`
  - `%pip install`
- Jupyter 终端 — 您可以直接使用 pip 和 conda 安装软件包。

在笔记本中你可以使用 system 命令语法（以! 开头的行）例如，要安装软件包，`!pip install`和`!conda install`. 最近，iPython 中添加了新命令：`%pip`和`%conda`. 这些命令是从笔记本安装软件包的推荐方法，因为它们正确考虑了正在使用的激活环境或解释器。有关更多信息，请参阅 。添加 %pip 和 %conda 魔术函数.

Conda

Conda 是一个开源程序包管理系统和环境管理系统，它可以安装软件包及其依赖项。 SageMaker 支持将 Conda 用于两个主频道之一，默认频道和 conda-FORGE 频道。有关更多信息，请参阅 。Conda 频道. conda-forge 频道是一个社区频道，贡献者可以在其中上传包裹。

> Note
>
> 由于 Conda 如何解决依赖关系图，从 conda-forge 安装软件包可能需要更长的时间（在最坏的情况下，10 分钟以上）。

深度学习 AMI 附带了许多 conda 环境和许多预安装的软件包。由于预安装的软件包的数量，很难找到一组保证兼容的软件包。你可能会看到一条警告：“环境不一致，请仔细检查套餐计划”。尽管有这个警告，SageMaker 确保所有 SageMaker 前提是环境是正确的。 SageMaker 无法保证任何用户安装的软件包都能正常运行。

Conda 有两种激活环境的方法：conda 激活/停用和源代码激活/停用。有关更多信息，请参阅 。我应该在 Linux 中使用 “conda 激活” 还是 “源代码激活”.

SageMaker 支持将 Conda 环境移动到 Amazon EBS 卷上，该卷将在实例停止时保留。当环境安装到根卷时，环境不会持续存在，这是默认行为。有关示例生命周期脚本，请参阅持久性 conda-ebs.

支持的 conda 操作（请参阅本主题底部的注释）

- conda 在单个环境中安装软件包
- conda 在所有环境中安装软件包
- conda 在 R 环境中安装 R 软件包
- 从主 conda 存储库安装软件包
- 从 conda-Forge 安装软件包
- 将 Conda 安装位置更改为使用 EBS
- 同时支持 conda 激活和源激活

Pip

Pip 是安装和管理 Python 软件包的事实上的工具。默认情况下，Pip 会在 Python Package 索引 (PyPI) 上搜索包。与 Conda 不同，pip 没有内置的环境支持，在涉及具有本机/系统库依赖关系的软件包方面也不像 Conda 那么彻底。Pip 可以用来在 Conda 环境中安装软件包。

您可以将替代软件包存储库与 pip 一起使用，而不是 PyPI。有关示例生命周期脚本，请参阅on-start.sh.

支持的点操作（请参阅本主题底部的注释）

- 使用 pip 在没有活动 conda 环境的情况下安装软件包（在系统范围内安装软件包）
- 使用 pip 在 conda 环境中安装软件包
- 使用 pip 在所有 conda 环境中安装软件包

- 更改 pip 安装位置以使用 EBS
- 使用替代存储库使用 pip 安装软件包

不支持

SageMaker 旨在支持尽可能多的软件包安装操作。但是，如果软件包是由 SageMaker 或者 DLAMI，并且您对这些软件包使用以下操作，可能会使笔记本实例不稳定：

- 卸载
- 降级
- 升级

我们不支持通过 yum 安装或从 CRAN 安装 R 软件包来安装软件包。

由于网络条件或配置的潜在问题，或者 Conda 或 PyPi 的可用性，我们无法保证软件包将在固定或确定性的时间内安装。

> **Note**
>
> 我们无法保证软件包安装会成功。尝试在具有不兼容依赖关系的环境中安装软件包可能会导致失败。在这种情况下，你应该联系库维护者，看看是否可以更新软件包依赖关系。或者，您可以尝试以允许安装的方式修改环境。但是，这种修改可能意味着删除或更新现有软件包，这意味着我们无法再保证此环境的稳定性。

笔记本实例软件更新

亚马逊 SageMaker 定期测试和发布在笔记本实例上安装的软件。这包括：

- 内核更新
- 安全补丁
- Amazon 开发工具包更新
- 亚马逊 SageMaker Python 开发工具包更新
- 开源软件更新

在笔记本实例运行时，SageMaker 不会更新软件。要确保您拥有最新的软件更新，请在 SageMaker 控制台或通过调用
`StopNotebookInstance`.

您还可以使用终端或笔记本中的更新命令手动更新笔记本实例运行时在其中安装的软件。

> **Note**
>
> 更新内核和某些程序包可能取决于是否为笔记本实例启用了根访问权限。有关更多信息，请参阅 控制对 SageMaker 笔记本实例 (p. 2685)。

如果您运行的是过时的软件，则笔记本实例不会通知您。你可以查看Personal Health Dashboard或者安全公告安全公告更新。

使用笔记本控制 Amazon EMR Spark 实例

您可以使用使用自定义生命周期配置脚本创建的笔记本实例访问。Amazon笔记本电脑提供的服务。例如，您可以创建一个脚本来让您将笔记本与 Sparkmagic 一起使用以控制其他Amazon资源，例如 Amazon EMR 实例。然后，您可以使用 Amazon EMR 实例处理您的数据，而不是在笔记本上运行数据分析。这允许您创建一个较小的笔记本实例，因为您不会使用该实例来处理数据。当您的大型数据集需要大型笔记本实例来处理数据时，此功能非常有用。

该过程需要使用亚马逊执行三个过程 SageMaker 控制台：

- 创建亚马逊 EMR Spark 实例
- 创建 Jupyter 笔记本
- 测试笔记本到亚马逊的 EMR 连接

使用 Sparkmagic 创建可以从笔记本控制的 Amazon EMR Spark 实例

1. 通过以下链接打开 Amazon EMR 控制台：https://console.aws.amazon.com/elasticmapreduce/。
2. 在导航窗格中，选择 Create cluster (创建集群)。
3. 在存储库的创建群集-快速选项页面，下软件配置，选择Spark：使用 Ganglia 3.7.2 和齐柏林 0.8.2 在 Hadoop 2.8.5 YARN 上使用 Spark 2.4.4.
4. 在页面上设置其他参数，然后选择 Create cluster (创建集群)。
5. 在 Cluster (集群) 页面上，选择您创建的集群名称。记下 Master Public DNS (主公有 DNS)、EMR master's security group (EMR 主设备的安全组) 以及在其中创建 EMR 集群的 VPC 名称和子网 ID。在创建笔记本时，将使用这些值。

要创建使用 Sparkmagic 控制 Amazon EMR Spark 实例的笔记本

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中的 Notebook instances (笔记本实例) 下，选择 Create notebook (创建笔记本)。
3. 输入笔记本实例名称并选择实例类型。
4. 选择 Additional configuration (其他配置)，然后在 Lifecycle configuration (生命周期配置) 下选择 Create a new lifecycle configuration (创建新的生命周期配置)。
5. 将以下代码添加到生命周期配置脚本中：

```
# OVERVIEW
# This script connects an Amazon EMR cluster to an Amazon SageMaker notebook instance
 that uses Sparkmagic.
#
# Note that this script will fail if the Amazon EMR cluster's master node IP address is
 not reachable.
#   1. Ensure that the EMR master node IP is resolvable from the notebook instance.
#      One way to accomplish this is to have the notebook instance and the Amazon EMR
 cluster in the same subnet.
#   2. Ensure the EMR master node security group provides inbound access from the
 notebook instance security group.
#       Type        - Protocol - Port - Source
#       Custom TCP  - TCP      - 8998 - $NOTEBOOK_SECURITY_GROUP
#   3. Ensure the notebook instance has internet connectivity to fetch the SparkMagic
 example config.
#
# https://aws.amazon.com/blogs/machine-learning/build-amazon-sagemaker-notebooks-
backed-by-spark-in-amazon-emr/

# PARAMETERS
EMR_MASTER_IP=your.emr.master.ip


cd /home/ec2-user/.sparkmagic

echo "Fetching Sparkmagic example config from GitHub..."
wget https://raw.githubusercontent.com/jupyter-incubator/sparkmagic/master/sparkmagic/
example_config.json

echo "Replacing EMR master node IP in Sparkmagic config..."
```

```
sed -i -- "s/localhost/$EMR_MASTER_IP/g" example_config.json
mv example_config.json config.json

echo "Sending a sample request to Livy.."
curl "$EMR_MASTER_IP:8998/sessions"
```

6. 在PARAMETERS脚本的部分，替换your.emr.master.ip使用主公有 DNS 名称对于 Amazon EMR 实例。
7. 选择 Create configuration（创建配置）。
8. 在 Create notebook (创建笔记本) 页面上，选择 Network - optional (网络 - 可选)。
9. 选择 Amazon EMR 实例所在的 VPC 和子网。
10. 选择 Amazon EMR 主节点使用的安全组。
11. 选择创建笔记本实例。

创建笔记本实例时，状态为 Pending (待处理)。创建实例并成功运行生命周期配置脚本后，状态为 InService。

Note

如果笔记本实例无法连接到 Amazon EMR 实例， SageMaker 无法创建笔记本实例。如果 Amazon EMR 实例和笔记本不在同一 VPC 和子网中，笔记本未使用 Amazon EMR 主安全组，或者如果脚本中的主公有 DNS 名称不正确，则连接可能会失败。

测试 Amazon EMR 实例与笔记本之间的连接

1. 当笔记本的状态为 InService 时，选择 Open Jupyter (打开 Jupyter) 以打开笔记本。
2. 选择 New (新建)，然后选择 Sparkmagic (PySpark)。
3. 在代码单元格中，输入 **%%info** 并运行单元格。

该输出值应该类似于以下内容

```
Current session configs: {'driverMemory': '1000M', 'executorCores': 2, 'kind':
 'pyspark'}
                    No active sessions.
```

示例笔记本

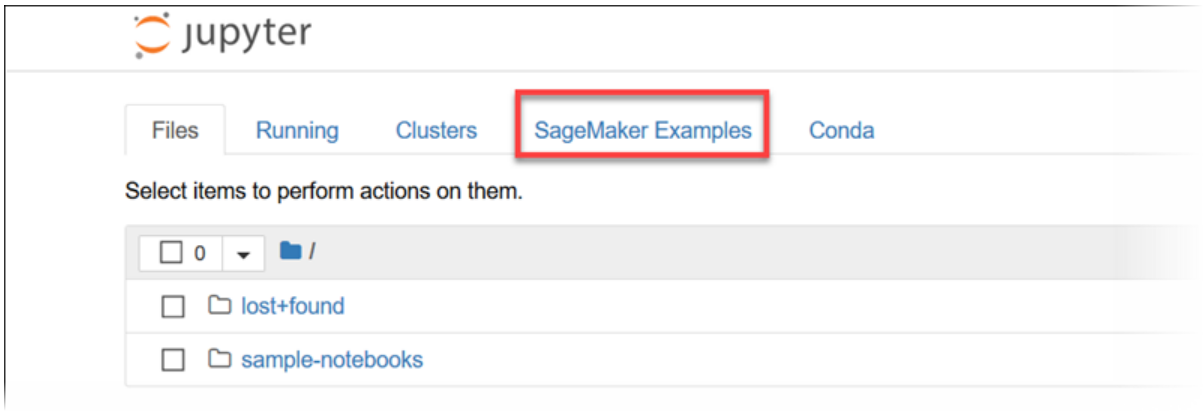
您的笔记本实例包含 Amazon SageMaker 提供的示例笔记本。示例笔记本包含代码，演示如何使用 SageMaker 应用机器学习解决方案。笔记本实例使用 nbexamples Jupyter 扩展，让您能够查看示例笔记本的只读版本，或创建它的副本以便可以修改和运行它。有关 nbexamples 扩展的更多信息，请参阅 https://github.com/danielballan/nbexamples。有关用于的笔记本示例的信息 SageMaker 工作室，请参阅使用 Amazon SageMaker Studio Notebooks (p. 82).

Note

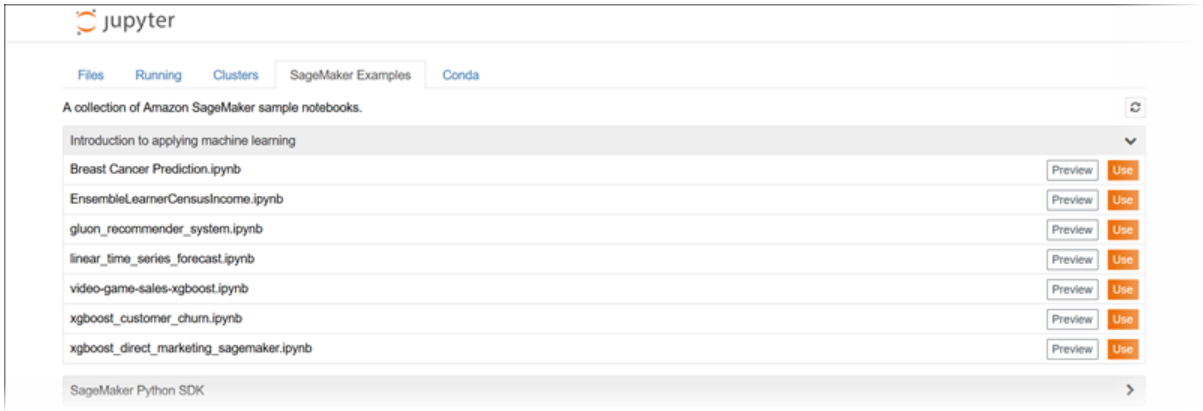
示例笔记本通常从 Internet 下载数据集。如果您在创建笔记本实例时禁用 SageMaker 提供的 Internet 访问，示例笔记本可能会无法工作。有关更多信息，请参阅将 VPC 中的笔记本实例 Connect 到外部资源 (p. 2783)。

在 Jupyter 经典视图中使用或查看示例笔记本

要在 Jupyter 经典视图中查看或使用示例笔记本，请选择 SageMaker Examples (SageMaker 示例) 选项卡。

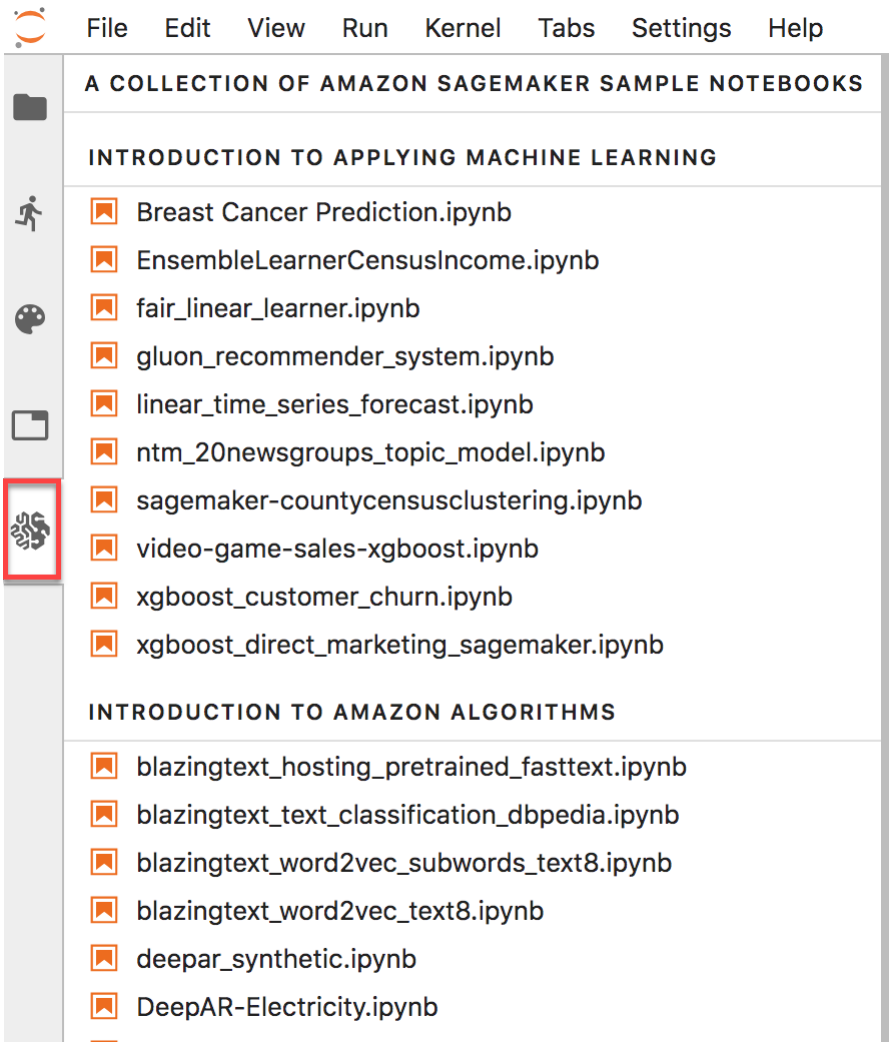


要在 Jupyter 经典视图中查看示例笔记本的只读版本，请在 SageMaker Examples (SageMaker 示例) 选项卡上，为该笔记本选择 Preview (预览)。要在笔记本实例的主目录中创建示例笔记本的副本，请选择 Use (使用)。在对话框中，您可在保存前更改笔记本名称。



在 Jupyterlab 视图中使用或查看示例笔记本

要在 Jupyterlab 视图中查看或使用示例笔记本，请选择左侧导航面板中的示例图标。



要查看示例笔记本的只读版本，请选择笔记本的名称。这将在主区域中以选项卡的方式打开笔记本。要在笔记本实例的主目录中创建示例笔记本的副本，请在顶部横幅中选择 Create a Copy (创建副本)。在对话框中，键入笔记本的名称，然后选择 CREATE COPY (创建副本)。

有关示例笔记本的更多信息，请参阅GitHub 存储库示例.

设置笔记本内核

亚马逊 SageMaker 为 Jupyter 提供了多种内核，针对 Python 2 和 3、Apache MXNet、TensorFlow 和 PySpark 提供支持。要在 Jupyter 笔记本控制面板中为新笔记本设置内核，请选择 New (新建)，然后从列表中选择内核。

您也可以创建可在笔记本实例中使用的自定义内核。有关信息，请参阅在笔记本实例中安装外部库和内核 (p. 154)。

将 Git 存储库与关联 SageMaker 笔记本实例

将 Git 存储库与您的笔记本实例关联，以将笔记本保存到源代码控制环境中，即使您停止或删除笔记本实例仍可持久保存。您可以将一个默认存储库与最多三个其他存储库与一个笔记本实例关联。存储库可以托管在 Amazon CodeCommit、GitHub 或任何其他 Git 服务器中。将 Git 存储库与笔记本实例关联可用于下列情况：

- 持久性-笔记本实例中的笔记本存储在持久的 Amazon EBS 卷上，但不会在笔记本实例的生命周期之外保存。将笔记本存储在 Git 存储库中使您可以存储和使用笔记本，即使您停止或删除了笔记本实例。
- 协作 - 团队中的同事通常会一起处理机器学习项目。在 Git 存储库中存储笔记本使得同事可以处理不同笔记本实例，以在源代码控制环境中共享笔记本并进行协作。
- 学习 - 许多演示机器学习技术的 Jupyter 笔记本可在公开托管的 Git 存储库上提供，例如 GitHub。您可以将笔记本实例与存储库关联，从而轻松地加载包含在该存储库中的 Jupyter 笔记本。

有两种方法将 Git 存储库与笔记本实例关联：

- 添加 Git 存储库作为亚马逊中的资源 SageMaker account. 然后，要访问存储库，您可以指定包含凭证的 Amazon Secrets Manager 密钥。这样，您可以访问需要身份验证的存储库。
- 关联并非您账户中资源的公有 Git 存储库。如果这样做，您不能指定访问存储库的凭证。

主题

- 将 Git 存储库添加到您的亚马逊 SageMaker 账户 (p. 161)
- 创建具有关联 Git 存储库的笔记本实例 (p. 164)
- 关联 CodeCommit 不同的存储库Amazon笔记本实例的账户 (p. 165)
- 在笔记本实例中使用 Git 存储库 (p. 166)

将 Git 存储库添加到您的亚马逊 SageMaker 账户

管理 GitHub 存储库，轻松将其与笔记本实例关联，并为需要身份验证的存储库关联凭证，请在 Amazon 中添加存储库作为资源。 SageMaker account. 您可以在中查看存储在账户中的存储库列表以及各个存储库的详细信息。 SageMaker 控制台和使用 API。

您可以将 Git 存储库添加到 SageMaker 中的账户 SageMaker 控制台或使用Amazon CLI.

> Note
>
> 您可以使用 SageMaker API

`CreateCodeRepository`将 Git 存储库添加到你的 SageMaker 账户，但 step-by-step 这里没有提供说明。

将 Git 存储库添加到您的 SageMaker 账户（控制台）

将 Git 存储库添加为您的中的资源 SageMaker 帐户

1. 打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. UNDER笔记本，选择Git 存储库，然后选择添加存储库.
3. 添加 CodeCommit 存储库，选择AmazonCodeCommit. 添加 GitHub 或其他基于 Git 的存储库，请选择GitHub/OthOthHub/Oth基于GitHub.

添加现有的 CodeCommit 知识库

1. 选择 Use existing repository (使用现有存储库)。
2. 对于 Repository (存储库)，从列表中选择存储库。
3. 在 SageMaker 中输入要用于存储库的名称。名称必须介于 1 到 63 个字符之间。有效字符为 a-z、A-Z、0-9 和 -（连字符）。
4. 选择 Add repository (添加存储库)。

创建新的 CodeCommit 知识库

1. 选择 Create new repository (创建新存储库)。
2. 输入存储库的名称，您可以在两者中使用的名称。 CodeCommit 和 SageMaker。名称必须介于 1 到 63 个字符之间。有效字符为 a-z、A-Z、0-9 和 -（连字符）。
3. 选择 Create repository（创建存储库）。

要添加在不是 CodeCommit 之外位置托管的 Git 存储库，

1. 选择 GitHub/Other Git-based repo (GitHub/其他基于 Git 的存储库)。
2. 输入最多 63 个字符的名称。有效字符包括字母数字字符、连字符 (-) 和 0-9。
3. 为存储库输入 URL。不要在 URL 中提供用户名。请在 Amazon Secrets Manager 中添加用户名和密码，如下一步中所述。
4. 对于 Git credentials (Git 凭证)，选择用于对存储库进行身份验证的凭证。只有在 Git 存储库是私有时才需要此项。

   Note

   如果为 Git 存储库启用了双重身份验证，请使用 Git 服务提供商生成的个人访问令牌而非密码。

   a. 要使用现有 Amazon Secrets Manager 密钥，请选择 Use existing secret (使用现有密钥)，然后从列表中选择密钥。有关创建和存储密钥的更多信息，请参阅创建基本密钥中的AmazonSecrets Manager 用户指南. 您使用的密钥名称必须包含 `sagemaker` 字符串。

      Note

      密钥必须具有 `AWSCURRENT` 的暂存标签并且必须使用以下格式：
      `{"username": UserName, "password": Password}`
      适用于 GitHub 存储库，我们建议使用个人访问令牌而不是您的账户密码。有关信息，请参阅 https://help.github.com/articles/creating-a-personal-access-token-for-the-command-line/。

   b. 要创建新 Amazon Secrets Manager 密钥，请选择 Create secret (创建密钥)，输入密钥的名称，然后输入用于对存储库进行身份验证的用户名和密码。密钥名称必须包含 `sagemaker` 字符串。

Note

您用于创建密钥的 IAM 角色必须具有`secretsmanager:GetSecretValue`IAM 策略中的权限。
密钥必须具有 `AWSCURRENT` 的暂存标签并且必须使用以下格式：
`{"username": UserName, "password": Password}`
适用于 GitHub 存储库，我们建议使用个人访问令牌而不是您的账户密码。

c. 要不使用任何凭证，请选择 No secret (无密钥)。

5. 选择 Create secret (创建密钥)。

将 Git 存储库添加到您的 Amazon SageMaker 账户 (CLI)

使用 `create-code-repository`Amazon CLI 命令。指定存储库的名称作为 `code-repository-name` 参数的值。名称必须介于 1 到 63 个字符之间。有效字符为 a-z、A-Z、0-9 和 - (连字符) 。另请指定以下内容：

- 默认分支
- Git 存储库的 URL

  Note

  不要在 URL 中提供用户名。请在 Amazon Secrets Manager 中添加用户名和密码，如下一步中所述。

- Amazon Secrets Manager 密钥的 Amazon 资源名称 (ARN)，该密钥包含用于进行存储库身份验证的凭证作为 `git-config` 参数的值。

有关创建和存储密钥的信息，请参阅。创建基本密钥中的AmazonSecrets Manager 用户指南. 以下命令创建名为的新存储库：`MyRespository`在你的亚马逊中 SageMaker 指向托管在的 Git 存储库的账户`https://github.com/myprofile/my-repo"`.

针对 Linux、OS X 或 Unix：

```
aws sagemaker create-code-repository \
                    --code-repository-name "MyRepository" \
                    --git-config '{"Branch":"master", "RepositoryUrl" :
                    "https://github.com/myprofile/my-repo", "SecretArn" :
 "arn:aws:secretsmanager:us-east-2:012345678901:secret:my-secret-ABc0DE"}'
```

对于 Windows：

```
aws sagemaker create-code-repository ^
                    --code-repository-name "MyRepository" ^
                    --git-config "{\"Branch\":\"master\", \"RepositoryUrl\" :
                    \"https://github.com/myprofile/my-repo\", \"SecretArn\" :
 \"arn:aws:secretsmanager:us-east-2:012345678901:secret:my-secret-ABc0DE\"}"
```

Note

密钥必须具有 `AWSCURRENT` 的暂存标签并且必须使用以下格式：
`{"username": UserName, "password": Password}`
适用于 GitHub 存储库，我们建议使用个人访问令牌而不是您的账户密码。

创建具有关联 Git 存储库的笔记本实例

在使用 Amazon Web Services Management Console 或 Amazon CLI 创建笔记本实例时，您可以将 Git 存储库与笔记本实例关联。如果你想使用不同的 CodeCommit 存储库Amazon账户而不是笔记本实例，请为存储库设置跨账户访问权限。有关信息，请参阅关联 CodeCommit 不同的存储库Amazon笔记本实例的账户 (p. 165)。

主题

- 创建具有关联 Git 存储库的笔记本实例（控制台） (p. 164)
- 创建具有关联 Git 存储库的笔记本实例 (CLI) (p. 164)

创建具有关联 Git 存储库的笔记本实例（控制台）

在 Amazon SageMaker 控制台中创建笔记本实例并关联 Git 存储库

1. 按照第 1 步：创建 Amazon SageMaker 笔记本实例 (p. 59)中的说明进行操作。
2. 对于 Git repositories (Git 存储库)，选择与笔记本实例关联的 Git 存储库。

   a. 适用于默认存储库中，选择您要用作默认存储库的存储库。SageMaker 克隆此存储库作为位于的 Jupyter 启动目录中的子目录。`/home/ec2-user/SageMaker`. 当您打开笔记本实例时，它在此存储库中打开。要选择作为资源存储在您账户中的存储库，请从列表中选择其名称。要将新存储库添加为您账户中的资源，请选择将存储库添加到 SageMaker （在新窗口中打开 Add repository (添加存储库)然后按照中的说明操作创建具有关联 Git 存储库的笔记本实例（控制台） (p. 164). 要克隆存储在您账户中的公有存储库，请选择 Clone a public Git repository to this notebook instance only (仅将公有 Git 存储库克隆到此笔记本实例)，然后指定该存储库的 URL。

   b. 适用于额外的存储库 1中，选择您要添加作为额外目录的存储库。SageMaker 克隆此存储库作为位于的 Jupyter 启动目录中的子目录。`/home/ec2-user/SageMaker`. 要选择作为资源存储在您账户中的存储库，请从列表中选择其名称。要将新存储库添加为您账户中的资源，请选择将存储库添加到 SageMaker （在新窗口中打开 Add repository (添加存储库)然后按照中的说明操作创建具有关联 Git 存储库的笔记本实例（控制台） (p. 164). 要克隆存储在您账户中的存储库，请选择 Clone a public Git repository to this notebook instance only (仅将公有 Git 存储库克隆到此笔记本实例)，然后指定该存储库的 URL。

   重复此步骤最多三次，以添加最多三个额外的存储库到您的笔记本实例。

创建具有关联 Git 存储库的笔记本实例 (CLI)

要使用 Amazon CLI 创建笔记本实例并关联 Git 存储库，请使用 `create-notebook-instance` 命令，如下所示：

- 指定您要用作默认存储库的存储库作为`default-code-repository`参数。Amazon SageMaker 克隆此存储库作为位于的 Jupyter 启动目录中的子目录。`/home/ec2-user/SageMaker`. 当您打开笔记本实例时，它在此存储库中打开。使用作为资源存储在您的中的存储库 SageMaker 账户，请指定存储库的名称作为`default-code-repository`参数。要使用未存储在您账户中的存储库，请指定存储库的 URL 作为 `default-code-repository` 参数的值。
- 指定最多三个额外存储库作为`additional-code-repositories`参数。 SageMaker 将此存储库克隆为 Jupyter 启动目录中的子目录`/home/ec2-user/SageMaker`，并且通过将存储库添加到默认存储库中将其排除在`.git/info/exclude`默认存储库的目录。使用作为资源存储在 SageMaker 账户中，指定存储库的名称作为的值。`additional-code-repositories`参数。要使用未存储在您账户中的存储库，请指定这些存储库的 URL 作为 `additional-code-repositories` 参数的值。

例如，以下命令创建一个笔记本实例，具有名为 `MyGitRepo` 的存储库作为默认存储库，存储作为您 SageMaker 账户中的资源，另有托管在 GitHub 上的额外存储库：

```
aws sagemaker create-notebook-instance \
                    --notebook-instance-name "MyNotebookInstance" \
                    --instance-type "ml.t2.medium" \
                    --role-arn "arn:aws:iam::012345678901:role/service-role/
AmazonSageMaker-ExecutionRole-20181129T121390" \
                    --default-code-repository "MyGitRepo" \
                    --additional-code-repositories "https://github.com/myprofile/my-other-
repo"
```

Note

如果您使用Amazon CodeCommit 名称中不包含 “SageMaker” 的存储库，请添加`codecommit:GitPull`和`codecommit:GitPush`对您作为传递的角色的权限`role-arn`参数到`create-notebook-instance`命令。有关如何向角色添加权限的信息，请参阅添加和删除 IAM 策略中的Amazon Identity and Access Management用户指南.

关联 CodeCommit 不同的存储库Amazon笔记本实例的账户

关联 CodeCommit 存储库在另一个Amazon为您的笔记本实例设置跨账户访问权限 CodeCommit 存储库。

设置跨账户访问权限 CodeCommit 存储库并将其与笔记本实例关联：

1. 在Amazon包含 CodeCommit 创建一个 IAM 策略，该策略允许包含笔记本实例的账户中的用户访问存储库。想要了解有关信息，请参阅第 1 步：创建 Account 中的存储库访问策略中的CodeCommit 用户指南.
2. 在Amazon包含 CodeCommit 创建存储库，创建 IAM 角色，并将上一步中创建的策略附加到该角色。想要了解有关信息，请参阅步骤 2: 在 AccountA 中创建用于存储库访问的角色中的CodeCommit 用户指南.
3. 在笔记本实例中创建一个配置文件，该配置文件使用上一步中创建的角色：
   a. 打开笔记本实例。
   b. 在笔记本实例中打开终端。
   c. 通过在终端中键入以下内容，编辑新配置文件：

   ```
   vi /home/ec2-user/.aws/config
   ```

   d. 使用以下配置文件信息编辑该文件：

   ```
   [profile CrossAccountAccessProfile]
   region = us-west-2
   role_arn =
    arn:aws:iam::CodeCommitAccount:role/CrossAccountRepositoryContributorRole
   credential_source=Ec2InstanceMetadata
   output = json
   ```

   其中*CodeCommitAccount*是包含 CodeCommit 存储库*CrossAccountAccessProfile*是新配置文件的名称，*CrossAccountRepositoryContributorRole*是您在上一步中创建的角色的名称。
4. 在笔记本实例上，配置 Git 以使用上一步中创建的配置文件：
   a. 打开笔记本实例。
   b. 在笔记本实例中打开终端。
   c. 通过在终端中键入以下内容，编辑 Git 配置文件：

```
vi /home/ec2-user/.gitconfig
```

d. 使用以下配置文件信息编辑该文件：

```
[credential]
        helper = !aws codecommit credential-helper --
profile CrossAccountAccessProfile $@
        UseHttpPath = true
```

其中 *CrossAccountAccessProfile* 是您在上一步中创建的配置文件的名称。

在笔记本实例中使用 Git 存储库

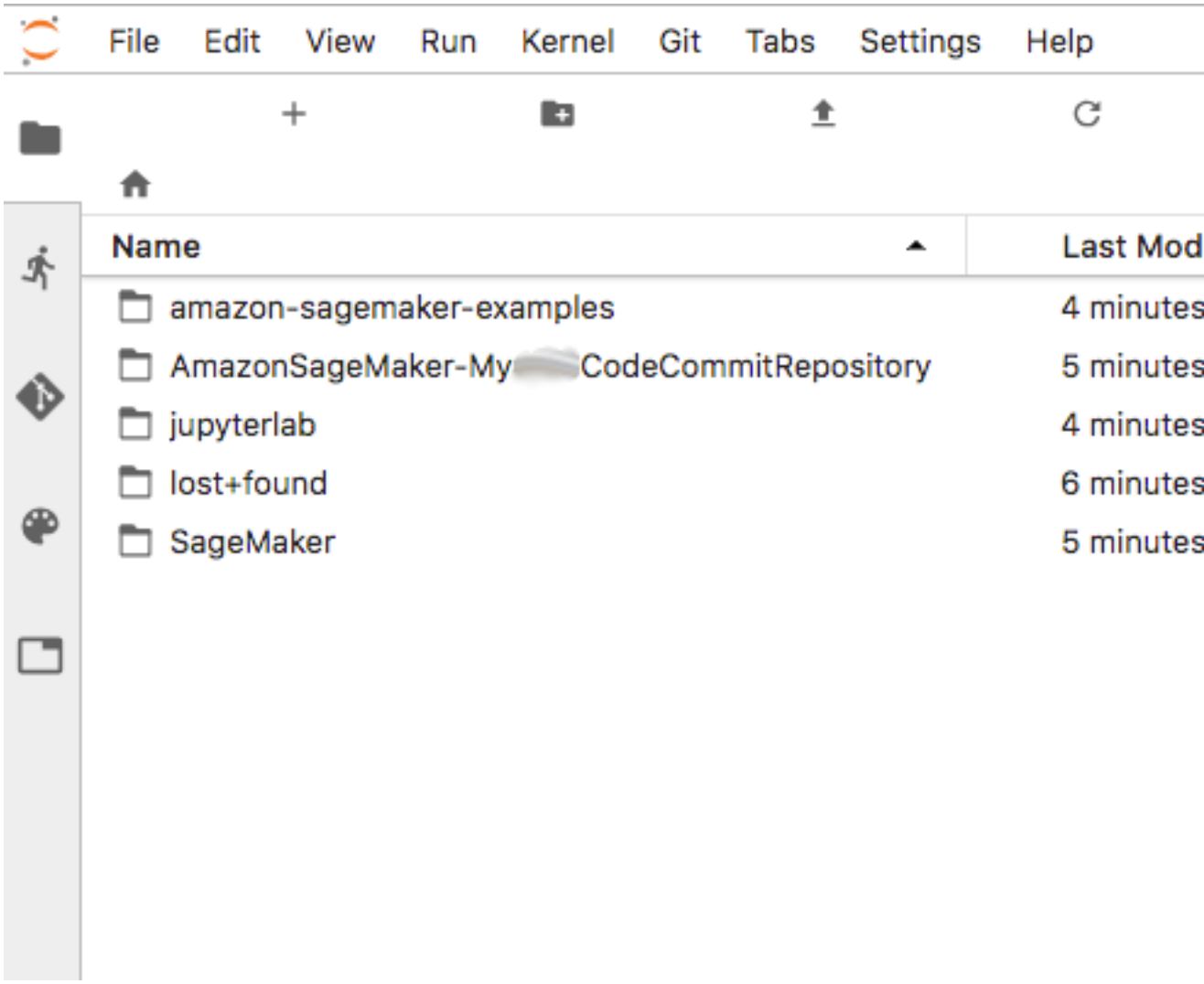
当您打开关联有 Git 存储库的笔记本实例时，它会在默认存储库中打开，默认存储库直接安装在您笔记本实例的 `/home/ec2-user/SageMaker` 下。您可以打开和创建笔记本，并可在笔记本单元中手动运行 Git 命令。例如：

```
!git pull origin master
```

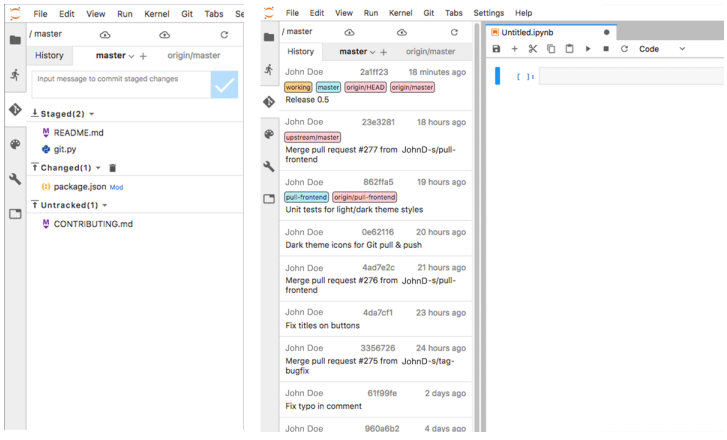
要打开任何额外的存储库，请从文件夹向上导航一级。额外的存储库也作为目录安装在 `/home/ec2-user/SageMaker` 下。

如果你用 JupyterLab 接口，jupyter-git 扩展已安装并可供使用。有关适用于 JupyterLab 的 jupyter-git 扩展的信息，请参阅 https://github.com/jupyterlab/jupyterlab-git。

当您在 JupyterLab 中打开笔记本实例时，可以看到与其关联的存储库位于左侧菜单中：



您可以使用 jupyter-git 扩展来直观管理 git，而不是使用命令行：



笔记本实例元数

在创建笔记本实例时，Amazon SageMaker 在该位置的实例上创建 JSON 文件`/opt/ml/metadata/resource-metadata.json`其中包含`ResourceName`和`ResourceArn`的笔记本实例。您可以从笔记本实例上的任意位置中访问该元数据，包括在生命周期配置中。有关笔记本实例生命周期配置的信息，请参阅使用生命周期配置脚本自定义笔记本实例 (p. 152)。

`resource-metadata.json` 文件具有以下结构：

```
{
    "ResourceArn": "NotebookInstanceArn",
    "ResourceName": "NotebookInstanceName"
}
```

您可以从笔记本实例中使用该元数据以获取有关笔记本实例的其他信息。例如，以下命令获取与笔记本实例关联的标签：

```
NOTEBOOK_ARN=$(jq '.ResourceArn'
            /opt/ml/metadata/resource-metadata.json --raw-output)
aws sagemaker list-tags --resource-arn $NOTEBOOK_ARN
```

输出类似于以下内容：

```
{
    "Tags": [
        {
            "Key": "test",
            "Value": "true"
        }
    ]
}
```

监控 Amazon Jupyter 日志 CloudWatch 日志

Jupyter 日志包括重要信息，例如事件、指标和健康信息，可在运行 Amazon 时提供可操作的见解。SageMaker 笔记本。通过将 Jupyter 登录导入 CloudWatch 日志，客户可以使用 CloudWatch 使用日志来检测异常行为、设置警报并发现见解，以便保持 SageMaker 笔记本电脑运行更顺畅。即使托管笔记本的 Amazon EC2 实例无响应，您也可以访问日志，并使用日志对无响应的笔记本进行故障排除。预签名 URL 中的敏感信息（例如 Amazon 账户 ID、私有密钥和身份验证令牌）将被删除，以便客户可以共享日志而不会泄露私人信息。

查看笔记本实例的 Jupyter 日志：

1. 登录到Amazon Web Services Management Console然后打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 选择 Notebook instance (笔记本实例)。
3. 在笔记本实例列表中，选择笔记本实例，选择要查看其 Jupyter 日志的笔记本实例。名称.

   这会将您带到该笔记本实例的详细信息页面。
4. 在笔记本实例详细信息页面的 Monitor (监控) 下，选择 View logs (查看日志)。
5. 在 CloudWatch 控制台中，选择笔记本实例的日志流。它的名称的格式为 *NotebookInstanceName*/jupyter.log。

有关监控的更多信息 CloudWatch SageMaker 的日志，请参阅使用 Amazon CloudWatch 记录 Amazon SageMaker 事件 (p. 2820).

使用亚马逊自动开发模型 SageMaker Autopilot

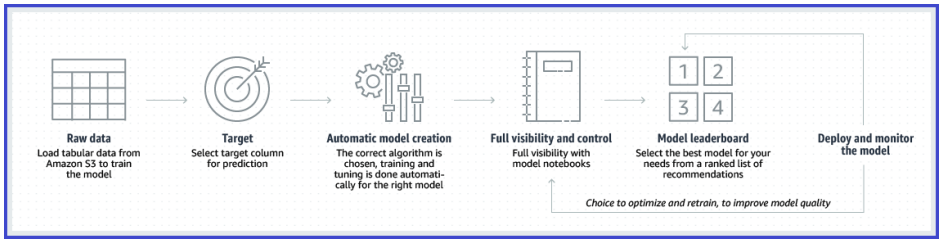
亚马逊 SageMaker Autopilot 是一种功能集，可自动执行自动机器学习 (AutoML) 过程的关键任务。它会探索您的数据，选择与问题类型相关的算法，并准备数据以便于模型训练和调整。Autopilot 在适当时将交叉验证重采样程序自动应用于所有候选算法，以测试其预测未受过训练的数据的能力。它还生成衡量标准来评估其机器学习模型候选项的预测质量。它通过自动化构成 AutoML 流程的这些关键任务来简化机器学习体验。它对所有通过性能测试的优化模型进行了排名。它会找到性能最佳的模型，您可以在通常需要的时间内部署该模型。

您可以通过不同的方式使用 Autopilot：在自动驾驶仪（因此得名）上或在各种程度的人工指导下（无需通过亚马逊进行编码）。 SageMaker Studio，或者使用其中一个代码Amazon开发工具包。Autopilot 目前支持回归以及二进制和多类分类问题类型。它支持格式化为 CSV 或 Paradic 文件的表格数据，其中每列都包含具有特定数据类型的要素，而每行都包含观测值。接受的列数据类型包括由逗号分隔数字字符串组成的数字、分类、文本和时间序列。Autopilot 支持在高达数百 GB 的大型数据集上构建机器学习模型。

Autopilot 还有助于解释模型如何使用为亚马逊开发的功能归因方法进行预测。 SageMaker 澄清。Autopilot 会自动生成一份报告，指示每个功能对于最佳候选人所做预测的重要性。这种可解释性功能可以使机器学习模型更容易理解Amazon客户。生成的模型治理报告可用于向风险和合规团队以及外部监管机构提供信息。

您可以完全了解数据处理方式以及为每个测试候选项选择、训练和调整模型的方式。这是由 Autopilot 为每个试用生成的笔记本提供的，其中包含用于探索数据和查找最佳候选人的代码。这些笔记本还提供教育工具来帮助您了解和进行自己的 ML 实验。您可以通过检查 Autopilot 公开的各种数据探索和候选定义笔记本来了解实验中各种输入和取舍的影响。您还可以通过自己修改笔记本并重新运行笔记本电脑，对表现较高的候选人进行进一步的实验。

下图概述了 Autopilot 管理的 AutoML 流程的主要任务。



使用 Amazon SageMaker，您可以按实际用量付费。您为其中的底层计算和存储资源付费 SageMaker 或者其他Amazon服务，取决于您的使用情况。有关 SageMaker 的使用成本的更多信息，请参阅Amazon SageMaker 定价.

主题

- 开始使用 Amazon SageMaker Autopilot (p. 171)
- 创建亚马逊 SageMaker Autopilot 实验 (p. 172)
- 亚马逊 SageMaker 自动驾驶仪数据集和问题类型 (p. 177)
- 使用亚马逊进行模型支持、指标和验证 SageMaker Autopilot (p. 178)
- 亚马逊 SageMaker 部署 Autopilot 模型 (p. 180)
- 亚马逊 SageMaker Autopilot 的可解释性 (p. 182)
- 亚马逊生成的模型 SageMaker Autopilot (p. 182)

- 亚马逊 SageMaker 为管理 AutoML 任务而生成的自动驾驶仪笔记本 (p. 186)
- 在 AutoPilot 生成的容器中配置推理输出 (p. 197)
- 亚马逊 SageMaker Autopilot 配额 (p. 199)
- Amazon API 参考指南 SageMaker Autopilot (p. 201)

开始使用 Amazon SageMaker Autopilot

亚马逊 SageMaker Autopilot 提供示例、视频和教程，以便开始使用亚马逊。 SageMaker Autopilot

主题

- 示例：使用 Amazon 探索建模 SageMaker Autopilot (p. 171)
- 视频: 使用 Autopilot 自动执行和探索机器学习过程 (p. 171)
- 教程: 开始使用 Amazon SageMaker Autopilot (p. 172)

示例：使用 Amazon 探索建模 SageMaker Autopilot

亚马逊 SageMaker Autopilot 提供以下示例笔记本。

- 亚马逊的直接营销 SageMaker Autopilot：本笔记本演示了如何使用银行营销数据集预测客户是否会在银行注册定期存款。您可以在此数据集中使用 Autopilot，通过探索各种候选管道中包含的选项来获得最准确的 ML 管道。Autopilot 通过两步过程生成每个候选人。第一步对数据集执行自动实施的特征工程。第二步训练和优化算法以生成模型。该笔记本包含有关如何训练模型以及如何部署模型以使用最佳候选项执行批量推断的说明。
- 亚马逊的客户流失预测 SageMaker Autopilot：该笔记本介绍使用机器学习自动识别不满意的客户，也称为客户流失预测。该示例说明如何分析公开提供的数据集并对其执行特征工程。接下来，它展示如何通过选择性能最佳的管道以及用于训练算法的最佳超参数来优化模型。最后，它演示了如何将模型部署到托管终端节点以及如何根据实际情况评估其预测结果。但是，ML 模型很少提供完美的预测。这就是为什么该笔记本还展示在确定使用 ML 的财务结果时如何考虑预测错误的相对成本。
- 亚马逊的顶级候选人客户流失预测 SageMaker 自动驾驶仪和 Batch 转换 (Python SDK)：该笔记本还介绍使用机器学习自动识别不满意的客户，也称为客户流失预测。本笔记本演示了如何配置模型以获取推理概率、选择前 N 个模型以及在暂停测试集上进行 Batch 转换以进行评估。

  Note

  这款笔记本适用于 SageMaker Python SDK >= 1.65.1 已于 2020 年 6 月 19 日发布。
- 将您自己的数据处理代码带入亚马逊 SageMaker Autopilot：本笔记本演示了使用亚马逊时如何合并和部署自定义数据处理代码 SageMaker Autopilot. 它添加了自定义功能选择步骤，以便将不相关的变量移除到 Autopilot 作业中。然后，它展示了如何在实时终端上部署自定义处理代码和 Autopilot 生成的模型，或者用于批处理。

视频: 使用 Autopilot 自动执行和探索机器学习过程

这是一个视频系列，这些视频提供了亚马逊之旅 SageMaker 使用 Studio 的自动驾驶仪功能。它们介绍了如何启动 AutoML 作业，分析和预处理数据，如何对候选模型进行特征工程和超参数优化，以及如何可视化和比较生成的模型指标。

主题

- 使用 Amazon 启动 AutoML 作业 SageMaker Autopilot (p. 172)
- 了解 Autopilot 中自动实施的数据探究和特征工程。 (p. 172)
- 调整模型以优化性能 (p. 172)

- 选择和部署最佳模型 (p. 172)
- 亚马逊 SageMaker Autopilot 教程 (p. 172)

使用 Amazon 启动 AutoML 作业 SageMaker Autopilot

该视频介绍了如何使用 Autopilot 启动 AutoML 作业。（长度：8:41）

亚马逊 SageMaker 工作室-AutoML 与亚马逊 SageMaker 自动驾驶仪（第 1 部分）

了解 Autopilot 中自动实施的数据探究和特征工程。

该视频介绍如何查看亚马逊生成的数据探索和候选定义笔记本。 SageMaker Autopilot.（长度：10:04）

亚马逊 SageMaker 工作室-AutoML 与亚马逊 SageMaker Autopilot（第 2 部分）

调整模型以优化性能

该视频介绍了如何在训练过程中使用超参数调整以优化模型性能。（长度：4:59）

Amazon SageMaker Studio-使用 Amazon 启动 AutoML SageMaker 自动驾驶仪（第 3 部分）

选择和部署最佳模型

该视频介绍了如何使用作业指标来选择最佳模型，然后介绍了如何进行部署。（长度：5:20）

Amazon SageMaker Studio-使用 Amazon 启动 AutoML SageMaker 自动驾驶仪（第 4 部分）

亚马逊 SageMaker Autopilot 教程

该视频指导您完成端到端演示，在该演示中，我们先使用亚马逊自动构建一个二元分类模型。 SageMaker Autopilot. 我们将看到如何使用自动生成的笔记本构建和优化候选模型。我们还会查看亚马逊的最佳候选项 SageMaker 实验。最后，我们部署最佳候选项（基于 XGBoost），并使用配置数据捕获。 SageMaker 模型监控器。

SageMaker 上使用 AutoML 进行端到端演示

教程: 开始使用 Amazon SageMaker Autopilot

Autopilot 入门教程演示如何在无需编写代码的情况下自动创建机器学习模型。它们向您展示 Autopilot 如何通过帮助您探索数据并尝试不同的算法来简化机器学习体验。使用 AutoML 功能为问题类型构建最佳的机器学习模型，同时允许完全控制和可见性。

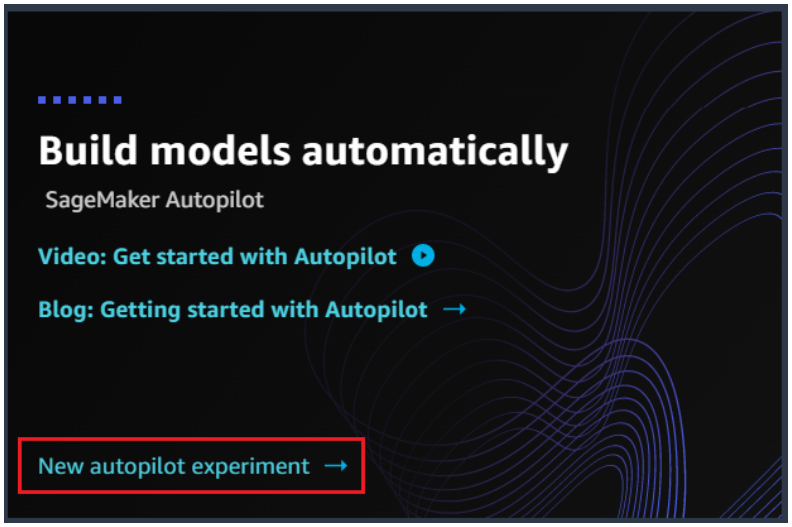
- 使用 Autopilot 自动创建机器学习模型：在本教程中，您承担在银行工作的开发人员的角色。您被要求开发一种机器学习模型，以预测客户是否会注册存单 (CD)。这是一个二元分类问题。您使用市场营销数据集对该模型进行训练，该数据集包含有关客户人口统计数据、市场营销活动响应情况和外部因素的信息。

创建亚马逊 SageMaker Autopilot 实验

创建 Autopilot 作业作为试点实验时，Autopilot 会分析数据并创建包含候选模型定义的笔记本。如果您选择运行完整实验选项，Autopilot 还会代表您训练和优化这些模型。您可以在实验运行时查看统计信息。运行后，您可以比较试验并深入研究详细信息。

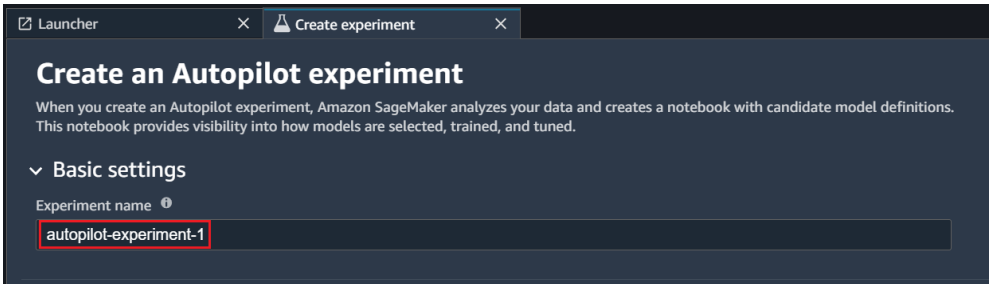
以下说明向您展示了如何创建 Amazon SageMaker Autopilot 实验。您将命名它，提供输入和输出数据的位置，并指定要预测的目标数据。另外，您还可以指定要解决的机器学习问题的类型。

1. 打开亚马逊 SageMaker 工作室并登录。如果需要有关启动 Studio 的信息，请参阅使用 Amazon SageMaker Studio 启动器 (p. 80).
2. 选择新 Autopilot 实验选项来自自动构建模型。

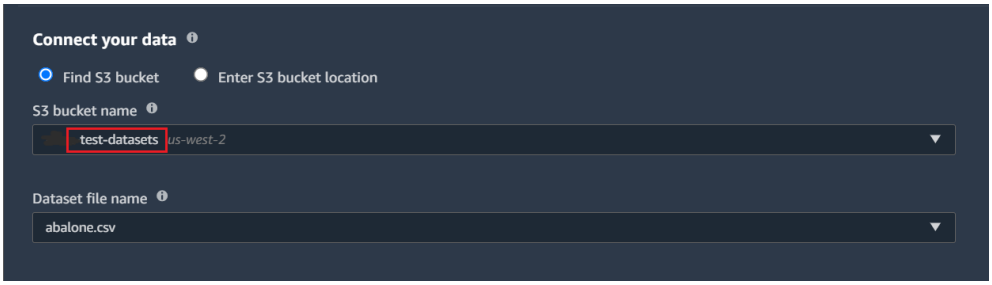


3. 在基本设置的 部分创建 Autopilot 实验页：

   - 实验名称— 在当前账户中必须是唯一的Amazon区域且最多包含 63 个字母数字字符。可以包括连字符 (-) 但不能包括空格。



   - Connect 你的数据— 提供 S3 存储桶名称和包含输入数据的数据集文件名。



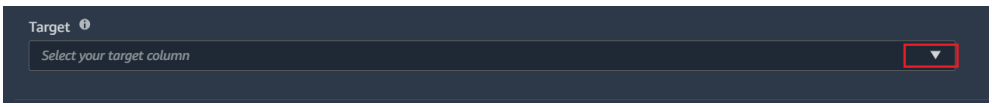
   Note

   亚马逊必须是 s3://格式的 URL SageMaker 具有写权限。S3 存储桶必须位于当前存储桶Amazon区域和文件必须为 CSV 或拼花格式，并且至少包含 500 行。

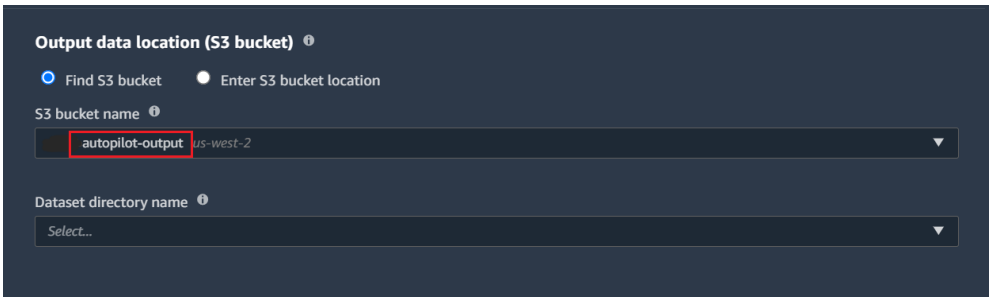
   - S3 存储桶名称— 存储桶名称在 Amazon S3 中的所有现有存储桶名称之间必须唯一。
   - S3 对象 key prefix— 存储桶中对象的文件名，包括存储桶内对象的路径。
   - S3 存储桶位置— S3 存储桶名称和 S3 对象 key prefix 的连接。

- 你的 S3 输入是清单文件吗？ — 清单文件包括带有输入数据的元数据。元数据指定数据在 Amazon S3 存储中的位置、数据的格式以及在训练模型时要使用的数据集中的哪些属性。在 Pipe 模式下流式传输标记的数据时，您可以使用清单文件作为预处理的替代方法。

- 目标— 希望模型针对的数据列名称以进行预测。



- 输出数据位置（S3 存储桶）— 要存储输出数据的 S3 存储桶和目录的名称。



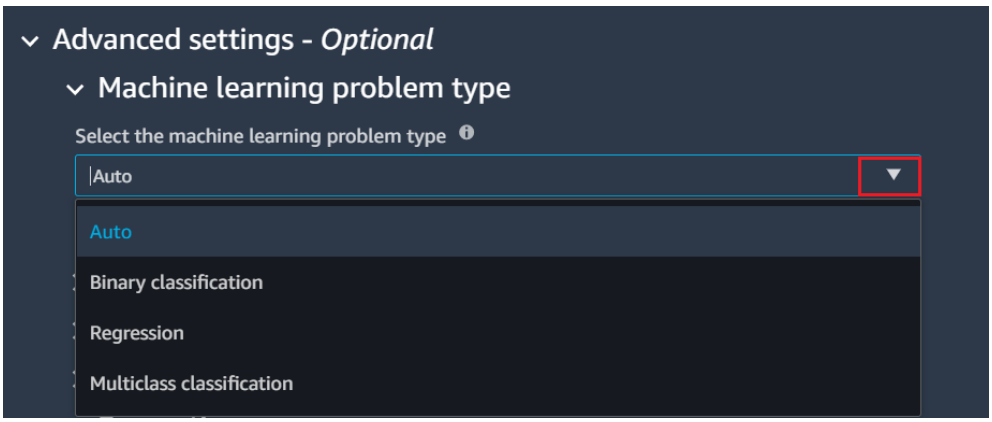
  Note

  亚马逊必须是 s3://格式的 URL SageMaker 具有写权限。S3 存储桶必须位于当前存储桶 Amazon区域。

  - S3 存储桶名称— 存储桶名称在 S3 中的所有现有存储桶名称之间必须唯一。
  - S3 对象 key prefix— 存储桶中对象的文件名，包括存储桶内对象的路径。
  - S3 存储桶位置— S3 存储桶名称和 S3 对象 key prefix 的连接。

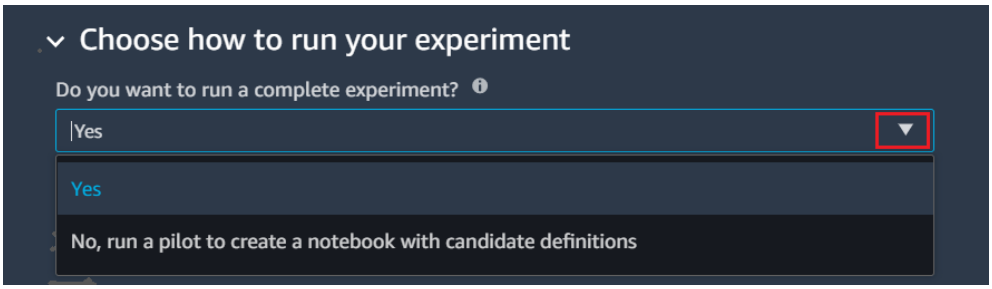
4. 高级设置-可选— Autopilot 提供额外的控件，允许您手动设置实验参数。

   - 机器学习问题类型— AutoPilot 可以自动选择机器学习问题类型。如果你更喜欢手动指定它，请使用选择机器学习问题类型下拉菜单。



   - 汽车— Autopilot 从要预测的属性值推断出问题类型。在某些情况下， SageMaker 无法准确推断。如果发生这种情况，您必须提供值才能使作业成功。
   - 二进制分类— 二元分类是一种受监督的学习类型，可根据个体的属性将个体分配给两个预定义且互斥的类别之一。例如，基于确定某人是否患有疾病的诊断测试结果进行医疗诊断。
   - 回归— 回归根据一个或多个与其相关的变量或属性来估计因果目标变量的值。例如，房价基于功能，例如平方英尺和浴室数量。

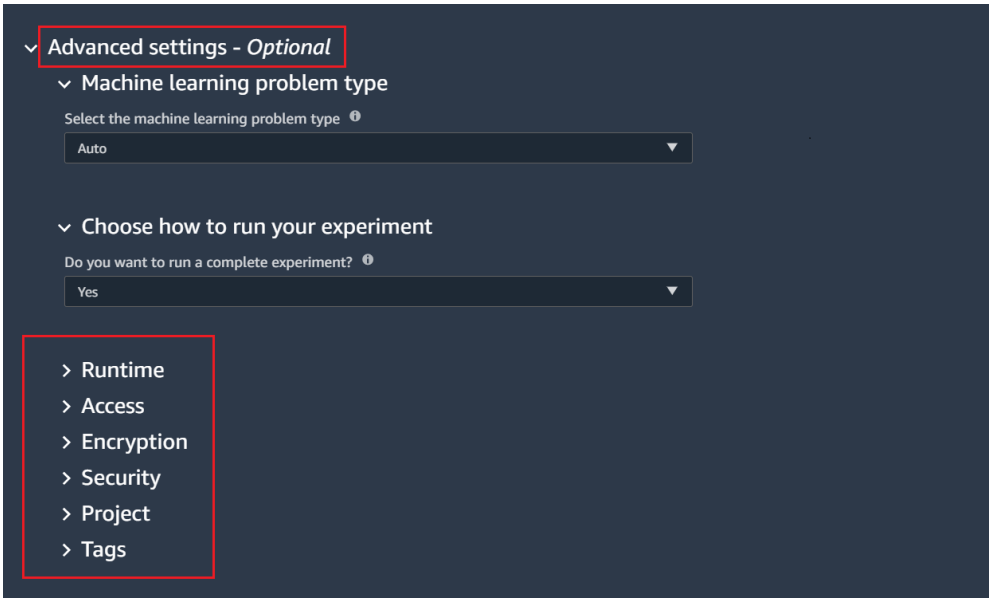
  - 多类别分类— 多类别分类是一种受监督的学习类型，可根据个体的属性将个体分配给多个类别之一。例如，预测与文本文档最相关的主题，如政治、金融或哲学。
- 选择如何运行实验— 你可以指定如何运行你的实验。



如果选择是，Autopilot 会生成一个模型和统计数据，您可以在实验运行时实时查看这些数据。在实验完成后，您可以查看试验，按客观指标进行排序，并部署模型以在其他环境中使用。

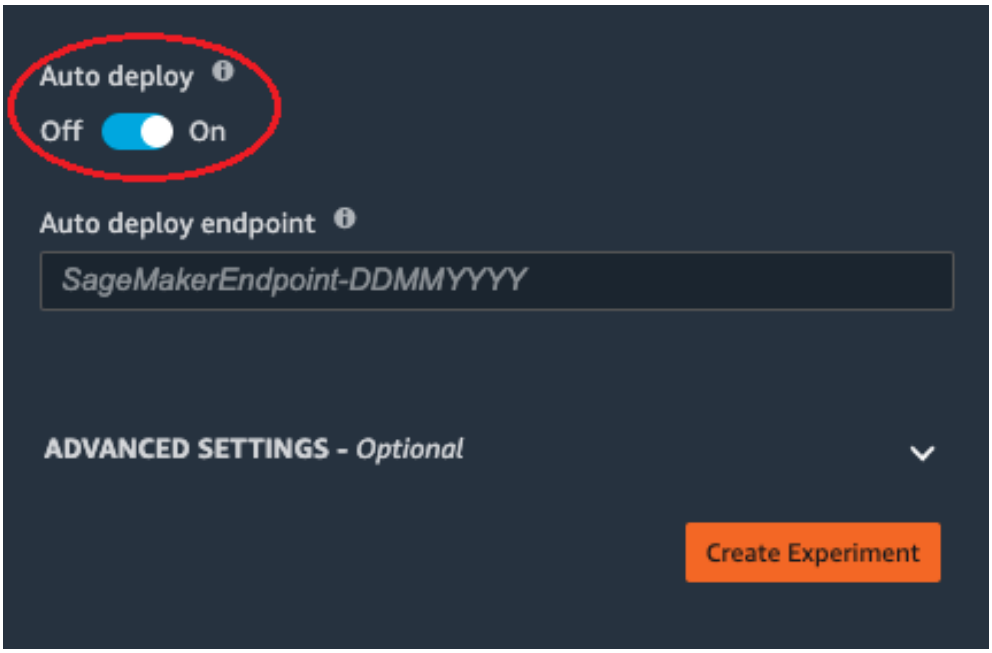
如果选择否，而不是运行整个工作流程，而是在生成具有候选定义的笔记本之后，Autopilot 会停止运行。候选项是数据预处理器、算法和算法参数设置的组合。您可以将笔记本用作起点来指导您自己的模型训练/调整过程。笔记本的突出显示部分突出显示了解释典型更改，例如更改实例类型和集群大小。

- 借助其他高级设置，您可以指定运行时限制和用于访问的 IAM 角色。您还可以指定加密密钥，以及是否使用虚拟私有云 (VPC) 来实现安全性、项目标签和标签。



工具提示中提供了每个高级设置的信息。

- 要自动将自动驾驶实验中的最佳模型部署到端点，请接受默认值自动部署值在 上创建实验时。

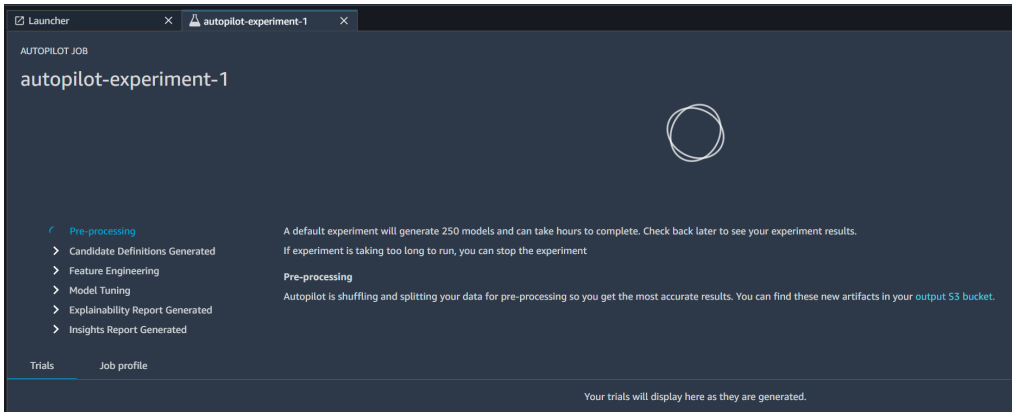


Note

如果默认资源配额或区域中终端节点实例的客户配额太有限，则自动部署将失败。目前，您需要至少有两个 ml.m5.2xlarge 实例。欧洲-北 1 区域（斯德哥尔摩）不符合此要求。此区域支持的实例类型列于欧洲（斯德哥尔摩）中的 SageMaker 实例类型 eu-north-1. 如果遇到此问题，您可以请求提高服务限制 SageMaker 通过遵循过程在支持的区域和配额 (p. 28). 在案例详细信面板中，选择SageMaker 端节点(对于)限制类型. 适用于请求 1，请选择：

- 区域：欧洲（斯德哥尔摩）
- 资源类型：SageMaker 托管
- 限制：ml.m5.2xlarge（至少）
- 新限制值：2

5. Select创建实验. 自动驾驶仪提供了实验过程中的状态。



Note

为避免产生不必要的费用：如果部署不再需要的模型，请删除在该部署期间创建的终端节点和资源。有关按区域划分的定价实例的信息，请访亚马逊 SageMaker 定价.

亚马逊 SageMaker 自动驾驶仪数据集和问题类型

亚马逊 SageMaker 您可以在 Studio 中或使用 AutoML API 提供选项，指定问题类型（例如二元分类或回归），或根据您提供的数据代表您检测问题类型。Autopilot 支持表格数据，其中每列都包含具有特定数据类型的要素，而每行都包含观测值。

主题

- Autopilot 数据集、数据类型和格式 (p. 177)
- 如何指定训练和验证数据集 (p. 177)
- 亚马逊 SageMaker Autopilot 问题类型 (p. 177)

Autopilot 数据集、数据类型和格式

Autopilot 支持格式化为 CSV 文件或拼花文件的表格数据。对于表格数据，每列都包含一个具有特定数据类型的要素，每行都包含一个观测值。这两种文件格式的属性差异很大。

- CSV（逗号分隔值）是一种基于行的文件格式，它以人类可读的纯文本存储数据，这是数据交换的普遍选择，因为它们受到各种应用程序的支持。
- Parquet是一种基于列的文件格式，其中存储和处理数据比基于行的文件格式更有效率。这使得它们成为解决大数据问题的更好选择。

这些区域有：数据类型接受列包括由逗号分隔数字字符串组成的数字、分类、文本和时间序列。如果自动驾驶仪检测到它正在处理时间序列序列，它通过由tsfresh库. 该库将时间序列作为输入，并输出一个功能，例如时间序列的最高绝对值或关于自相关的描述性统计数据。然后，这些输出的功能将用作三种问题类型之一的输入。

Autopilot 支持在高达数百 GB 的大型数据集上构建机器学习模型。有关输入数据集的默认资源限制以及如何提高此限制的详细信息，请参阅。亚马逊 SageMaker Autopilot 配额 (p. 199)

如何指定训练和验证数据集

使用时`CreateAutoMLJob`要创建 AutoML 作业，您必须使用`InputDataConfig`参数来指定`AutoMLChannel`提供输入数据源的对象。ELECT`AutoMLChannel`有`ChannelType`，可以将其设置为`training`要么`validation`指定构建机器学习模型时如何使用数据的值。必须至少提供一个数据源，最多允许两个数据源：一个用于训练数据，另一个用于验证数据。

如何将数据拆分为训练和验证数据集取决于您是有一个还是两个数据源。

- 如果你只有一个数据源，`ChannelType`设置为`training`默认情况下，必须具有此值。
  - 如果`ValidationFraction`中的值`AutoMLDataSplitConfig`未设置，默认情况下，该源的 0.2 (20%) 数据用于验证。
  - 如果`ValidationFraction`设置为介于 0 和 1 之间的值，则根据指定的值拆分数据集，其中值指定用于验证的数据集的分数。
- 如果您两个数据源，`ChannelType`其中一个`AutoMLChannel`必须将对象设置为`training`，默认值。这些区域有：`ChannelType`的另一个数据源必须设置为`validation`. 这两个数据源必须具有相同的格式，CSV 或 Picar，以及相同的架构。你不能为`ValidationFraction`在这种情况下，因为来自每个来源的所有数据都用于培训或验证。设置此值会导致错误。

亚马逊 SageMaker Autopilot 问题类型

您可以使用 `CreateAutoPilot.ProblemType` 参数设置问题类型。这限制了 Autopilot 尝试的预处理和算法种类。作业完成后，如果您设置了`CreateAutoPilot.ProblemType`，然

后ResolvedAttribute.ProblemType匹配ProblemType你设置了。如果你把它保留空白（或者null），ProblemType这将是 Autopilot 代表您决定的任何内容。

Note

在某些情况下，自动驾驶仪无法推断ProblemType在这种情况下，您必须提供值才能使作业成功。

您的问题类型选项如下所示：

主题

- 回归 (p. 178)
- 二进制分类 (p. 178)
- 多类别分类 (p. 178)

回归

回归根据一个或多个与其相关的其他变量或属性来估计因果目标变量的值。一个例子是预测房屋价格所使用的特征，如浴室和卧室的数量、房屋和花园的平方英尺数。回归分析可以创建一个模型，该模型将其中一个或多个特征作为输入并预测房屋价格。

二进制分类

二元分类是一种受监督的学习类型，可根据个体的属性将个体分配给两个预定义且互斥的类别之一。它受到监督，因为模型是使用示例进行训练的，其中为属性提供了正确标记的对象。基于诊断测试的结果对个人是否患有疾病的医学诊断是二元分类的一个示例。

多类别分类

多类别分类是一种受监督的学习类型，可根据个体的属性将个体分配给多个类别之一。它受到监督，因为模型是使用示例进行训练的，其中为属性提供了正确标记的对象。一个例子是预测与文本文档最相关的主题。文档可以分类为如宗教、政治或金融，或者是其他若干个预定义的主题类别之一。

使用亚马逊进行模型支持、指标和验证 SageMaker Autopilot

亚马逊 SageMaker Autopilot 支持三种类型的机器学习算法，以解决机器学习问题、报告质量和客观指标，并在需要时自动使用交叉验证。

主题

- 支持 Autopilot 算法 (p. 178)
- Autopilot 候选项指标 (p. 179)
- 自动驾驶仪交叉验证 (p. 180)

支持 Autopilot 算法

以下是 Autopilot 支持的三种类型的机器学习算法：

- 线性学习器算法 (p. 1657)— 用于求解分类或回归问题的指导式学习算法。
- XGBoost 算法 (p. 1731)— 一种指导式学习算法，它尝试将一组较简单和较弱模型的一系列估计值结合在一起，从而准确地预测目标变量。

- 深度学习算法 — 多层感知器 (MLP) 和前馈人工神经网络。处理不可线性分离的数据。

  Note

  您无需指定用于处理机器学习问题的算法。Autopilot 会自动选择适当的算法来训练。

Autopilot 候选项指标

亚马逊 SageMaker Autopilot 生成衡量机器学习模型候选项的预测质量的指标。为候选人计算的指标是使用数组指定的`MetricDatum`类型。以下列表包含当前可用的指标的名称。

- `Accuracy`：分类正确的商品数量与分类商品总数之比。它用于二进制分类和多类分类。准确度衡量预测的班级值与实际值的接近程度。准确度值介于零 (0) 和一 (1) 之间。值 1 表示完美的准确性，0 表示完美的不准确性。
- `AUC`：曲线 (AUC) 指标下的面积用于通过返回概率的算法比较和评估二进制分类，例如逻辑回归。为了将概率映射到分类中，请将它们与阈值进行比较。相关曲线是接收器的运行特征曲线（ROC 曲线）。ROC 曲线将预测（或召回）的真正正率（TPR）与假阳性率 (FPR) 作为阈值的函数绘制，超过该阈值的预测被视为正值。提高阈值会减少误报，但是误报更多。AUC 是该接收器运行特征曲线下的区域。因此，AUC 提供了对所有可能分类阈值的模型性能的综合度量。AUC 分数在 0 和 1 之间不等。分数为 1 表示完美的准确性，分数为一半（0.5）表示预测不比随机分类器更好。
- `BalancedAccuracy`：平衡准确度是衡量准确预测与所有预测的比率的指标。该比率是在按正值 (P) 和负值 (N) 的总数对真正值 (TP) 和真负值 (TN) 进行标准化后计算的。它用于二进制和多类分类，定义如下：0.5* ((TP/P) + (TN/N))，值在 0 到 1 之间。当不平衡数据集中的正值或负值的数量相互差异很大时，均衡准确度可以更好地衡量准确度。例如，只有 1% 的电子邮件是垃圾邮件。
- `F1`：F1 分数是精度和召回率的调和平均值。它用于二进制分类，分为传统上称为正和负的类。据说预测与实际（正确）等级匹配时是真的，如果不匹配，则是错误的。精度是真正的正面预测与所有正面预测的比率，它包括数据集中的误报。当预测积极等级时，精确度量预测的质量。召回（或敏感度）是真正的积极预测与所有实际积极实例的比率。回想衡量模型预测数据集中实际班级成员的完整程度。标准 F1 分数重量精确度并且能够同样召回。确定哪个指标至关重要通常取决于问题的具体方面。F1 分数在 0 和 1 之间不等。分数为 1 表示尽可能最好的表现，0 表示最差的表现。
- `F1macro`：F1Macroo 分数将 F1 评分应用于多类分类问题。它通过计算精度和召回率来实现这一目标，然后用他们的谐波均值计算每个班级的 F1 分数。最后，F1Macroo 将个人分数平均值以获得 F1Macroo 分数。F1Mack 分数在 0 和 1 之间有所不同。分数为 1 表示尽可能最好的表现，0 表示最差的表现。
- `LogLoss`：对数损失也称为交叉熵损失，是用于评估概率输出质量的指标，而不是输出本身。它用于二进制和多类分类、神经网络，也是逻辑回归的成本函数。日志丢失是一个重要的指标，用于指示模型何时以高概率进行错误的预测。值的范围从 0 到无限大。值 0 表示完美预测数据的模型。
- `MAE`：平均绝对误差 (MAE) 是衡量预测值和实际值之间的差异的度量，平均值与所有值相比。它通常用于回归分析来了解模型预测误差。在线性回归的情况下，MAE 表示从预测线到实际值的平均距离。MAE 被定义为绝对误差之和除以观测值数。值范围从 0 到无穷大，数字越小，表示模型更适合数据。
- `MSE`：均方误差 (MSE) 是预测值和实际值之间的平方差的平均值。它用于回归。MSE 值总是积极的。模型在预测实际值方面越好，MSE 值就越小。
- `Precision`：精确度衡量算法在其识别的所有正值中预测真正阳性（TP）的程度。其定义如下所示：精度 = TP/ (TP+FP)，值范围从零 (0) 到 1 (1)，用于二进制分类。当误报的成本很高时，精度是一个重要的指标。例如，假阳性的成本非常高，如果被错误地认为飞机安全系统可以安全飞行。误报 (FP) 反映了数据中实际上是负面的积极预测。
- `PrecisionMacro`：精度宏计算多类分类问题的精度。它通过计算每个班级的精度并平均分数以获得多个班级的精度来实现这一目标。 PrecisionMacro 分数范围从零 (0) 到一 (1)。较高的分数反映了该模型在其识别的所有积极值中预测真实阳性（TP）的能力，以及多个班级的平均值。
- `R2`R:^{2}也称为确定系数，用于量化模型可以解释依赖变量方差的程度。值范围从 1 (1) 到负值 (-1)。数字越高表示解释的变异比例越高，接近零 (0) 的数字表示模型能够解释的相关变量很少。它用于回归。负值表示拟合不佳，且模型的表现超过了常量函数，在线性回归的情况下，常量函数的表现优于水平线。
- `Recall`：回想衡量算法在多大程度上正确预测数据集中的所有真正阳性 (TP)。真正的积极预测是积极的预测，也是数据中的实际正值。召回的定义如下：召回 = TP/ (TP+FN)，值介于 0 到 1 之间。分数越高反

映了模型更好地预测数据中真正阳性 (TP) 的能力，并在二进制分类中使用。当查找所有真正阳性非常重要时，召回是一个重要的指标，例如癌症检测的情况。误报 (FP) 反映了数据中实际上是负面的积极预测。仅衡量召回通常是不够的，因为将每项产出预测为真正的正值将产生完美的召回分数。

- `RecallMacro`: 该 RecallMacro 通过计算每个班级的召回和平均分数以获得多个班级的召回来计算多类分类问题的召回情况。 RecallMacro 分数范围是 0 到 1。较高的分数反映了模型在数据集中预测真正阳性 (TP) 的能力，而真正的正反映了积极预测，该预测也是数据中的实际正值。仅衡量召回通常是不够的，因为将每项产出预测为真正的正值将产生完美的召回分数。
- `RMSE`：均方根误差 (RMSE) 测量预测值和实际值之间的平方差的平方根，在所有值上的平均值。它用于回归分析来了解模型预测误差，它是指示存在大型模型误差和异常值的重要指标。值的范围从零 (0) 到无穷大，数字越小，表示模型更适合数据。RMSE 取决于规模，不应用于比较不同大小的数据集。

为候选模型自动计算的指标取决于要解决的问题的类型。

- 回归：`MAE`、`MSE`、`R2`, RMSE
- 二进制分类：`Accuracy`、`AUC`、`BalancedAccuracy`、`F1`、`LogLoss`、`Precision`、`Recall`
- 多类别分
  类：`Accuracy`、`BalancedAccuracy`、`F1macro`、`LogLoss`、`PrecisionMacro`、`RecallMacro`

自动驾驶仪交叉验证

在需要时，Autopilot 自动使用 K 折交叉验证方法。使用此方法评估在数据集上训练的模型可以预测来自同一人群的看不见的验证数据集的值的程度。例如，在对训练实例数量有限的数据集进行训练时，此方法尤其重要。它可以防止诸如过度拟合和选择偏差等问题，因为这些问题可能会阻止模型更普遍地适用于抽样人口。

例如，波士顿房屋数据集只包含 861 个样本。如果您尝试在没有交叉验证的情况下使用此数据集构建模型来预测房屋销售价格，则可能会对不代表波士顿房屋存量的数据集进行培训。通常，您只需将数据拆分一次到训练和验证子集中。如果培训计划恰好包含主要来自郊区的数据，而这些数据并不代表城市其他地区，那么你可能会过分适应这种偏见的选择。交叉验证通过充分和随机地使用可用数据进行训练和验证，降低了这些错误的风险。

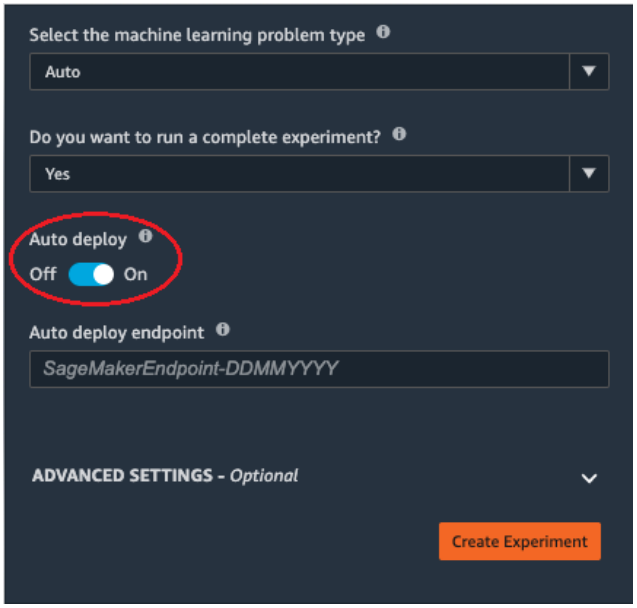
K-fold 交叉验证随机将训练数据集拆分为k同样大小的子样本或折叠。然后对模型进行训练k-1折叠并根据剩余折叠进行测试，剩余折叠保留为验证数据集。重复这个过程k次数，使用每个折叠一次作为验证数据集。Autopilot 将交叉验证方法应用于具有 50,000 个或更少训练实例的数据集。它使用了k用于建模数据集的候选算法的值为 5。在不同的拆分上训练多个模型，模型分开存储。训练过程结束后，将对每个模型的验证指标进行平均值，以生成单个估计指标。当 Autopilot 将交叉验证应用于较小的数据集时，训练时间平均增加 20%。如果你的数据集很复杂，训练时间可能会更大地增加。为了进行预测，Autopilot 使用试验中具有最佳验证指标的交叉验证模型组合。

你可以在你的中查看每折中的训练和验证指标`/aws/sagemaker/TrainingJobs` CloudWatch 日志。有关 的更多信息 CloudWatch 日志，请参阅使用 Amazon CloudWatch 记录 Amazon SageMaker 事件 (p. 2820). Autopilot 训练的模型的验证指标作为模型排行榜中的客观指标呈现。除非您另有指定，否则自动驾驶仪会对其处理的每种问题类型使用默认验证指标。有关 Autopilot 使用的指标的更多信息，请参阅`AutoMLJobObjective`. 您可以像使用交叉验证构建的 Autopilot 模型一样，部署使用交叉验证构建的 Autopilot 模型，就像 SageMaker 模型。

亚马逊 SageMaker 部署 Autopilot 模型

要部署在 Autopilot 实验中生成最佳验证指标的模型，您有多种选择。在中使用 Autopilot SageMaker Studio，您可以自动或手动部署模型。在另一个开发中工作时，您可以直接调用 Autopilot API 来部署模型。

- 自动：要自动将自动驾驶实验中的最佳模型部署到端点，请接受默认值自动部署值在 上在 SageMaker Studio 中创建实验时。

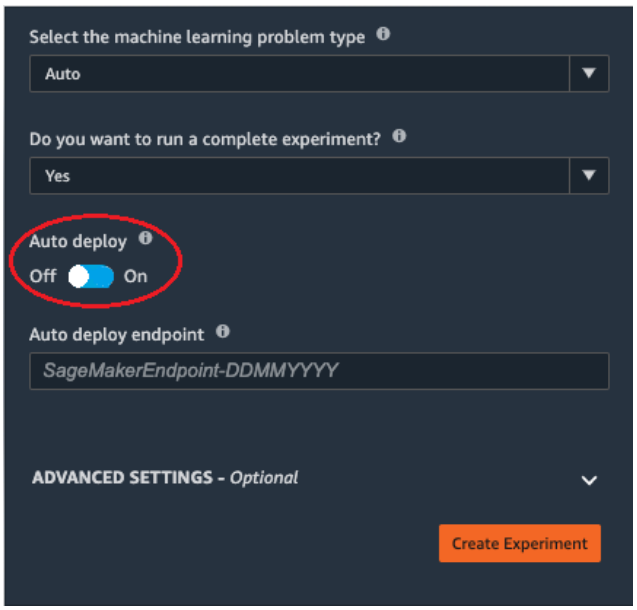


Note

如果区域中终端节点实例的默认资源配额或客户配额太有限，则自动部署将失败。目前的要求是你需要至少有两个 ml.m5.2xlarge 实例。例如，欧盟北部 1 区域（斯德哥尔摩）不符合这一要求。此区域支持的实例类型列于欧洲（斯德哥尔摩）eu-nor-1 中的 SageMaker 实例类型. 如果遇到此问题，您可以请求提高服务限制 SageMaker 通过按照中的过程操作终端实例支持的区域和配额 (p. 28). 在案例详细信面板中，选择SageMaker 端节点(对于)限制类型. 适用于请求 1Select, SELECT:

- 区域：欧洲（斯德哥尔摩）
- 资源类型：SageMaker 托管
- 限制：ml.m5.2xlarge（至少）
- 新限制值：2

- 手动方式：要手动将自动驾驶实验中的最佳模型部署到端点，请将自动部署值至Off在中创建实验时 SageMaker Studio。



- API 调用：进行以下一系列 API 调用：
  1. `CreateAutoMLJob`
  2. `DescribeAutoMLJob`
  3. `ListCandidatesForAutoMLJob`
  4. `CreateModel`
  5. `CreateEndpointConfig`
  6. `CreateEndpoint`

中的实验结果的自动部署 SageMaker 默认情况下，Studio 调用最后一个选项中列出的六个 API。有关如何创建实验的信息，请参阅创建亚马逊 SageMaker Autopilot 实验 (p. 172).

Note

为避免产生不必要的费用，请在不再需要后删除在部署模型时创建的终端节点和资源。有关按区域划分的实例定价的信息，请访问亚马逊 SageMaker 定价.

亚马逊 SageMaker Autopilot 的可解释性

亚马逊 SageMaker 自动驾驶仪使用亚马逊提供的工具 SageMaker 澄清以帮助解释机器学习 (ML) 模型如何做出预测。这些工具可以帮助机器学习建模者、开发人员和其他内部利益相关者在部署之前了解模型 您还可以使用工具调试模型部署后提供的预测。机器学习模型如何实现预测的透明度对于消费者和监管机构也至关重要，他们必须信任模型预测才能接受基于预测的决策。Autopilot 解释功能使用与模型无关的功能归因方法。你可以使用它来了解模型为什么在训练后进行预测，并使用它在推断过程中提供每个实例的解释。该实施包括可扩展且高效的实施SHAP. 这是基于来自合作博弈论领域的 Shapley 值的概念，该概念为每个特征分配特定预测的重要性值。

您可以使用解释来审计和满足监管要求、建立对模型的信任、支持人类决策以及调试和提高模型性能。

有关 Shapely 值和基线的其他信息，请参阅使用 Shapley 值的功能归因 (p. 2073)和SHAP 基准的可解释性 (p. 2073).

获取亚马逊指南 SageMaker 澄清文档，请参阅SageMaker 澄清文档指南 (p. 7).

亚马逊生成的模型 SageMaker Autopilot

此过程介绍如何查看有关 Amazon 的详细信息。 SageMaker 你已经运行的自动驾驶作业。有关 Autopilot 生成的候选模型的详细信息包括：

- 汇总 SHAP 值的图表，指示每个功能的重要性，以帮助解释模型预测。
- 包括客观指标在内的各种训练和验证指标的汇总统计数据。
- 用于训练和调整模型的超参数列表。

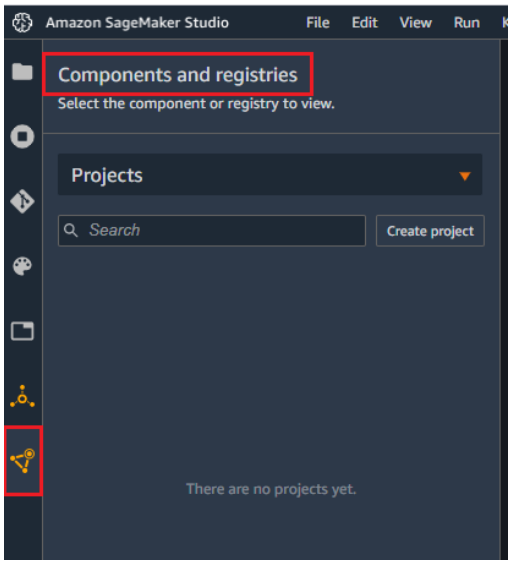
Note

本主题假定您已创建并运行了 Autopilot 实验。有关如何创建 Autopilot 实验的信息，请参阅创建亚马逊 SageMaker Autopilot 实验 (p. 172)

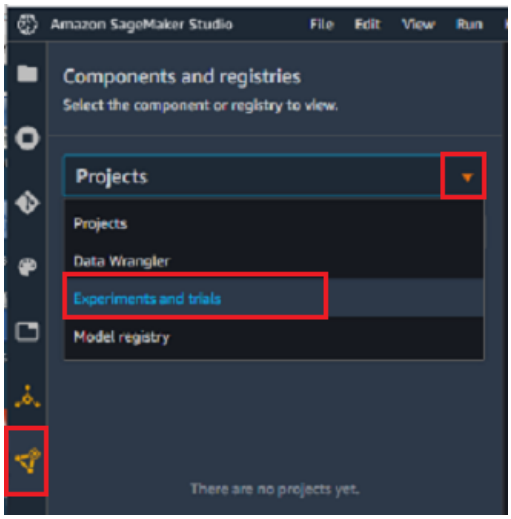
Note

要在此过程中访问功能重要性指标，您必须先选择。“文件” > “关闭”，然后从控制台重新启动 Studio。

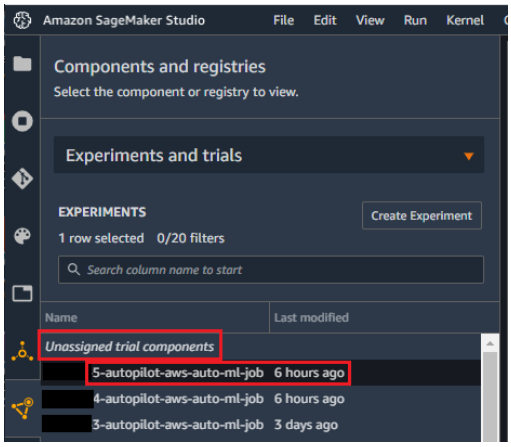
1. 运行亚马逊后查看模型详情 SageMaker AutoPilot Job，从左侧菜单中选择三角形图标以打开组件和注册表页.



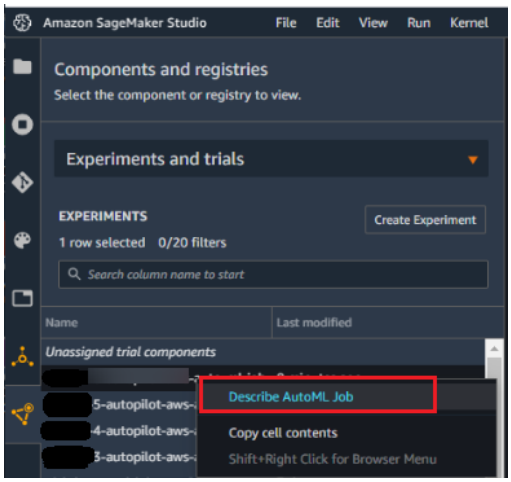
2. Select试验和试验从下拉菜单中进行。



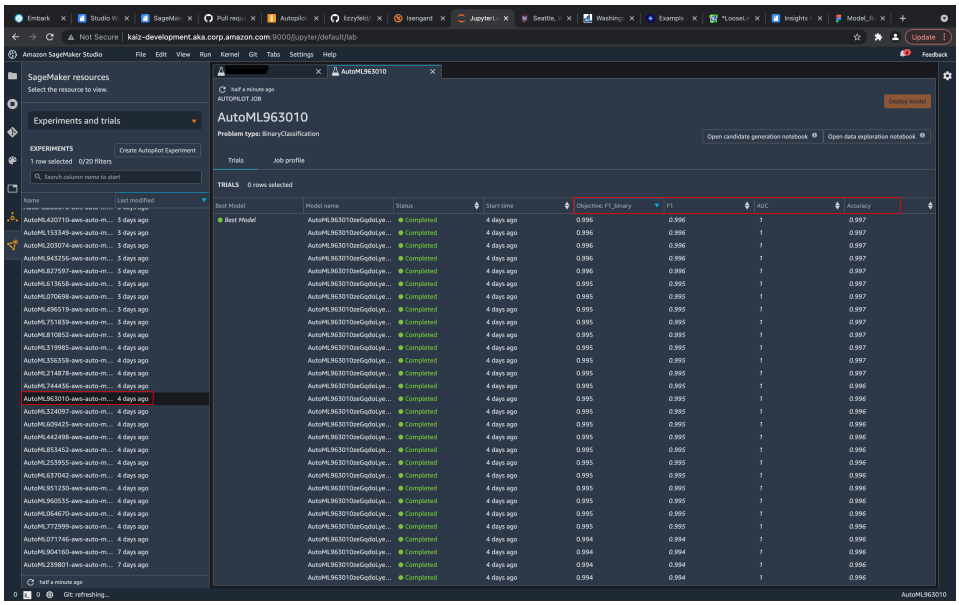
3. 找到您要在中查看的详细信息的 Autopilot 作业未分配试用组件列表。



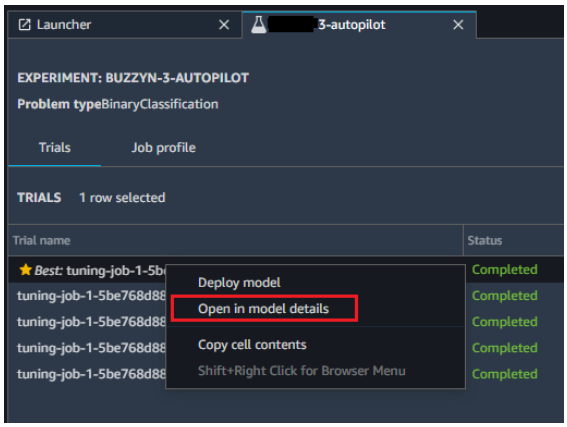
4. 右键单击 Autopilot 作业的名称，然后选择描述 AutoML Job从弹出菜单中。



要审核目标：AutoML 作业的 F1_Binary、F1、AUC 和准确度指标，请从实验面板。

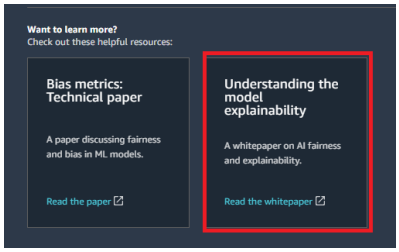


5. 要查看最佳调整作业的模型详细信息，请右键单击试用名标记为最佳位于客观指标分数最高的试验列表中的顶部。之后，选择开模型详细信息.

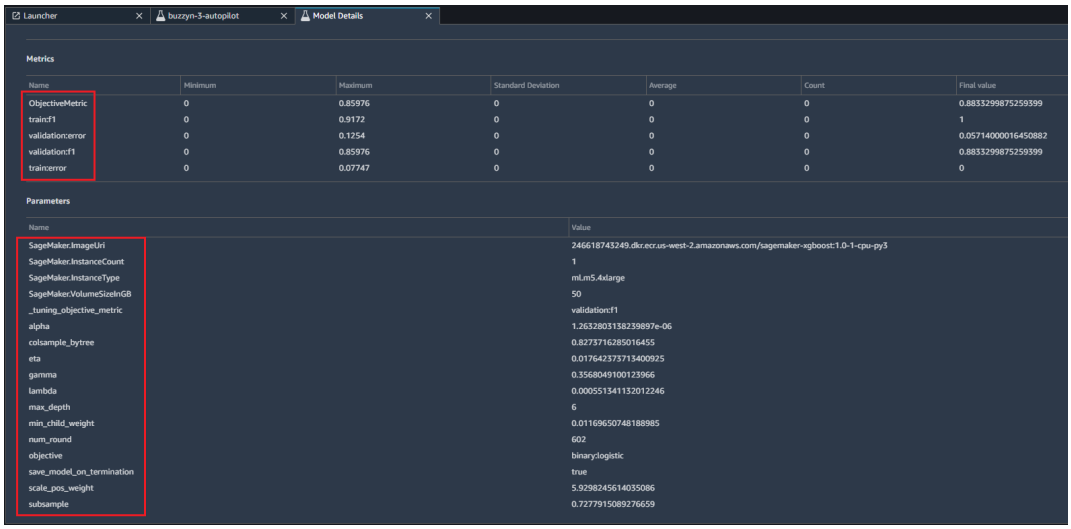


6. 表示每个要素重要性的聚合 SHAP 值的图显示在解释模型的预测部分。

7. 有关 SHAP 值如何帮助解释基于功能重要性的预测的更多信息，请向下滚动以查看指向了解模型的可解性白皮书。在中也可用亚马逊 SageMaker 澄清模型的可解释性 (p. 2072)中的主题内容 SageMaker 开发人员指南 的第一个版本。



8. 要查看目标指标的汇总统计信息、用于训练和验证的 f1 值、训练和验证错误以及用于训练和调整模型的参数值列表，请向下滚动到页面底部。



亚马逊 SageMaker 为管理 AutoML 任务而生成的自动驾驶仪笔记本

亚马逊 SageMaker Autopilot 管理自动机器学习 (AutoML) 过程中的关键任务。它们是由自动驾驶仪实施的，带有 AutoML 作业。AutoML 作业创建三个基于笔记本的报告，这些报告描述 Autopilot 为生成候选模型而需遵循的计划。候选模型由一个（管道、算法）对组成。首先，有一个数据探索笔记本，它描述了 Autopilot 从您提供的数据中了解到的信息。第二，有一个候选项定义笔记本，它使用数据相关信息来生成候选项。第三，模型见解报告可以帮助详细说明自动驾驶仪实验排行榜中最佳模型的性能特征。

主题

- 亚马逊 SageMaker Autopilot 数据探索报告 (p. 186)
- 候选项的笔记本 (p. 192)
- Autopilot 模型洞察 (p. 192)

你可以在亚马逊运行这些笔记本 SageMaker 或者如果你已经在本地安装了亚马逊 SageMaker Python 开发工具包。 您可以像共享其他笔记本一样共享笔记本 SageMaker Studio 笔记本。 系统为您创建这些笔记本以便进行实验。例如，您可以在笔记本中编辑以下项目：

- 数据使用的预处理器
- 超参数优化 (HPO) 运行的数量及其并行度
- 尝试的算法
- 用于 HPO 作业的实例类型
- 超参数范围

鼓励对候选定义笔记本进行修改以作为一种学习工具。使用此功能，您可以了解在机器学习过程中做出的决策如何影响您的结果。

> Note
>
> 在默认实例中运行笔记本电脑时，您将产生基准成本。但在从候选笔记本中运行 HPO 作业时，这些作业使用额外的计算资源，这会产生额外成本。

亚马逊 SageMaker Autopilot 数据探索报告

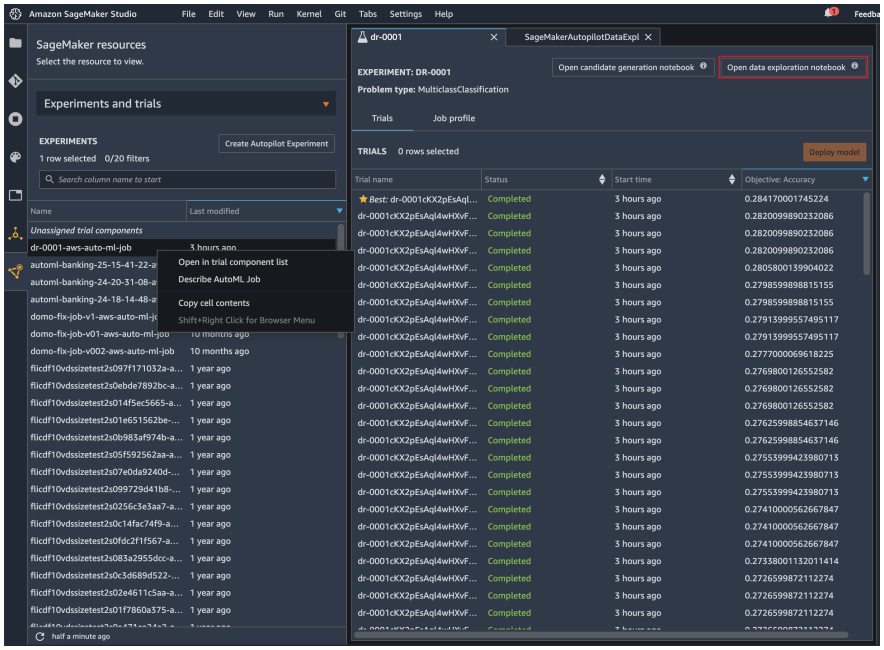
亚马逊 SageMaker Autopilot 会自动清理和预处理您的数据集。高数据质量可实现更高效的机器学习，并生成能够做出更准确预测的模型。如果没有某些领域知识的好处，客户提供的数据集存在的问题，这些问题无法自动修复。例如，回归问题的目标列中的大异常值可能会导致非异常值的预测不佳。根据建模目标，可能需要移除异常值。如果意外地将目标列包括为输入要素之一，则最终模型将验证良好，但对 future 的预测没有什么价值。为了帮助客户发现这些问题，Autopilot 提供了一份数据探索报告，其中包含对数据潜在问题的见解，并建议如何处理这些问题。

为完成管道推荐步骤的每个 Autopilot 作业生成包含报告的数据探索笔记本。该报告存储在 S3 存储桶中，可以从输出路径中访问。数据探索报告的路径通常遵循以下模式：

```
[s3 output path]/[name of the automl job]/sagemaker-automl-candidates/[name of processing
 job used for data analysis]/notebooks/SageMakerAutopilotDataExplorationNotebook.ipynb
```

数据探索笔记本的位置可以从 Autopilot API 中使用`DescribeAutoMLJob`操作响应，存储在数据探索笔记本电脑位置.

从运行自动驾驶仪时 SageMaker Studio，您可以打开描述 Autopilot 作业的 UI，然后选择开放数据探索笔记本来自自动驾驶仪职位描述页面。



数据探索报告是在训练过程开始之前根据您的数据生成的。这样，您就可以停止可能导致结果不佳或毫无意义的 AutoPilot 作业，并在重新运行 Autopilot 之前解决数据集的任何问题或改进。这使您有机会利用自己的领域专业知识来手动提高数据质量，然后再在精心策划的数据集上训练模型。

生成的数据报告只包含静态降价，可以在任何 Jupyter 环境中打开。包含报告的笔记本可以转换为其他格式，例如 PDF 或 HTML。有关转化的更多信息，请参阅使用 nbconvert 脚本将 Jupyter 笔记本转换为其他格式。.

主题

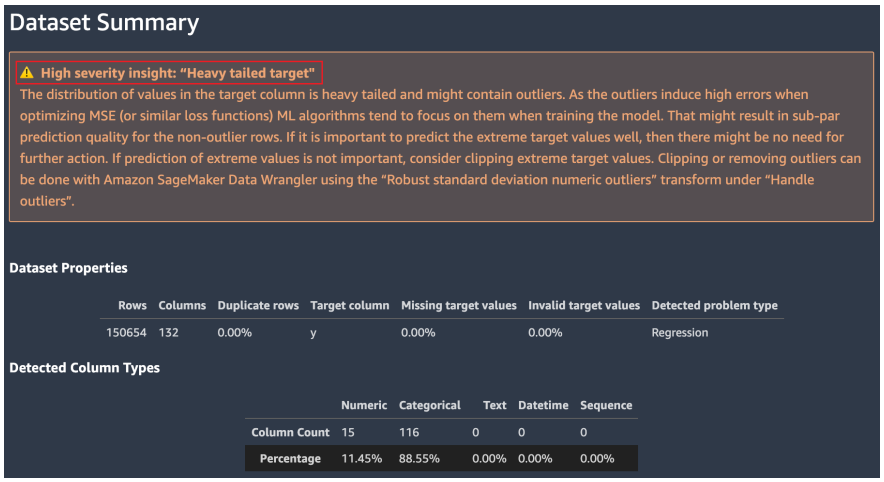
- 数据集摘要 (p. 187)
- 目标分析 (p. 188)
- 数据示例 (p. 189)
- 重复的行 (p. 190)
- 跨栏相关性 (p. 190)
- 异常行 (p. 191)
- 缺少值、基数和描述性统计 (p. 191)

数据集摘要

该数据集摘要提供了表征数据集的关键统计数据。它旨在在您的数据集出现亚马逊问题时向您提供快速警报 SageMaker 自动驾驶仪已检测到，这可能需要你的干预。这些见解显示为被归类为 “高” 或 “低” 严重性的警告。分类取决于对该问题将对模型性能产生不利影响的信心程度。

严重性高和低严重性见解以弹出窗口的形式出现在摘要中。对于大多数见解，我们提供了有关如何确认数据集存在需要注意的问题的建议。还提供了关于如何解决问题的建议。

Autopilot 提供了有关我们数据集中缺失或无效目标值的其他统计信息，以帮助您检测可能无法通过高严重性洞察捕获的其他问题。特定类型的列数意外可能表明数据集中可能缺少某些要使用的列。它还可能表明数据的准备或存储方式存在问题。修复 Autopilot 引起你注意的这些数据问题可以提高针对数据训练的机器学习模型的性能。



报告的摘要部分和其他相关章节中显示了严重性高的见解。通常根据数据报告的部分给出高度和低严重性见解的示例。

目标分析

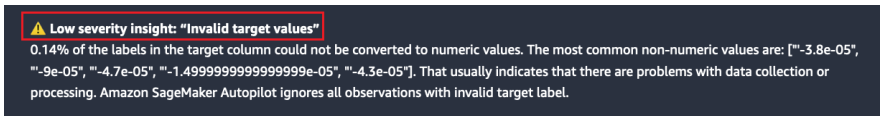
本节显示了与目标列中值分布相关的各种高严重性和低严重性见解。你应该检查目标列是否包含正确的值。目标列中的值不正确可能会导致机器学习模型不符合预期业务目的。本节介绍了严重性高和低严重性的几个数据见解。下面是几个示例。

- 异常值目标值-偏斜或不寻常的回归目标分布，例如重尾目标。
- 高或低目标基数-不常见的类别标签数量或大量的唯一分类进行分类。

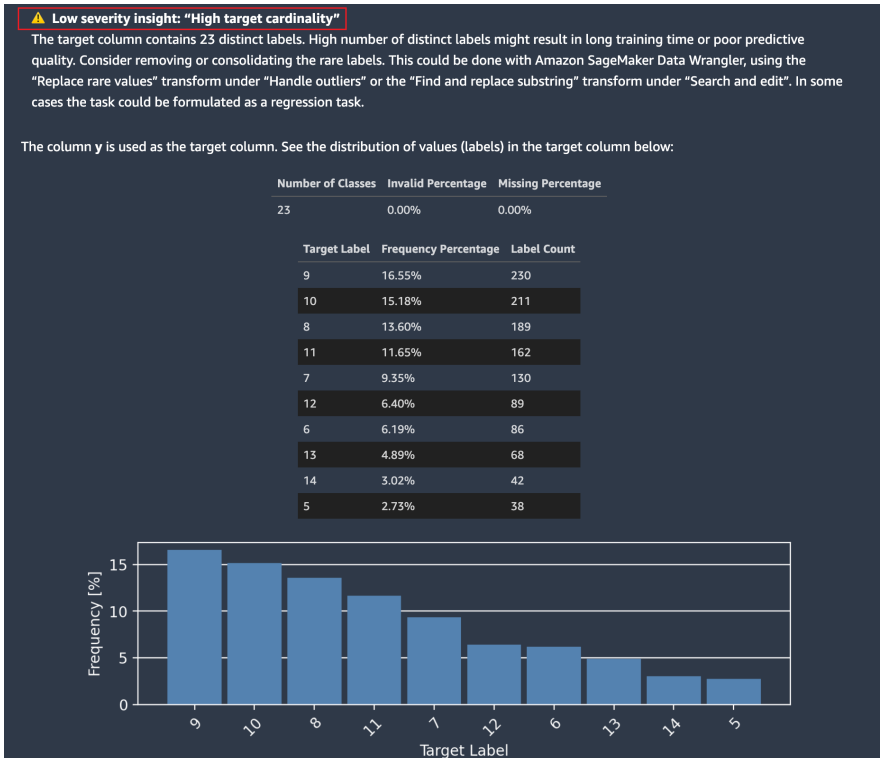
对于回归和分类问题类型，没有有效值，例如数字无穷大，NaN或者目标列中的空白空间浮出水面。根据问题类型，会显示不同的数据集统计信息。回归问题的目标列值分布允许您验证分布是否符合预期值。

以下示例显示了有关目标列值分布的 Autopilot 数据报告。

显示了有关目标值及其分布的多个统计数据。如果任何异常值、无效值或缺失百分比大于零，则会显示这些值，以便您可以调查数据包含不可用的目标值的原因。一些不可用的目标值被突出显示为严重性较低的洞察警告。在以下示例中，`符号被意外添加到目标列中，这使得无法解析目标的数字值。



为了帮助您识别有问题的值和一些受影响的行，Autopilot 提供了包含不可用或异常目标值的行的示例。将用于分类的标签分布制表和绘制，以便您也可以对其进行分析。



Note

您可以在中找到本节和其他章节中介绍的所有术语的定义定义报告笔记本底部的部分。

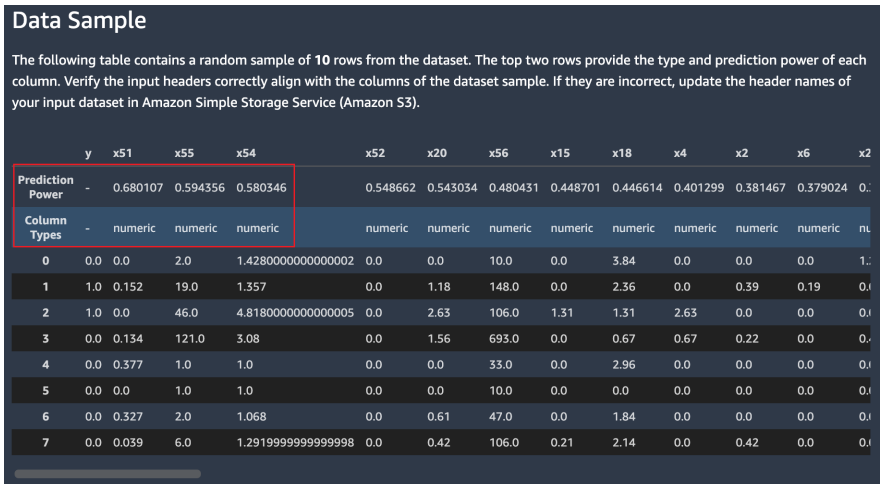
数据示例

为了进一步帮助您发现数据集的问题，亚马逊会提供实际数据样本供您检查 SageMaker Autopilot. 样本表水平滚动。它可用于验证所使用的数据集中是否存在所有必要的列。如果缺少数据列，则在导入需要调查的数据集之前可能会出现预处理问题。

亚马逊计算预测能力的度量 SageMaker Autopilot 可用于识别伪装为输入列的目标列。它有助于将注意力集中在可能很重要的专栏上，因为它们具有较高的预测能力。有关预测功率的更多信息，请参阅定义部分。

Note

除非您确定预测能力是适合自己的使用案例的合适度量，否则建议您不要使用预测能力来替代功能重要性。



Data Sample

The following table contains a random sample of **10** rows from the dataset. The top two rows provide the type and prediction power of each column. Verify the input headers correctly align with the columns of the dataset sample. If they are incorrect, update the header names of your input dataset in Amazon Simple Storage Service (Amazon S3).

	y	x51	x55	x54	x52	x20	x56	x15	x18	x4	x2	x6	x2
Prediction Power	-	0.680107	0.594356	0.580346	0.548662	0.543034	0.480431	0.448701	0.446614	0.401299	0.381467	0.379024	0.
Column Types	-	numeric	numeric	numeric	numeric	numeric	numeric	numeric	numeric	numeric	numeric	numeric	nu
0	0.0	0.0	2.0	1.4280000000000002	0.0	0.0	10.0	0.0	3.84	0.0	0.0	0.0	1.
1	1.0	0.152	19.0	1.357	0.0	1.18	148.0	0.0	2.36	0.0	0.39	0.19	0.
2	1.0	0.0	46.0	4.8180000000000005	0.0	2.63	106.0	1.31	1.31	2.63	0.0	0.0	0.
3	0.0	0.134	121.0	3.08	0.0	1.56	693.0	0.0	0.67	0.67	0.22	0.0	0.
4	0.0	0.377	1.0	1.0	0.0	0.0	33.0	0.0	2.96	0.0	0.0	0.0	0.
5	0.0	0.0	1.0	1.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.
6	0.0	0.327	2.0	1.068	0.0	0.61	47.0	0.0	1.84	0.0	0.0	0.0	0.
7	0.0	0.039	6.0	1.2919999999999998	0.0	0.42	106.0	0.21	2.14	0.0	0.42	0.0	0.

重复的行

如果数据集中存在重复行，则亚马逊 SageMaker 自动驾驶仪会显示其中的一个样本。

Note

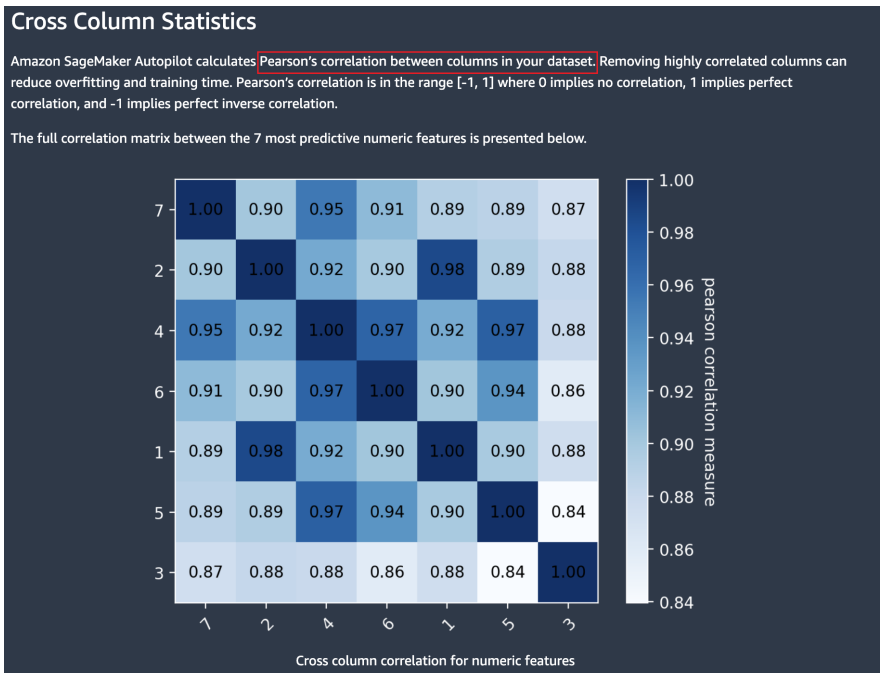
不建议在将数据集提供给 Autopilot 之前通过向上取样来平衡数据集。这可能会导致 Autopilot 训练的模型的验证分数不准确，并且生成的模型可能无法使用。

跨栏相关性

还使用标准的互相关矩阵图表显示数字列相关性。您可以使用它来减少数据集中的要素数量。较少的功能减少了模型过度拟合的可能性，并可以通过两种方式降低生产成本。它减少了所需的 Autopilot 运行时间，对于某些应用程序而言，可以使数据收集过程更便宜。

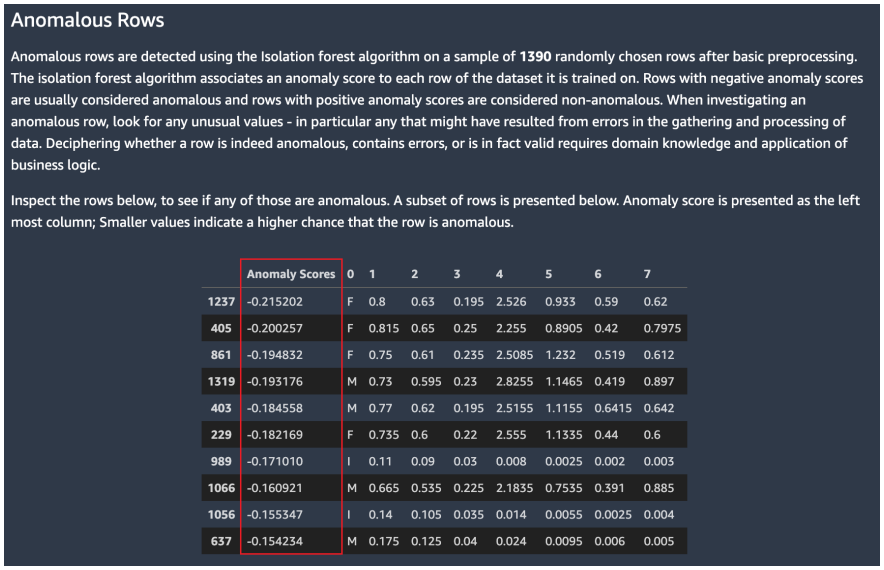
Note

接近 +1 的值和接近 -1 的值表示两个要素分别是正面和负面的两个要素高度相关性。



异常行

亚马逊 SageMaker Autopilot 可以检测数据集中的哪些行可能是异常的。然后，它会为每行分配一个异常分数。具有负异常分数的行被认为是异常的。



Anomalous Rows

Anomalous rows are detected using the Isolation forest algorithm on a sample of **1390** randomly chosen rows after basic preprocessing. The isolation forest algorithm associates an anomaly score to each row of the dataset it is trained on. Rows with negative anomaly scores are usually considered anomalous and rows with positive anomaly scores are considered non-anomalous. When investigating an anomalous row, look for any unusual values - in particular any that might have resulted from errors in the gathering and processing of data. Deciphering whether a row is indeed anomalous, contains errors, or is in fact valid requires domain knowledge and application of business logic.

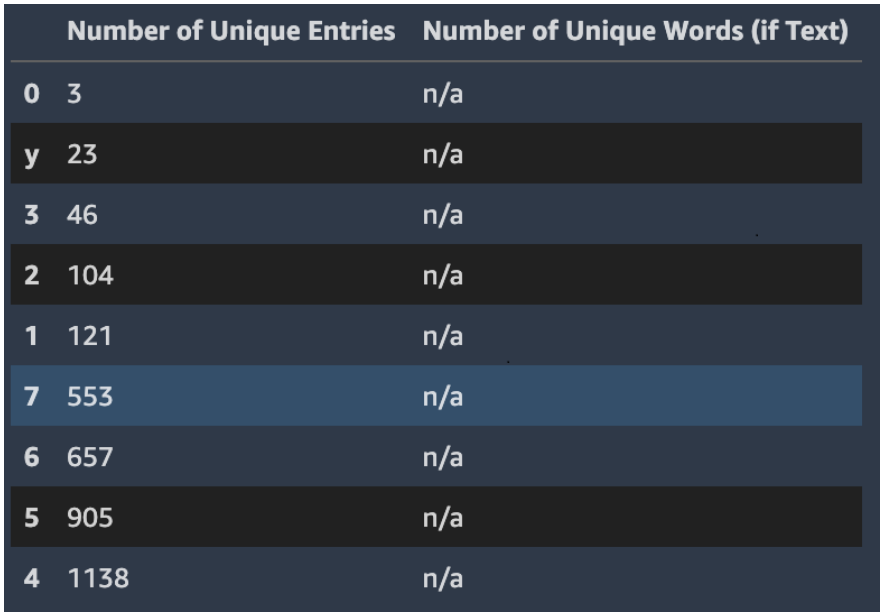
Inspect the rows below, to see if any of those are anomalous. A subset of rows is presented below. Anomaly score is presented as the left most column; Smaller values indicate a higher chance that the row is anomalous.

	Anomaly Scores	0	1	2	3	4	5	6	7
1237	-0.215202	F	0.8	0.63	0.195	2.526	0.933	0.59	0.62
405	-0.200257	F	0.815	0.65	0.25	2.255	0.8905	0.42	0.7975
861	-0.194832	F	0.75	0.61	0.235	2.5085	1.232	0.519	0.612
1319	-0.193176	M	0.73	0.595	0.23	2.8255	1.1465	0.419	0.897
403	-0.184558	M	0.77	0.62	0.195	2.5155	1.1155	0.6415	0.642
229	-0.182169	F	0.735	0.6	0.22	2.555	1.1335	0.44	0.6
989	-0.171010	I	0.11	0.09	0.03	0.008	0.0025	0.002	0.003
1066	-0.160921	M	0.665	0.535	0.225	2.1835	0.7535	0.391	0.885
1056	-0.155347	I	0.14	0.105	0.035	0.014	0.0055	0.0025	0.004
637	-0.154234	M	0.175	0.125	0.04	0.024	0.0095	0.006	0.005

缺少值、基数和描述性统计

亚马逊 SageMaker Autopilot 检查并报告数据集各列的属性。在提供此分析的数据报告的每个部分中，内容都按顺序排列，以便您可以首先检查最 “可疑” 的值。使用这些统计数据，您可以改进单个列的内容，并提高 Autopilot 生成的模型的质量。

Autopilot 计算包含它们的列中的分类值的几个统计数据。其中包括唯一条目的数量以及对于文本而言，唯一单词的数量。这些都在表格中显示，供您检查。



	Number of Unique Entries	Number of Unique Words (if Text)
0	3	n/a
y	23	n/a
3	46	n/a
2	104	n/a
1	121	n/a
7	553	n/a
6	657	n/a
5	905	n/a
4	1138	n/a

Autopilot 计算包含这些数值的列中的数值的几个标准统计数据。其中包括平均值、中值、最小值和最大值，以及数值类型和异常值的百分比。这些都在表格中显示，供您检查。

	% of Numerical Values	Mean	Median	Min	Max	% of Outlier Values
y	100.0%	9.93957	9.0	3.0	27.0	nan
1	100.0%	0.523612	0.545	0.11	0.815	0.0
2	100.0%	0.407799	0.425	0.09	0.65	0.0
3	100.0%	0.13995	0.145	0.015	0.515	0.1
4	100.0%	0.828266	0.81	0.008	2.8255	0.0
5	100.0%	0.358844	0.339	0.0025	1.2395	0.0
6	100.0%	0.180348	0.1725	0.002	0.6415	0.0
7	100.0%	0.238783	0.235	0.003	1.005	0.2

候选项的笔记本

候选定义笔记本包含每个建议的预处理步骤、算法和超参数范围。如果您选择生成笔记本而不运行 AutoML 作业，则可以决定要训练和调整哪些候选项。它们会自动优化以识别最终的最佳候选项。如果您直接运行作业而未先看到候选项，则在作业完成后打开笔记本时，将仅显示最佳候选项。

Autopilot 模型洞察

亚马逊 SageMaker 模型监视器报告提供了 AutoML 作业在自动驾驶仪实验排行榜中生成的候选模型的见解和质量信息。该报告仅为最佳分类模型候选人提供模型见解图表。这包括了解误报/假负面，真正的阳性和误报之间的权衡，以及精度与召回之间的权衡。

Autopilot 还为用于衡量其预测质量的所有候选模型提供标量指标。默认情况下，排行榜视图包括这些指标。为候选模型自动计算的指标取决于要解决的问题的类型。

- 回归：MSE
- 二进制分类：准确性，F1，AUC
- 多类别分类：准确性，f1宏

您可以使用相关指标对候选模型进行排序，以帮助您选择和部署满足业务需求的模型。有关这些指标的定义，请参阅Autopilot 候选项指标主题。

这些区域有： SageMaker 模型监视器报告包含描述 Autopilot 作业特征的详细信息、指标表和几个模型见解。其中包括与分类问题类型相关的模型图表。您可以在中访问这些报告 SageMaker 来自工作室性能打开的页面上的选项卡以确认 AutoML 作业已完成。有关如何创建和运行 AutoML 作业的说明 SageMaker 工作室，请参阅创建亚马逊 SageMaker Autopilot 实验 (p. 172).

主题

- 模型详细信息和指标表 (p. 192)
- 混乱矩阵 (p. 194)
- 接收器操作特征曲线下的面积 (p. 195)
- 精确召回曲线 (p. 196)

模型详细信息和指标表

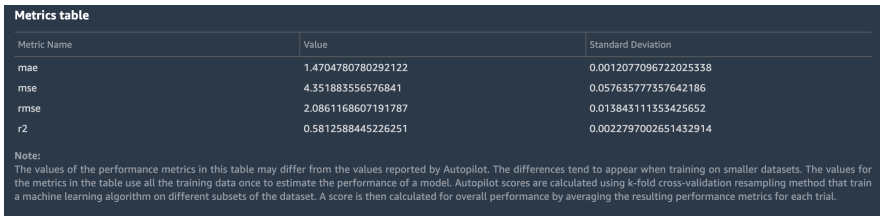
模型详细信息包括以下信息。

- Autopilot 候选项

- Autopilot Job 名称
- 问题类型
- 目标指标
- 优化方向

模型质量信息是由预先构建的 SageMaker 模型监控器容器。生成的报告的内容取决于解决的问题类型：回归、二进制分类或多类分类。该报告指定了评估数据集中包含的行数以及评估的发生时间。

以下是 AutoML 作业为回归问题生成的模型监视器报告中的指标表示例。

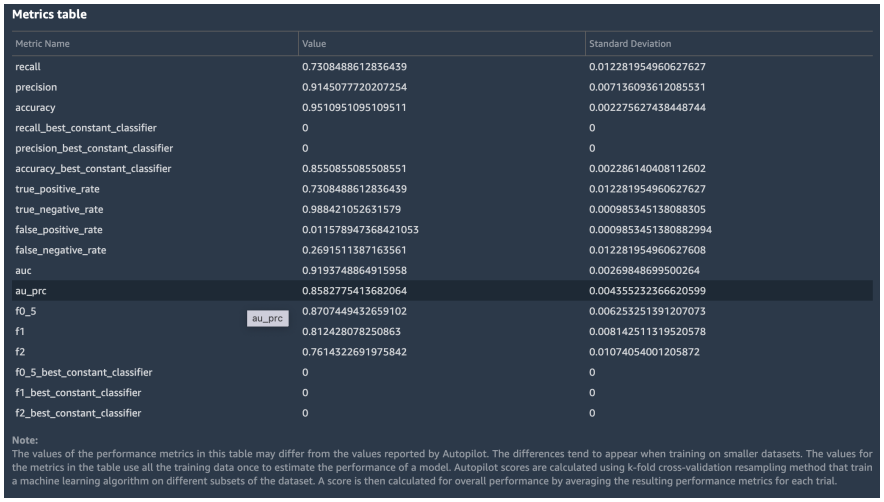


Metrics table

Metric Name	Value	Standard Deviation
mae	1.470478078029212 2	0.0012077096722025338
mse	4.351883556576841	0.057635777357642186
rmse	2.086116860719178 7	0.013843111353425652
r2	0.5812588445226251	0.0022797002651432914

Note:
The values of the performance metrics in this table may differ from the values reported by Autopilot. The differences tend to appear when training on smaller datasets. The values for the metrics in the table use all the training data once to estimate the performance of a model. Autopilot scores are calculated using k-fold cross-validation resampling method that train a machine learning algorithm on different subsets of the dataset. A score is then calculated for overall performance by averaging the resulting performance metrics for each trial.

以下是 AutoML 作业为二进制分类问题生成的模型监视器报告中的指标表示例。



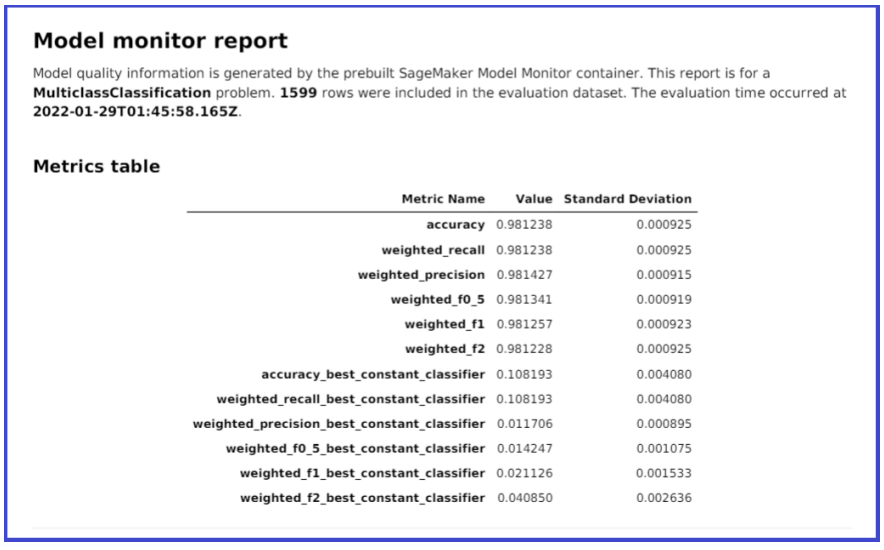
Metrics table

Metric Name	Value	Standard Deviation
recall	0.7308488612836439	0.012281954960627627
precision	0.9145077720207254	0.007136093612085531
accuracy	0.9510951095109511	0.002275627438448744
recall_best_constant_classifier	0	0
precision_best_constant_classifier	0	0
accuracy_best_constant_classifier	0.8550855085508551	0.002286140408112602
true_positive_rate	0.7308488612836439	0.012281954960627627
true_negative_rate	0.988421052631579	0.000985345138088305
false_positive_rate	0.011578947368421053	0.0009853451380882994
false_negative_rate	0.2691511387163561	0.012281954960627608
auc	0.9193748864915958	0.00269848699500264
au_prc	0.8582775413682064	0.004355232366620599
f0_5	0.8707449432659102	0.006253251391207073
f1	0.812428078250863	0.008142511319520578
f2	0.7614322691975842	0.01074054001205872
f0_5_best_constant_classifier	0	0
f1_best_constant_classifier	0	0
f2_best_constant_classifier	0	0

au_prc

Note:
The values of the performance metrics in this table may differ from the values reported by Autopilot. The differences tend to appear when training on smaller datasets. The values for the metrics in the table use all the training data once to estimate the performance of a model. Autopilot scores are calculated using k-fold cross-validation resampling method that train a machine learning algorithm on different subsets of the dataset. A score is then calculated for overall performance by averaging the resulting performance metrics for each trial.

以下是 AutoML 作业为多类分类问题生成的模型监视器报告中的指标表示例。



Model monitor report

Model quality information is generated by the prebuilt SageMaker Model Monitor container. This report is for a **MulticlassClassification** problem. **1599** rows were included in the evaluation dataset. The evaluation time occurred at **2022-01-29T01:45:58.165Z**.

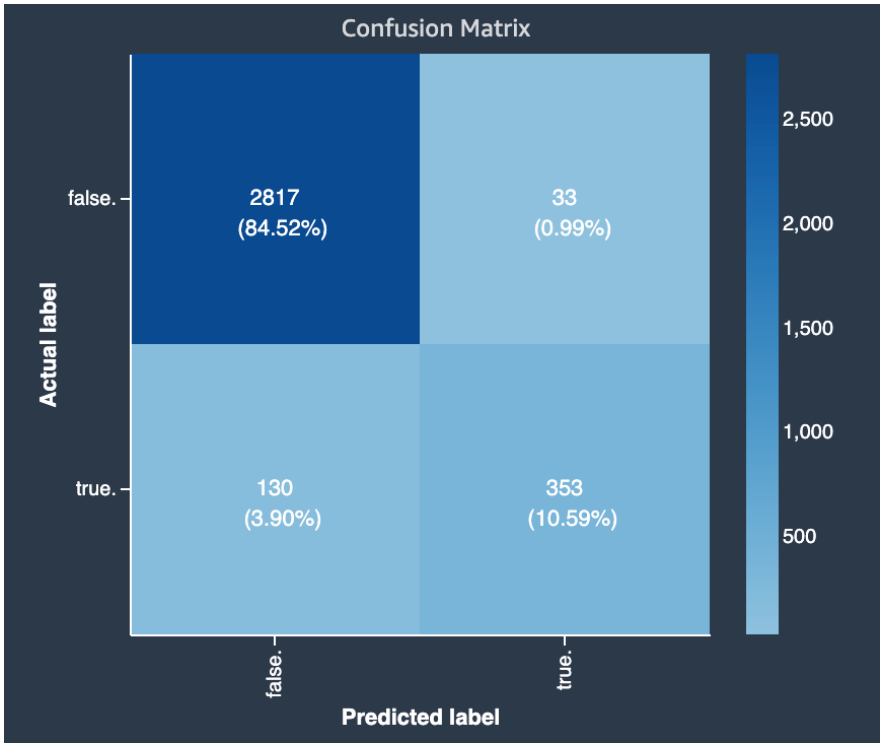
Metrics table

Metric Name	Value	Standard Deviation
accuracy	0.981238	0.000925
weighted_recall	0.981238	0.000925
weighted_precision	0.981427	0.000915
weighted_f0_5	0.981341	0.000919
weighted_f1	0.981257	0.000923
weighted_f2	0.981228	0.000925
accuracy_best_constant_classifier	0.108193	0.004080
weighted_recall_best_constant_classifier	0.108193	0.004080
weighted_precision_best_constant_classifier	0.011706	0.000895
weighted_f0_5_best_constant_classifier	0.014247	0.001075
weighted_f1_best_constant_classifier	0.021126	0.001533
weighted_f2_best_constant_classifier	0.040850	0.002636

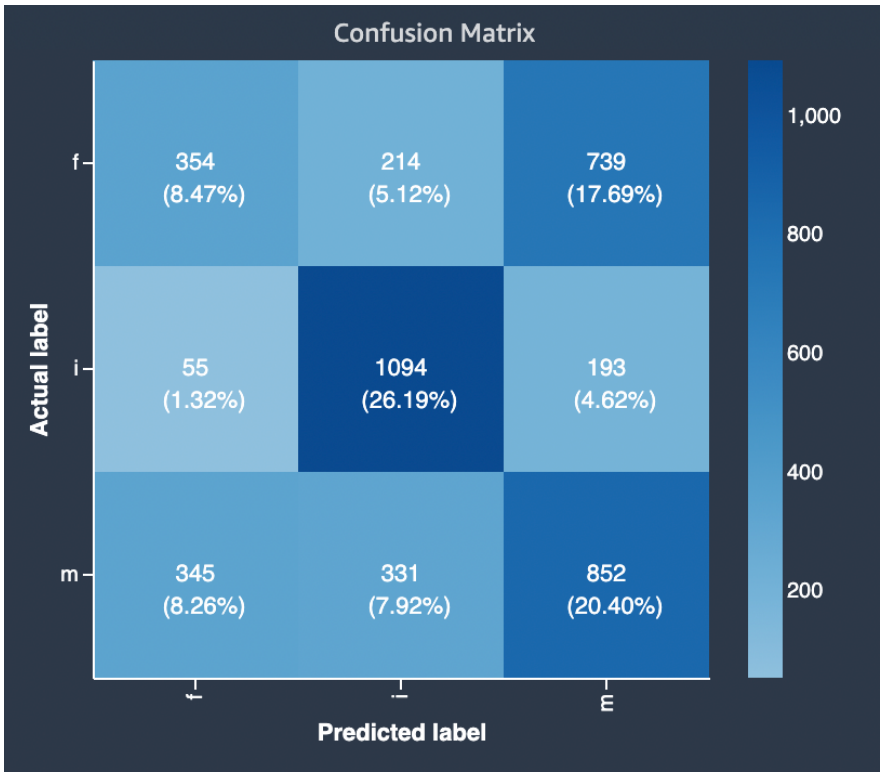
混乱矩阵

混淆矩阵提供一种可视化为不同类别进行二元和多类别分类预测的准确性的方法。混淆矩阵是一个表，其中包含实际标签的正确和不正确预测的百分比。混淆矩阵中的每一行都表示如何按模型预测的标签对实际标签进行分类。准确预测的百分比在对角线上，从左上角到右下角。非对角线百分比表示模型预测的错误分类类型。这些错误的预测就是混淆值。

以下是一个二元分类问题的混淆矩阵示例。



此为多类别分类问题的混淆矩阵示例。



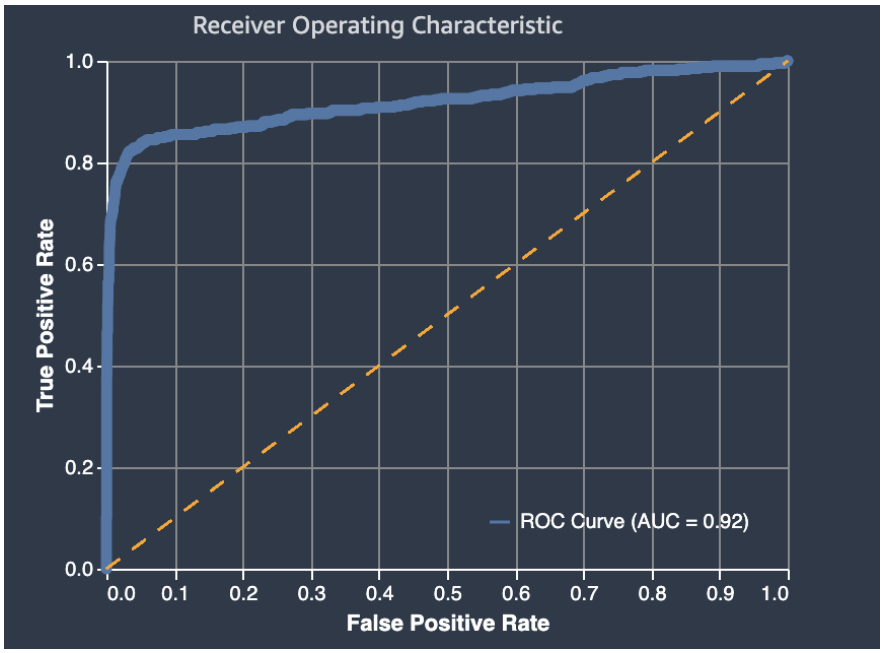
此报告提供了一个混淆矩阵，最多可容纳 15 个多类分类问题类型的标签。标签按顺序列出，从预测最不准确的标签到预测最准确的标签。如果行显示`Nan`，这意味着验证数据集没有该标签的行。

接收器操作特征曲线下的面积

接收机操作特征曲线下的区域（AUC ROC 曲线）表示真正阳性和误报率之间的权衡。AUC ROC 曲线是用于二进制分类模型的行业标准准确度指标。AUC 衡量模型为正面示例预测出相比负面示例更高分数的能力。AUC 指标提供了对所有可能分类阈值中模型性能的综合度量。

AUC 指标返回从零 (0) 到一 (1) 的十进制值。接近 1 的 AUC 值指示高度准确的 ML 模型。接近 0.5 的值指示 ML 模型比随便猜测好不了多少。值接近 0 的情况很少见，这些值通常表示数据有问题。基本上，接近 0 的 AUC 表示 ML 模型已学习了正确的模式，但使用它们来预测尽可能不准确。例如，0 预测为 1，1 为 0。有关 AUC 指标的更多信息，请参阅接收器运行特点维基百科上的文章。

分类的二进制模型 no-better-than-random 猜测，真实和误报率相等，AUC 分数为 0.5。表示随机二进制分类器的曲线是接收器运行特征图中的对角线虚线红线。更准确的分类模型的曲线高于这个随机基线，其中真正阳性的比率超过了误报率。



这些区域有：误报率(FPR) 衡量错误警报率或虚假预测为正例的实际负例的比率。范围为 0 至 1。较小的值表示更好的预测准确度。

- 对于 = FP/ (FP+TN)

这些区域有：真正汇率（TPR）衡量预测为阳性的实际阳性分数。范围为 0 至 1。较大的值（1 为最大）表示更好的预测准确度。

- TPR = TP/ (TP+FN)

这些费率的定义如下。

- 正确预测
  - 真正向(TP): 预测的值为 1，真正的值为 1。
  - 真正负值(TN): 预测的值为 0，真正的值为 0。
- 错误预测
  - 误报(FP): 预测的值为 1，但真正的值为 0。
  - 误报(FN): 预测的值为 0，但真正的值为 1。

精确召回曲线

精度召回曲线代表了在二进制分类问题中使用的不同阈值的精度和召回之间的权衡。二进制分类问题的目的是尽可能多地对训练数据集中标记为正面的相关元素进行正确分类。召回率高但精度低的系统会返回大量相关结果，但与训练标签相比，其预测标签中有很高百分比的标签是不正确的。高精度但召回率低的系统返回的相关结果较少，但与训练标签相比，很高百分比的预测标签是正确的。具有高精度和高召回能力的完美系统可产生许多正确标记的结果。有关更多信息，请参阅 。精度和召回维基百科中的文章。

精度精度衡量预测为正向的实际正例的比率。范围为 0 至 1。较大的值表示预测的值准确度越高。

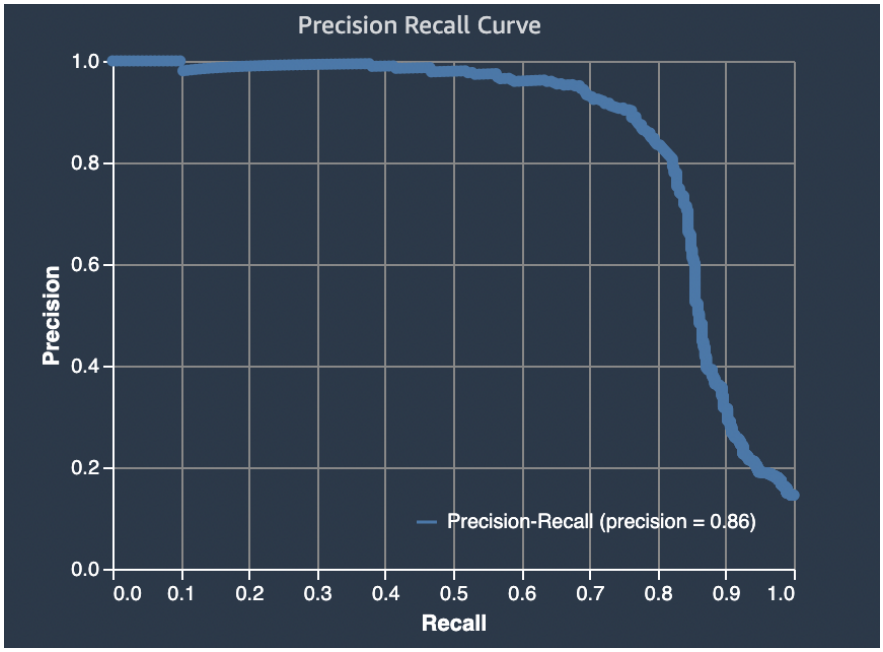
- 精度 = TP/（TP+FP）

调用精度衡量实际正例与样本中预测为正向的实际正例的比率。这也称为敏感度和真正的正率。范围为 0 至 1。值越大表示从样本中检测到的正值得更好。

- 召回 = TP/ (TP+FN)

亚马逊 SageMaker 自动驾驶仪报告了精确召回曲线 (AUPRC) 下的区域。AUPRC 指标提供了对所有可能分类阈值中模型性能的综合度量。

下面是一个示例，比较了在同一数据集上训练的四种不同模型的精度召回曲线及其 AUPRC 值。



在 AutoPilot 生成的容器中配置推理输出

亚马逊 SageMaker 自动驾驶仪生成有序`ContainerDefinition`可用于构建模型以在机器学习管道中进行部署的列表。该模型可用于在线托管和推理。客户可以使用`ListCandidateForAutoMLJob`API。代表最佳候选项的推断容器定义列表也可作为的一部分提供。`DescribeAutoMLJob`响应。

主题

- 回归和分类问题类型的推理容器定义 (p. 197)
- 为分类模型选择推理响应 (p. 198)

回归和分类问题类型的推理容器定义

生成的推理容器取决于作业的问题类型。

- 回归：生成两个容器：
  1. 首先是要素工程容器，它将原始要素转换为回归算法可以训练的要素。
  2. 其次是转换要素并为数据集生成回归分数的算法容器。
- Classification：生成三个容器：
  1. 将原始要素转换为分类算法可以训练的要素的要素的要素工程容器。
  2. 生成胜利的算法容器`predicted_label`. 它还可以产生与推理响应中的分类结果相关的各种概率。

3. 执行算法预测的后期处理的要素工程容器。例如，将预测的标签反向转换为原始标签。

为分类模型选择推理响应

使用分类推理容器，您可以选择推理响应的内容。有四个预定义的密钥：

- `predicted_label`：由自动驾驶仪确定的获奖标签。
- `probability`: 的概率`True`二进制分类的类。多类别分类中奖类别的概率。
- `probabilities`：所有相应的概率列表。
- `labels`：所有标签列表

默认情况下，推理容器配置为生成`predicted_label`仅限 。

使用三个环境变量来选择可选的推理内容：

- `SAGEMAKER_INFERENCE_SUPPORTED`：设置这是为了向您提供有关每个容器支持哪些内容的提示。
- `SAGEMAKER_INFERENCE_INPUT`: 应该设置为容器在输入有效负载中预期的密钥。
- `SAGEMAKER_INFERENCE_OUTPUT`: 应该填充容器输出的一组密钥。

为了选择推理响应内容，我们需要添加`SAGEMAKER_INFERENCE_INPUT`、`SAGEMAKER_INFERENCE_OUTPUT`适当地放在集装箱清单中的第二个和第三个集装箱中，以解决分类问题。

第三个分类模型容器支持的密钥是`predicted_label`、`labels`、`probability`和`probabilities`因此`SAGEMAKER_INFERENCE_SUPPORTED`环境中包括所有这些密钥的名称。

第二个容器（算法）支持的密钥是预测的 _label、概率和概率。请注意，`标签 '是故意不会添加到支持的 SAGEMAKER_INFERENCE_ 中。

以下是如何更新要接收的推理容器的定义`predicted_label`和`probability`.

```
containers[1]['Environment'].update({'SAGEMAKER_INFERENCE_OUTPUT': 'predicted_label,
 probability'})
containers[2]['Environment'].update({'SAGEMAKER_INFERENCE_INPUT': 'predicted_label,
 probability'})
containers[2]['Environment'].update({'SAGEMAKER_INFERENCE_OUTPUT': 'predicted_label,
 probability'})
```

以下是如何更新要接收的推理容器的定义`predicted_label`和`probabilities`和`labels`. 请注意，您不需要传递`labels`到第二个容器，算法容器。这是多余的，因为它可以由第三个容器独立生成。这减少了延迟。

```
containers[1]['Environment'].update({'SAGEMAKER_INFERENCE_OUTPUT':
 'predicted_label,probabilities'})
containers[2]['Environment'].update({'SAGEMAKER_INFERENCE_INPUT':
 'predicted_label,probabilities'})
containers[2]['Environment'].update({'SAGEMAKER_INFERENCE_OUTPUT': 'predicted_label,
 probabilities,labels'})
```

您可以使用亚马逊 SageMaker Python 开发工具包以完成以下操作：

```
from sagemaker import AutoML

aml = AutoML.attach(auto_ml_job_name='AUTOML_JOB_NAME')
```

```
aml_best_model = aml.create_model(name='SELECT_MODEL_NAME',
                                  candidate=None,
                                  inference_response_keys=['probabilities', 'labels'])

aml_transformer = aml_best_model.transformer(accept='text/csv',
                                             assemble_with='Line',
                                             instance_type='ml.m5.xlarge',
                                             instance_count=1,)

aml_transformer.transform(test_data_s3_path,
                          content_type='text/csv',
                          split_type='Line',
                          job_name=<Add jobName>,
                          wait=True)
```

亚马逊 SageMaker Autopilot 配额

在使用亚马逊时，有一些配额限制了您可用的资源 SageMaker Autopilot. 其中一些限制是可以增加的，有些则没有。

> Note
>
> 以下章节中记录的资源配额对于亚马逊版本有效 SageMaker 3.22.2 及更高版本。有关更新的版本的信息 SageMaker 工作室，请参阅关闭并更新 SageMaker 工作室和工作室应用 (p. 128).

主题

- 你可以增加的配额 (p. 199)
- 资源配额 (p. 200)

你可以增加的配额

对于输入数据集的大小有默认限制：单个 Pacic 文件的文件大小 (*)、目标数据集大小子采样 (**) 以及可以使用 Amazon 运行的并发作业数量 SageMaker 每个自动驾驶仪Amazon每个账户Amazon区域。

资源限制

资源	区域	默认限制	最多可增加到
输入数据集的大小	全部	100 GB	数百 GBs
单个镶木地板文件的大小 *	全部	2GB	数十 GB
用于子采样的目标数据集大小 **	全部	5GB	数百 GBs
并发 Autopilot 作业数量	us-east-1、us-east-2、us-west-2、ap-nornor-1、eu-west-1、eu-west-1、eu-nor-1、eu-nor-1	4	数百
	ap-northeast-2、ap-northeast-2、eu-west-2、ap-南-1	2	数百
	所有其他区域	1	十

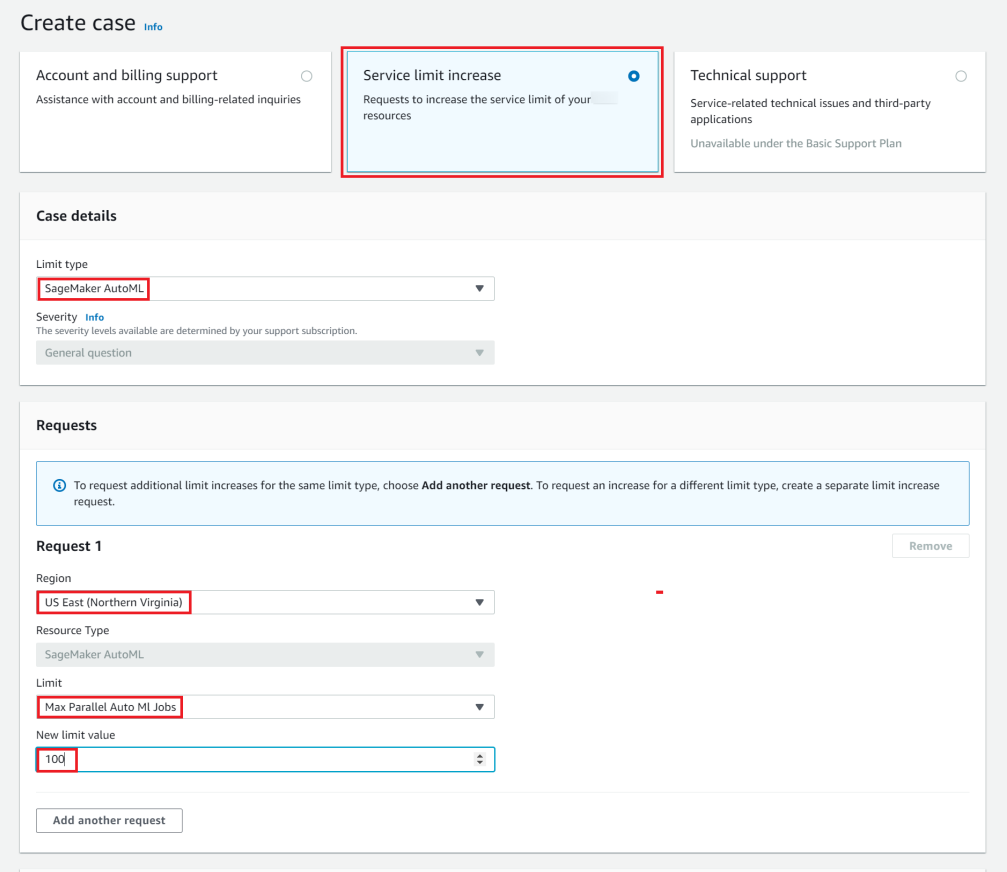
Note

* 这个 2 GB 的大小限制适用于单个压缩的 Pacia 文件。您可以提供一个包含多个压缩 Placia 文件的 Placia 数据集。解压缩文件后，它们可能会逐个扩展到更大的大小。
** Autopilot 会自动对大于目标数据集大小的输入数据集进行子采样，同时考虑到类别不平衡并保留稀有类标签。

您可以通过联系 Support 来提高这些限制。

要请求提高配额：

1. 打开Amazon支持中心页面，登录（如有必要），然后选择创建案例.
2. 在 Create case (创建案例) 页面上，选择 Service limit increase (增加服务限制)。
3. 在案例详细信面板中，选择SageMaker AutoML(对于)限制类型.
4. 在存储库的请求用于的面板请求 1中，选择区域，资源限制要增加和新限制值你正在请求。Select添加其他请求如果你有额外的增加配额请求。



5. 提供首选项联系选项然后选择提交.

资源配额

下表包含 Amazon 的运行时间资源限制。 SageMaker 在一个自动驾驶仪工作Amazon区域。

每个 AutoPilot 作业的资源限制

资源	每个自动驾驶仪作业的限制
Autopilot 作业的最大运行时间	30 天

Amazon API 参考指南 SageMaker Autopilot

亚马逊 SageMaker 提供 API 参考文档，该文档描述 Autopilot 使用的所有 REST 操作和数据类型以及更高级别。亚马逊 SageMaker Python 开发工具包您可以使用它来创建和管理 AutoML 作业。它还提供命令行界面 (CLI)、Amazon适用于 Python 的开发工具包 (Boot) SageMaker 服务以及适用于 .NET、C++、Go、Java、JavaScript、PHP V3 和 Ruby V3 的开发工具包。以下各节介绍了这些 Autopilot 编程接口。

主题

- SageMaker API 参考 (p. 201)
- 亚马逊 SageMaker Python 开发工具包 (p. 202)
- Amazon 命令行界面（CLI）(p. 202)
- AmazonSDK for Python (Boto) (p. 202)
- 适用于 .NET 的Amazon开发工具包 (p. 203)
- 适用于 C++ 的 Amazon 软件开发工具包 (p. 203)
- 适用于 Go 的 Amazon 开发工具包 (p. 203)
- Amazon SDK for Java (p. 203)
- Amazon适用于 JavaScript 的开发工具包 (p. 203)
- Amazon适用于 PHP V3 的开发工具包 (p. 204)
- Amazon适用于 Ruby V3 的开发工具包 (p. 204)

SageMaker API 参考

此 API 提供用于创建和管理 Amazon 的 HTTP 服务 API SageMaker Autopilot 资源。

操作

- `CreateAutoMLJob`
- `DescribeAutoMLJob`
- `ListAutoMLJobs`
- `ListCandidatesForAutoMLJob`
- `StopAutoMLJob`

数据类型

- `AutoMLCandidate`
- `AutoMLCandidateStep`
- `AutoMLChannel`
- `AutoMLContainerDefinition`
- `AutoMLDataSource`
- `AutoMLJobArtifacts`
- `AutoMLJobCompletionCriteria`
- `AutoMLJobConfig`

- `AutoMLJobObjective`
- `AutoMLJobSummary`
- `AutoMLOutputDataConfig`
- `AutoMLPartialFailureReason`
- `AutoMLS3DataSource`
- `AutoMLSecurityConfig`
- `CandidateArtifactLocations`
- `CandidateProperties`
- `FinalAutoMLJobObjectiveMetric`
- `MetricDatum`
- `ModelDeployConfig`
- `ModelDeployResult`
- `ResolvedAttributes`
- `TuningJobCompletionCriteria`

有关整个内容的更多信息 SageMaker REST API，请参阅API 和开发工具包参考.

亚马逊 SageMaker Python 开发工具包

此 Python 库提供几个用于处理 SageMaker 的高级抽象。以下类可用于管理 AutoML 作业。

- `AutoML`
- `AutoMLInput`
- `AutoMLJob`
- `CandidateEstimator`
- `CandidateStep`

有关此 Python SDK 如何简化模型训练和部署的更多信息，请参阅。使用亚马逊 SageMaker Python 开发工具包.

Amazon 命令行界面（CLI）

这些区域有：Amazon CLI提供 API 以创建和管理 SageMaker 资源的费用。以下是Amazon CLIAmazon SageMakerAutopilot 命令。

- `create-auto-ml-job`
- `describe-auto-ml-job`
- `list-auto-ml-jobs`
- `list-candidates-for-auto-ml-job`
- `stop-auto-ml-job`

AmazonSDK for Python (Boto)

Boot 是 Amazon Web Services (Amazon) 适用于 Python 的工具包。它使 Python 开发人员能够创建、配置和管理AmazonSageMaker 等服务。Boot 提供低级别客户端映射到底层账户的 API SageMaker 服务 API。以下是用来通过 Client 类管理 AutoML 作业的方法列表。

- `create_auto_ml_job()`

- describe_auto_ml_job()
- list_auto_ml_jobs()
- list_candidates_for_auto_ml_job()
- stop_auto_ml_job()

适用于 .NET 的Amazon开发工具包

.NET 开发工具包使开发人员能够创建、配置和管理AmazonSageMaker 等服务。此 API 映射到底层 SageMaker 服务 API。以下是用来通过 Client 类管理 AutoML 作业的方法列表。

- CreateAutoMLJob
- DescribeAutoMLJob
- ListAutoMLJobs
- ListCandidatesForAutoMLJob
- StopAutoMLJob

适用于 C++ 的 Amazon 软件开发工具包

C++ 开发工具包使开发人员能够创建、配置和管理AmazonSageMaker 等服务。此 API 映射到底层 SageMaker 服务 API。有关用于通过 Client 类管理 AutoML 作业的方法的信息，请参阅 Aws::SageMaker::SageMakerClient Class Reference。

适用于 Go 的 Amazon 开发工具包

Go 开发工具包使开发人员能够创建、配置和管理AmazonSageMaker 等服务。此 API 映射到底层 SageMaker 服务 API。以下是用来通过 Client 类管理 AutoML 作业的方法列表。

- CreateAutoMLJob
- DescribeAutoMLJob
- ListAutoMLJobs
- ListCandidatesForAutoMLJob
- StopAutoMLJob

Amazon SDK for Java

Java 开发工具包使开发人员能够创建、配置和管理AmazonSageMaker 等服务。此 API 映射到底层 SageMaker 服务 API。以下是用来通过 Client 类管理 AutoML 作业的方法列表。

- createAutoMLJob
- describeAutoMLJob
- listAutoMLJobs
- listCandidatesForAutoMLJob
- stopAutoMLJob

Amazon适用于 JavaScript 的开发工具包

这些区域有： JavaScript 开发工具包使开发人员能够创建、配置和管理AmazonSageMaker 等服务。此 API 映射到底层 SageMaker 服务 API。以下是用来通过 Client 类管理 AutoML 作业的方法列表。

- createAutoMLJob
- describeAutoMLJob
- listAutoMLJobs
- listCandidatesForAutoMLJob
- stopAutoMLJob

Amazon适用于 PHP V3 的开发工具包

PHP V3 开发工具包使开发人员能够创建、配置和管理AmazonSageMaker 等服务。此 API 映射到底层 SageMaker 服务 API。以下是用来通过 Client 类管理 AutoML 作业的方法列表。

- CreateAutoMLJob
- DescribeAutoMLJob
- ListAutoMLJobs
- ListCandidatesForAutoMLJob
- StopAutoMLJob

Amazon适用于 Ruby V3 的开发工具包

Ruby V3 开发工具包使开发人员能够创建、配置和管理AmazonSageMaker 等服务。此 API 映射到底层 SageMaker 服务 API。以下是用来通过 Client 类管理 AutoML 作业的方法列表。

- create_auto_ml_job()
- describe_auto_ml_job()
- list_auto_ml_jobs()
- list_candidates_for_auto_ml_job()
- stop_auto_ml_job()

标记数据

这些功能在中国区域中不可用。

要训练机器学习模型，您需要一个大型、高质量、已标记的数据集。您可以使用 Amazon SageMaker Ground Truth 为您的数据添加标签。选择一个 Ground Truth内置任务类型或者创建您自己的定制标记工作流. 要提高数据标签的准确性并减少标记数据的总成本，请使用 Ground Truth 增强型数据标记功能，例如自动标记数据和注释合并.

主题

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- 创建和管理人力 (p. 463)
- Crowd HTML 元素参考 (p. 485)

使用 Amazon SageMaker 标记数据的 Ground Truth

此功能在中国区域中不可用。

要训练机器学习模型，您需要一个大型、高质量、已标记的数据集。Ground Truth 可帮助您为机器学习模型构建高质量训练数据集。借助 Ground Truth，您可以将来自 Amazon Mechanical Turk（您选择的供应商公司）或内部私有人力资源的工作人员与机器学习相结合，以便创建已标记的数据集。您可以使用来自 Ground Truth 的已标记数据集来训练您自己的模型。您也可以将输出用作 Amazon 的训练数据集。SageMaker 模型。

根据您的 ML 应用程序，您可以选择一种 Ground Truth 内置任务类型，以使工作人员为您的数据生成特定类型的标签。您也可以构建自定义标记工作流，以便为标记数据的工作人员提供您自己的 UI 和工具。要了解内置任务类型 Ground Truth 更多信息，请参阅内置任务类型 (p. 351). 要了解如何创建自定义标签工作流，请参阅创建自定义标记工作流 (p. 325)。

为了自动标注训练数据集，您可以选择使用自动数据标记，这 Ground Truth 个使用机器学习来确定人们需要标记哪些数据的流程。自动数据标记功能可以减少所需的标记时间和人工操作。有关更多信息，请参阅 自动数据标记 (p. 438)。要创建自定义标记工作流，请参阅创建自定义标记工作流 (p. 325).

使用预构建或自定义工具为训练数据集分配标记任务。一个为 UI 模板贴标是一个网页，Ground Truth 使用它来向工作人员提供任务和说明。这些区域有： SageMaker 控制台提供内置的模板来标记数据。您可以从这些模板入手，也可以使用我们的 HTML 2.0 组件构建您自己的任务和说明。有关更多信息，请参阅 创建自定义标记工作流 (p. 325)。

使用您选择的人力来标记您的数据集。您可以从以下选项中选择人力：

- 由世界各地超过 50 万独立承包商组成的 Amazon Mechanical Turk 人力。
- 您基于您的员工或承包商创建的用于处理组织内数据的私有人力。
- 您可以在 Amazon Web Services Marketplace 中找到的专门从事数据标记服务的供应商公司。

有关更多信息，请参阅 创建和管理人力 (p. 463)。

您将数据集存储在 Amazon S3 存储桶中。存储桶包含三件事：要标记的数据、Ground Truth 用于读取数据文件的输入清单文件，以及输出清单文件。输出文件包含标记作业的结果。有关更多信息，请参阅 使用输入和输出数据 (p. 377)。

来自标签工作的事件显示在亚马逊 CloudWatch 在下`/aws/sagemaker/LabelingJobs`组中)。CloudWatch 使用标记作业名称作为日志流的名称。

您是 Ground Truth 的新用户吗？

如果您是 Ground Truth 的新用户，我们建议您执行以下操作：

1. Read开始使用 (p. 206)— 此部分引导您完成首个 Ground Truth 标记作业的过程。
2. 探索其他主题— 根据您的需求，执行以下操作：
   - 探索内置任务类型— 使用内置任务类型简化标签作业的创建过程。请参阅内置任务类型 (p. 351)要了解 Ground Truth 内置任务类型的更多信息。
   - 管理标签工作人员— 创建新的工作团队和管理您的现有人力。有关更多信息，请参阅 创建和管理人力 (p. 463)。
   - 了解流标记作业的信息— 创建流式标注作业并使用永久运行的标注作业将新的数据集对象实时发送给工作人员。只要标签作业处于活动状态且正在向其发送新对象，工作人员就会持续接收要标记的新数据对象。要了解更多信息，请参阅 Ground Truth 直播标签职位 (p. 381)。
3. 请参阅**`Reference`**— 本节介绍自动执行 Ground Truth 操作的操作。

开始使用

此视频介绍了如何设置和使用 Amazon SageMaker Ground Truth。（长度：9:37）

要开始使用 Amazon SageMaker Ground Truth，请按照以下各节中的说明操作。以下几节介绍了如何使用控制台创建标记作业，分配公有或私有人力，以及将标记作业发送到您的人力。您还可以了解如何监控标记作业的进度。

如果您要创建自定义标记工作流，请参阅创建自定义标记工作流 (p. 325)有关说明。

在创建标记作业之前，您必须将数据集上传到 Amazon S3 存储桶。有关更多信息，请参阅 使用输入和输出数据 (p. 377)。

主题

- 第 1 步：开始前的准备工作 (p. 206)
- 第 2 步：创建标记作业 (p. 207)
- 第 3 步：选择员工 (p. 208)
- 第 4 步：配置边界框工具 (p. 209)
- 第 5 步：监控标记 Job (p. 210)

第 1 步：开始前的准备工作

在开始使用 SageMaker 控制台创建标记作业之前，您必须设置要使用的数据集。请执行此操作：

1. 在可公开访问的 HTTP URL 中保存两个图像。在创建用于完成标记任务的说明时要使用这些图像。这些图像应具有大约 2:1 的宽高比。对于本练习，图像的内容并不重要。
2. 创建 Amazon S3 存储桶以存放输入和输出文件。存储桶必须位于运行 Ground Truth 的同一区域。记下存储桶名称，因为您在步骤 2 中使用它。

Ground Truth 要求所有包含标签作业输入映像数据的 S3 存储桶都附加 CORS 策略。要了解有关此更改的更多信息，请参CORS 许可要求 (p. 446).

3. 将以下权限策略分配给要创建标记作业的用户：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "sagemakergroundtruth",
            "Effect": "Allow",
            "Action": [
                "cognito-idp:CreateGroup",
                "cognito-idp:CreateUserPool",
                "cognito-idp:CreateUserPoolDomain",
                "cognito-idp:AdminCreateUser",
                "cognito-idp:CreateUserPoolClient",
                "cognito-idp:AdminAddUserToGroup",
                "cognito-idp:DescribeUserPoolClient",
                "cognito-idp:DescribeUserPool",
                "cognito-idp:UpdateUserPool"
            ],
            "Resource": "*"
        }
    ]
}
```

下一页

第 2 步：创建标记作业 (p. 207)

第 2 步：创建标记作业

在本步骤中，您使用控制台来创建标记作业。您可以将存储清单文件的 Amazon S3 存储桶告知 Amazon S3 存储桶，并配置作业的参数。有关在 Amazon S3 存储桶中存储数据的更多信息，请参阅使用输入和输出数据 (p. 377).

创建标记作业

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在左侧导航栏中，选择 Labeling jobs (标记作业)。
3. 选择 Create labeling job (创建标记作业) 以开始作业创建过程。
4. 在 Job overview (作业概览) 部分中，提供以下信息：
   - 作业名称— 为标记作业提供一个描述作业的名称。此名称将显示在作业列表中。该名称在您的账户和 Amazon 区域中都必须是唯一的。
   - 标记属性名称— 请将此选项保持不选中，因为默认值是此入门作业的最佳选项。
   - 输入数据设置— 选择自动化数据设置. 此选项允许您自动连接到 S3 中的输入数据。
   - 输入数据集的 S3 位置— 输入您在步骤 1 中添加映像的 S3 位置。
   - 输出数据集的 S3 位置— 在 S3 中写入输出数据的位置。
   - 数据类型— 使用下拉菜单选择Image. Ground Truth 将使用在 S3 位置找到的所有图像用于输入数据集的图像作为标注作业的输入。
   - IAM 角色— 创建或选择附加了 AmazonSageMakerFullAccess IAM 策略的 IAM 角色。
5. 在任务类型部分，用于任务类别字段中，选择Image.
6. 在任务选择选择边界框.

7. 选择 Next (下一步) 继续配置您的标记作业。

下一页

第 3 步：选择员工 (p. 208)

第 3 步：选择员工

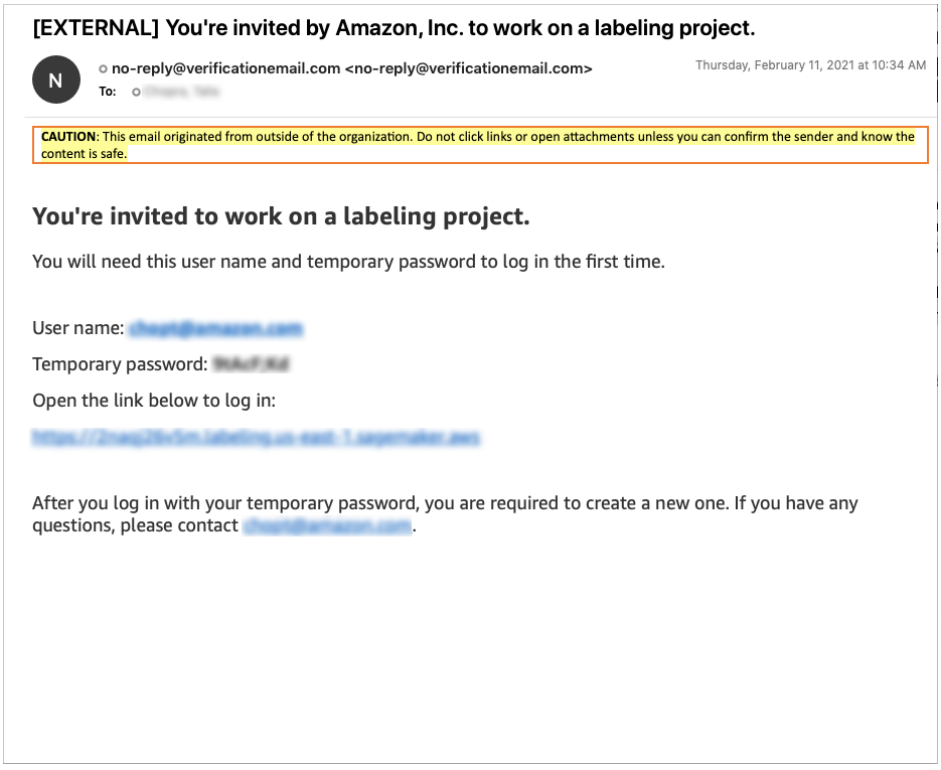
在此步骤中，您选择用于标记数据集的人力。建议您创建一支私人劳动力来测试 Amazon SageMaker Ground Truth。使用电子邮件地址邀请您的人力的成员。如果您在此步骤中创建私有人力，则以后将无法导入 Amazon Cognito 用户池。如果您要使用 Amazon Cognito 用户池创建私有人力，请参阅管理私有人力（Amazon Cognito）(p. 470)然后在本教程中使用 Mechanical Turk 员。

> Tip
>
> 要了解可与 Ground Truth 配合使用的其他人力选项，请参阅创建和管理人力 (p. 463).

要创建私有人力：

1. 在 Workers (工作人员) 部分，选择 Private (私有)。
2. 如果这是您第一次使用私有人力，请在 Email addresses (电子邮件地址) 字段中输入最多 100 个电子邮件地址。地址必须用逗号分隔。您应包含您自己的电子邮件地址，以便您成为人力的成员并可以查看数据对象标记任务。
3. 在组织名称字段中，输入组织的名称。此信息用于自定义所发送的用于邀请他人加入私有人力的电子邮件。您可以在通过控制台创建用户池后更改组织名称。
4. 在联系电子邮件字段中，输入人力成员用于报告任务相关问题的电子邮件地址。

如果您将自己加入私有人力，您将收到类似于以下内容的电子邮件。亚马逊 Inc.将替换为您在上述过程的步骤 3 中输入的组织。选择电子邮件中的链接以使用提供的临时密码登录。如果出现提示，请更改密码。成功登录后，您会看到标签任务显示的工作人员门户。



Tip

您可以在标记人力SageMaker 控制台的 Ground Truth 区域的部分。要查看链接，请选择私密选项卡。链接位于标记门户登录 URL中的标头私人劳动力总结.

如果您选择使用 Amazon Mechanical Turk 人力来标记数据集，将向您收取在数据集上完成标记任务的费用。

要使用 Amazon Mechanical Turk 员队伍：

1. 在 Workers (工作人员) 部分，选择 Public (公有)。
2. 设置每个任务的价格.
3. 如果适用，请选择数据集不包含成人内容以确认示例数据集没有成人内容。此信息使 Amazon SageMaker Ground Truth 能够警告 Mechanical Turk 上的外部工作人员，他们可能会在数据集中遇到潜在的冒犯性内容
4. 选中以下语句旁的复选框，以确认示例数据集不包含任何个人身份信息 (PII)。这是使用具有 Ground Truth 的 Mechanical Turk 的要求。如果您的输入数据确实包含 PII，请在本教程中使用私人工作人员。

   您了解并同意，Amazon Mechanical Turk 人力由位于世界各地的独立承包商组成，您不应与此人力共享机密信息、个人信息或受保护的健康信息。

下一页

第 4 步：配置边界框工具 (p. 209)

第 4 步：配置边界框工具

最后，配置边界框工具以向工作人员提供说明。您可以配置任务标题，以描述任务并为工作人员提供概括性说明。您可以提供快速说明和完整说明。快速说明显示在要标记的图像旁边。完整说明包含完成任务的详细说明。在该示例中，您仅提供快速说明。您可以选择该部分底部的 Full instructions (完整说明) 以查看完整说明示例。

配置边界框工具

1. 在任务描述字段中，键入任务的简短说明。例如：

   Draw a box around any *objects* in the image.

   将 *objects* 替换为图像中显示的对象的名称。
2. 在标签字段中，键入工作人员应围绕着它绘制边界框的对象的类别名称。例如，如果您要求工作人员围绕足球运动员绘制框，您可以在此字段中使用 “足球播放器”。
3. 在 Short instructions (简短说明) 部分中，您可以使用工作人员要标记的图像创建在页面上显示的说明。我们建议您包含绘制正确的边界框示例以及绘制错误的边界框示例。要创建您自己的说明，请使用以下步骤：

   a. 选择 GOOD EXAMPLE (好示例) 与图像占位符之间的文本。将其替换为以下文本：

      Draw the box around the object with a small border.

   b. 选择第一个图像占位符并将其删除。

   c. 选择图像按钮，然后输入您在步骤 1 中创建的其中一个图像的 HTTPS URL。也可以直接在简短说明部分嵌入图像，但是本部分的配额为 100 千字节（包括文本）。如果您的图像和文字超过 100 千字节，则会收到错误。

   d. 选择 BAD EXAMPLE (坏示例) 与图像占位符之间的文本。将其替换为以下文本：

      Don't make the bounding box too large or cut into the object.

e. 选择第二个图像占位符并将其删除。

f. 选择图像按钮，然后输入您在步骤 1 中创建的另一个图像的 HTTPS URL。

4. Select预览以预览工作人员 UI。预览会在新标签页中打开，因此如果浏览器阻止弹出窗口，则可能需要手动启用该选项卡才能打开。向预览添加一个或多个批注然后选择提交您可以看到注释将创建的输出数据的预览。

5. 配置并验证说明后，选择Create创建标记作业。

如果您使用了私人员工队伍，则可以导航到登录的员工门户第 3 步：选择员工 (p. 208)以查看你的标签任务。此任务可能需要几分钟才能显示。

下一页

第 5 步：监控标记 Job (p. 210)

第 5 步：监控标记 Job

在创建您的标记作业后，将会看到一个列表，其中包含创建的所有作业的列表。您可以使用此列表监视标记作业的状态。此列表包含以下字段：

- 名称— 您创建作业时为其分配的名称。
- 状态— 作业的完成状态。状态可以是以下值之一：完成、失败、正在进行或已停止。
- 标签对象/总数— 显示标记作业中的总对象数以及已标记了多少个对象。
- Creation time— 创建作业的日期和时间。

您也可以克隆、链接或停止作业。选择一个作业，然后从 Actions (操作) 菜单中选择以下选项之一：

- Clone— 使用从所选作业复制的配置创建新的标记作业。如果要更改并重新运行作业，您可以克隆该作业。例如，您可以克隆发送到私有人力的作业，以便您可以将其发送到 Amazon Mechanical Turk 人力。或者，您可以克隆一个作业，以便对与原始作业存储在相同位置的新数据集重新运行此作业。
- Chain— 创建新的标记作业，它可以基于停止、失败或完成的作业的数据和模型（如果有）而构建。有关使用案例以及如何使用该选项的更多信息，请参阅链接标记作业 (p. 444)。
- 停止— 停止正在运行的作业。您无法重新启动停止的作业。您可以克隆一个作业以重新启动作业，或链接作业从上次停止的位置继续执行。任何已标记对象的标签都将写入到输出文件位置。有关更多信息，请参阅输出数据 (p. 414)。

标签图片

使用 Ground Truth 标记图像。选择以下内置任务类型之一以了解有关该任务类型的详细信息。每个页面都包含帮助您使用该任务类型创建标记作业的说明。

> Tip
>
> 要详细了解支持的文件类型和输入数据配额，请参阅输入数据 (p. 378).

主题

- Bounding Box (p. 211)
- 图像语义分割 (p. 215)
- 图像分类（单个标签）(p. 218)
- 图像分类（多标签）(p. 221)

- 图像标签验证 (p. 223)

Bounding Box

用于训练机器学习模型的映像通常包含多个对象。要对图像中的一个或多个对象进行分类和本地化，请使用 Amazon SageMaker Ground Truth 边界框标记作业任务类型。在这种情况下，本地化意味着边界框的像素位置。

您可以使用 Amazon SageMaker 控制台的 “Ground Truth” 部分或`CreateLabelingJob`operation.

> Important
>
> 对于此任务类型，如果您创建自己的清单文件，请使用`"source-ref"`以识别 Amazon S3 中您要标记的每个图像文件的位置。有关更多信息，请参阅 输入数据 (p. 378)。

创建边界框标记作业（控制台）

你可以按照说明进行操作创建标记作业（控制台） (p. 353)以了解如何在 SageMaker 控制台中创建边界框标记作业。在步骤 10 中，选择Image来自 的任务类别下拉菜单，然后选择边界框作为任务类型。

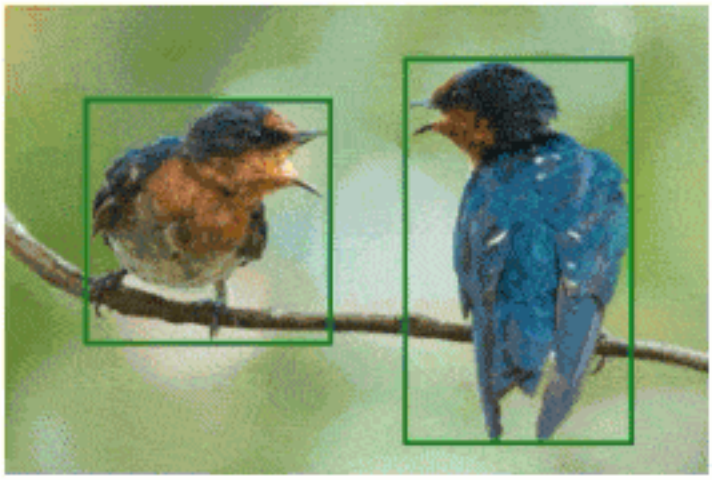
为标记任务提供类似于以下内容的工作人员 UI。在您使用控制台创建标记作业时，需要指定说明，以帮助工作人员完成作业和工作人员可以从中选择的多达 50 个标签。

Instructions | Shortcuts | Select a category from the list and draw a box a

Instructions

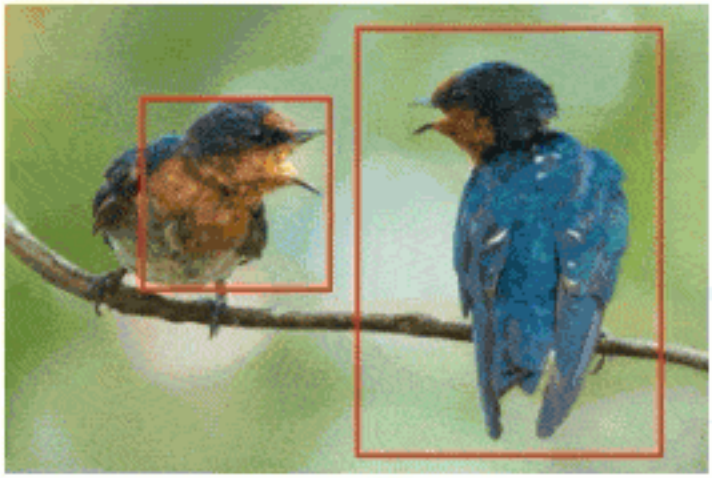
Good example

Fit each box tightly around the boundaries of the object.



Bad example

Boxes should not overlap with the boundaries of objects.



More Instructions

创建边界框标记作业 (API)

要创建边界框标记作业，请使用 SageMaker API 操作。`CreateLabelingJob`. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅部分`CreateLabelingJob`.

请按照创建标记作业 (API) (p. 356)中的说明进行操作，并在配置请求时执行以下操作：

- 此任务类型的注释前 Lambda 函数以结尾`PRE-BoundingBox`. 要查找您的区域的注释前 Lambda ARN，请参阅PreHumanTaskLambdaArn.
- 此任务类型的注释合并 Lambda 函数以结尾`ACS-BoundingBox`. 要查找您的区域的注释合并 Lambda ARN，请参阅AnnotationConsolidationLambdaArn.

以下是一个示例：AmazonPython 软件开发工具包 (Boto3) 请求在美国东部（弗吉尼亚北部）区域创建标记作业。所有红色参数都应替换为您的规范和资源。

```
response = client.create_labeling_job(
    LabelingJobName='example-bounding-box-labeling-job',
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://bucket/path/manifest-with-input-data.json'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/file-to-store-output-data',
        'KmsKeyId': 'string'
    },
    RoleArn='arn:aws:iam::*:role/*',
    LabelCategoryConfigS3Uri='s3://bucket/path/label-categories.json',
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:region:*:workteam/private-crowd/*',
        'UiConfig': {
            'UiTemplateS3Uri': 's3://bucket/path/worker-task-template.html'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda:us-east-1:432418664414:function:PRE-
BoundingBox',
        'TaskKeywords': [
            'Bounding Box',
        ],
        'TaskTitle': 'Bounding Box task',
        'TaskDescription': 'Draw bounding boxes around objects in an image',
        'NumberOfHumanWorkersPerDataObject': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'MaxConcurrentTaskCount': 123,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-BoundingBox'
        },
    Tags=[
```

```
        {
            'Key': 'string',
            'Value': 'string'
        },
    ]
)
```

为边界框标记作业提供模板

如果要使用 API 创建标记作业，必须在 `UiTemplateS3Uri` 中提供工作人员任务模板。复制并修改以下模板。仅修改 `short-instructions`、`full-instructions` 和 `header`。将此模板上传到 S3，并在 `UiTemplateS3Uri` 中为此文件提供 S3 URI。

边界框输出数据

创建了边框标签作业后，您的输出数据将位于`S3OutputPath`使用 API 时的参数或输出数据集位置字段的Job 概述控制台的部分。

例如，成功完成的单类边界框任务的输出清单文件将包含以下内容：

```
[
  {
    "boundingBox": {
      "boundingBoxes": [
        {
          "height": 2832,
          "label": "bird",
          "left": 681,
          "top": 599,
          "width": 1364
        }
      ],
      "inputImageProperties": {
        "height": 3726,
        "width": 2662
      }
    }
  }
]
```

`boundingBoxes` 参数标识围绕标识为“鸟”的对象绘制的边界框相对于图像左上角的位置，该位置被认为是 (0,0) 像素坐标。在上一个示例中，**`left`** 和 **`top`** 确定边界框左上角的像素相对于图像左上角的位置。边界框的尺寸使用 **`height`** 和 **`width`** 进行标识。`inputImageProperties` 参数给出原始输入图像的像素尺寸。

当您使用边界框任务类型时，您可以创建单类和多类边界框标记作业。成功完成的多类边界框的输出清单文件将包含以下内容：

```
[
  {
    "boundingBox": {
      "boundingBoxes": [
        {
          "height": 938,
          "label": "squirrel",
          "left": 316,
          "top": 218,
          "width": 785
        },
        {
          "height": 825,
```

```
          "label": "rabbit",
          "left": 1930,
          "top": 2265,
          "width": 540
        },
        {
          "height": 1174,
          "label": "bird",
          "left": 748,
          "top": 2113,
          "width": 927
        },
        {
          "height": 893,
          "label": "bird",
          "left": 1333,
          "top": 847,
          "width": 736
        }
      ],
      "inputImageProperties": {
        "height": 3726,
        "width": 2662
      }
    }
  }
]
```


要了解有关边界框标记作业产生的输出清单文件的更多信息，请参阅边界框作业输出 (p. 419)。

要详细了解 Ground Truth 生成的输出清单文件以及 Ground Truth 用于存储输出数据的文件结构，请参阅。输出数据 (p. 414).

图像语义分割

要在像素级别识别图像内容，请使用 Amazon SageMaker Ground Truth 语义分割标记任务。在分配语义分割标记作业时，工作人员会将图像中的像素分类为一组预定义的标签或类。Ground Truth 支持单类和多类语义分段标记作业。

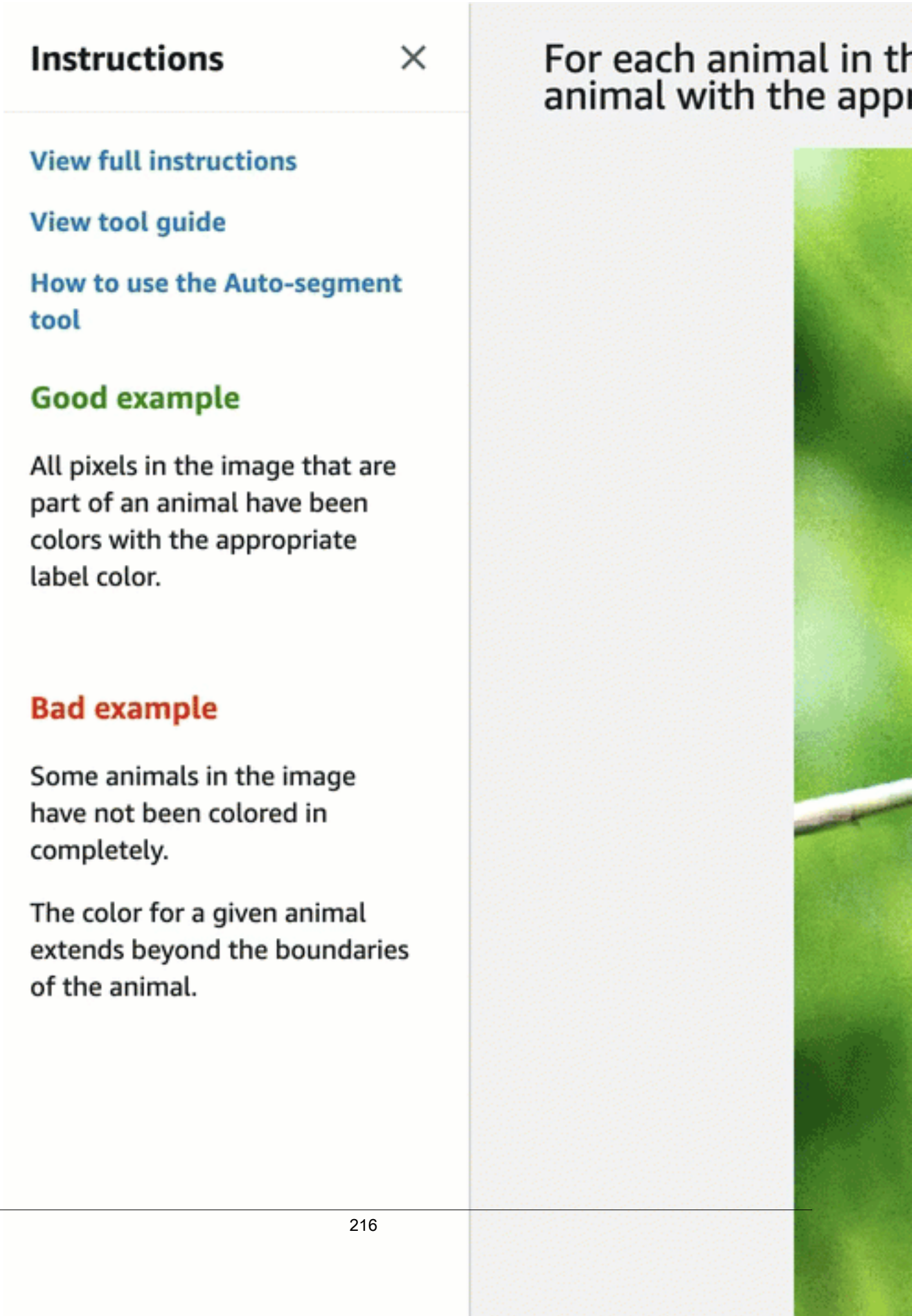
您可以使用 Amazon SageMaker 控制台的 “Ground Truth” 部分或`CreateLabelingJob`operation.

> Important
>
> 对于此任务类型，如果您创建自己的清单文件，请使用`"source-ref"`以识别 Amazon S3 中您要标记的每个图像文件的位置。有关更多信息，请参阅 输入数据 (p. 378)。

创建语义分段标记作业（控制台）

要按照中的说明进行操作创建标记作业（控制台） (p. 353)以了解如何在 SageMaker 控制台中创建语义分割标记作业。在步骤 10 中，选择Image来自 的任务类别选择下拉菜单，然后选择语义分割作为任务类型。

为标记任务提供类似于以下内容的工作人员 UI。使用控制台创建标记作业时，需要指定说明，以帮助工作人员完成工作人员可以从中选择的作业和标签。


Instructions
View full instructions
View tool guide
How to use the Auto-segment tool
Good example
All pixels in the image that are part of an animal have been colors with the appropriate label color.
Bad example
Some animals in the image have not been colored in completely.
The color for a given animal extends beyond the boundaries of the animal.
For each animal in th
animal with the appr

创建语义分段标记作业 (API)

要创建语义分割标记作业，请使用 SageMaker API 操作。`CreateLabelingJob`. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅部分的`CreateLabelingJob`.

请按照创建标记作业 (API) (p. 356)中的说明进行操作，并在配置请求时执行以下操作：

- 此任务类型的注释前 Lambda 函数以结尾。`PRE-SemanticSegmentation`. 要查找您的区域的注释前 Lambda ARN，请参阅。PreHumanTaskLambdaArn.
- 此任务类型的注释合并 Lambda 函数以结尾。`ACS-SemanticSegmentation`. 要查找您的区域的注释合并 Lambda ARN，请参阅。AnnotationConsolidationLambdaArn.

以下是一个示例：AmazonPython SDK (Boto3) 请求在美国东部（弗吉尼亚北部）区域创建标记作业。所有红色参数都应替换为您的规范和资源。

```
response = client.create_labeling_job(
    LabelingJobName='example-semantic-segmentation-labeling-job,
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://bucket/path/manifest-with-input-data.json'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/file-to-store-output-data',
        'KmsKeyId': 'string'
    },
    RoleArn='arn:aws:iam::*:role/*,
    LabelCategoryConfigS3Uri='s3://bucket/path/label-categories.json',
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:region:*:workteam/private-crowd/*',
        'UiConfig': {
            'UiTemplateS3Uri': 's3://bucket/path/worker-task-template.html'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda:us-east-1:432418664414:function:PRE-
SemanticSegmentation,
        'TaskKeywords': [
            'Semantic Segmentation',
        ],
        'TaskTitle': 'Semantic segmentation task',
        'TaskDescription': 'For each category provided, segment out each relevant object
 using the color associated with that category',
        'NumberOfHumanWorkersPerDataObject': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'MaxConcurrentTaskCount': 123,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-SemanticSegmentation'
        },
```

```
    Tags=[
        {
            'Key': 'string',
            'Value': 'string'
        },
    ]
)
```

为语义分段标记作业提供模板

如果要使用 API 创建标记作业，必须在 `UiTemplateS3Uri` 中提供工作人员任务模板。复制并修改以下模板。仅修改 `short-instructions`、`full-instructions` 和 `header`。

将此模板上传到 S3，并在 `UiTemplateS3Uri` 中为此文件提供 S3 URI。

语义分割输出数据

一旦您创建了语义分段标签任务，您的输出数据将位于`S3OutputPath`使用 API 时的参数或在输出数据集位置字段的Job 概述控制台的部分。

要详细了解 Ground Truth 生成的输出清单文件以及 Ground Truth 用于存储输出数据的文件结构，请参阅。输出数据 (p. 414).

要查看语义分段标记作业的输出清单文件示例，请参阅 3D 点云语义分割输出 (p. 429)。

图像分类（单个标签）

当您需要员工使用您指定的预定义标签对图像进行分类时，请使用 Amazon SageMaker Ground Truth 图像分类标记任务。将向工作人员显示图像，并要求工作人员为每个图像选择一个标签。

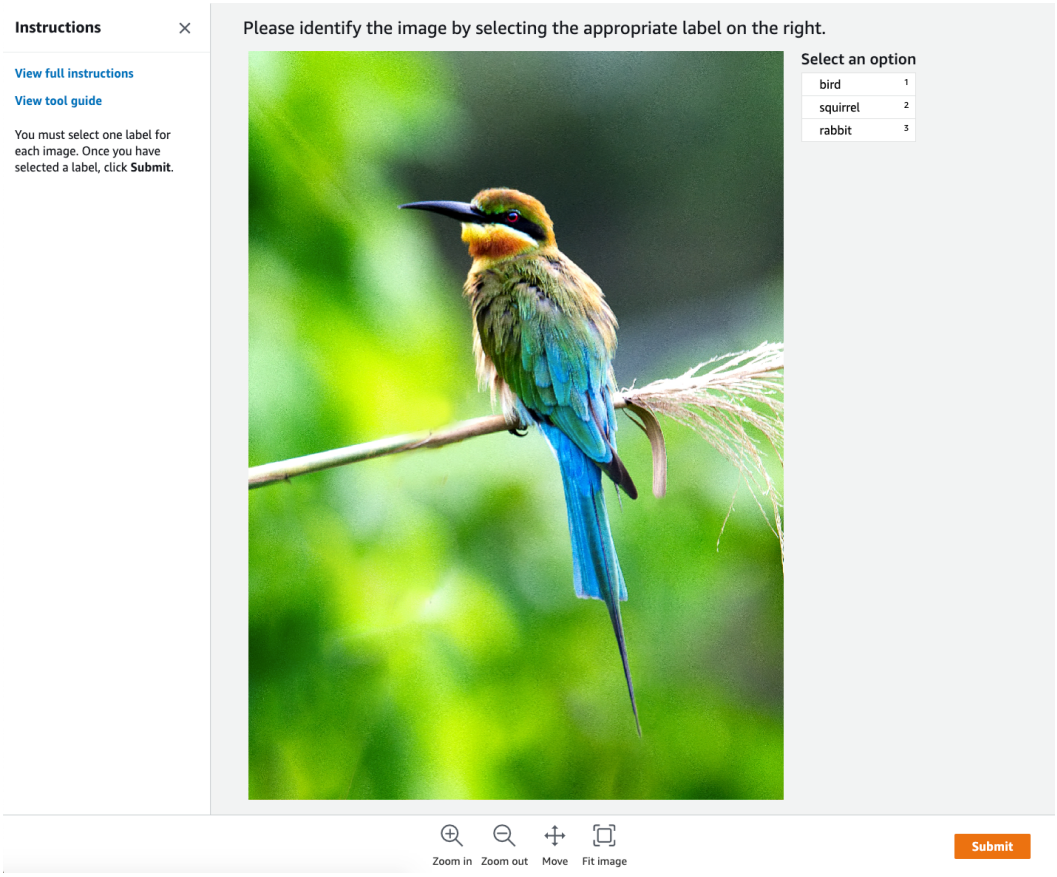
您可以使用 Amazon SageMaker 控制台的 “Ground Truth” 部分或`CreateLabelingJob`operation.

> Important
>
> 对于此任务类型，如果您创建自己的清单文件，请使用`"source-ref"`以识别 Amazon S3 中您要标记的每个图像文件的位置。有关更多信息，请参阅 输入数据 (p. 378)。

创建图像分类标记作业（控制台）

你可以按照中的说明操作创建标记作业（控制台） (p. 353)以了解如何在 SageMaker 控制台中创建图像分类标记作业。在步骤 10 中，选择Image来自 的任务类别然后选择下拉菜单，然后选择图像分类（单个标签）作为任务类型。

为标记任务提供类似于以下内容的工作人员 UI。使用控制台创建标记作业时，需要指定说明，以帮助工作人员完成工作人员可以从中选择的作业和标签。



创建图像分类标记作业 (API)

要创建图像分类标记作业，请使用 SageMaker API 操作。`CreateLabelingJob`. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅部分`CreateLabelingJob`.

请按照创建标记作业 (API) (p. 356)中的说明进行操作，并在配置请求时执行以下操作：

- 此任务类型的注释前 Lambda 函数以结尾。`PRE-ImageMultiClass`. 要查找您所在区域的注释前 Lambda ARN，请参阅PreHumanTaskLambdaArn.
- 此任务类型的注释合并 Lambda 函数以结尾。`ACS-ImageMultiClass`. 要查找您所在区域的注释合并 Lambda ARN，请参阅AnnotationConsolidationLambdaArn.

以下是一个示例：AmazonPython SDK (Boto3) 请求以在美国东部（弗吉尼亚北部）区域创建标记作业。所有红色参数都应替换为您的规范和资源。

```
response = client.create_labeling_job(
    LabelingJobName='example-image-classification-labeling-job',
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://bucket/path/manifest-with-input-data.json'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
```

```
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/file-to-store-output-data',
        'KmsKeyId': 'string'
    },
    RoleArn='arn:aws:iam::*:role/*',
    LabelCategoryConfigS3Uri='s3://bucket/path/label-categories.json',
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:region:*:workteam/private-crowd/*',
        'UiConfig': {
            'UiTemplateS3Uri': 's3://bucket/path/worker-task-template.html'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda:us-east-1:432418664414:function:PRE-
ImageMultiClass,
        'TaskKeywords': [
            Image classification',
        ],
        'TaskTitle': Image classification task',
        'TaskDescription': 'Carefully inspect the image and classify it by selecting one
 label from the categories provided.',
        'NumberOfHumanWorkersPerDataObject': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'MaxConcurrentTaskCount': 123,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-ImageMultiClass'
        },
    Tags=[
        {
            'Key': 'string',
            'Value': 'string'
        },
    ]
)
```

为图像分类标记作业提供模板

如果要使用 API 创建标记作业，必须在 `UiTemplateS3Uri` 中提供工作人员任务模板。复制并修改以下模板。仅修改 `short-instructions`、`full-instructions` 和 `header`。

将此模板上传到 S3，并在 `UiTemplateS3Uri` 中为此文件提供 S3 URI。

图像分类输出数据

创建映像分类标签任务后，输出数据将位于`S3OutputPath`使用 API 时的参数或在输出数据集位置字段的Job 概述控制台的部分。

要详细了解 Ground Truth 生成的输出清单文件以及 Ground Truth 用于存储输出数据的文件结构，请参阅。输出数据 (p. 414).

要查看来自图像分类标记作业的输出清单文件示例，请参阅 分类作业输出 (p. 417)。

图像分类（多标签）

当您需要工作人员对图像中的多个对象进行分类时，请使用 Amazon SageMaker Ground Truth 多标签图像分类标记任务。例如，下图显示了一只狗和一只猫。您可以使用多标签图像分类将标签“dog”和“cat”与此图像相关联。

在处理多标签图像分类任务时，工作人员应选择所有适用的标签，且必须选择至少一个标签。使用此任务类型创建作业时，您最多可提供 50 个标签类别。

在控制台中创建标记作业时，Ground Truth 不会针对没有任何标签应用于图像的情况提供 “无” 类别。要向工作人员提供此选项，请在创建多标签图像分类作业时，包含类似于“无”或“其他”的标签。

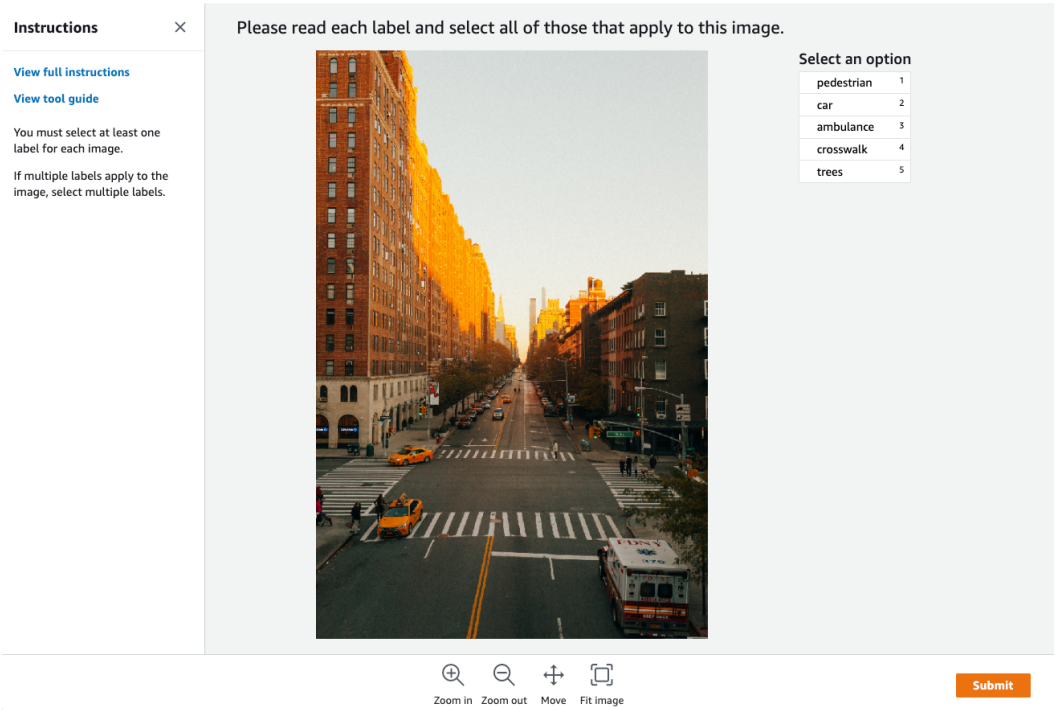
要限制工作人员为每个图像选择单个标签，请使用 图像分类（单个标签）(p. 218) 任务类型。

> Important
>
> 对于此任务类型，如果您创建自己的清单文件，请使用`"source-ref"`识别 Amazon S3 中您要标记的每个图像文件的位置。有关更多信息，请参阅 输入数据 (p. 378)。

创建多标签图像分类标记作业（控制台）

你可以按照中的说明创建标记作业（控制台）(p. 353)以了解如何在 SageMaker 控制台中创建多标签图像分类标记作业。在步骤 10 中，选择Image来自 的任务类别然后选择下拉菜单图像分类（多标签）作为任务类型。

为标记任务提供类似于以下内容的工作人员 UI。在控制台中创建标记作业时，需要指定说明，以帮助工作人员完成工作人员可以从中选择的作业和标签。



创建多标签图像分类标记作业 (API)

要创建多标签图像分类标记作业，请使用 SageMaker API 操作。`CreateLabelingJob`. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅部分`CreateLabelingJob`.

请按照创建标记作业 (API) (p. 356)中的说明进行操作，并在配置请求时执行以下操作：

- 此任务类型的注释前 Lambda 函数以结尾。`PRE-ImageMultiClassMultiLabel`. 要查找您所在区域的注释前 Lambda ARN，请参阅PreHumanTaskLambdaArn.
- 此任务类型的注释合并 Lambda 函数以结尾`ACS-ImageMultiClassMultiLabel`. 要查找您所在区域的注释合并 Lambda ARN，请参阅AnnotationConsolidationLambdaArn.

以下是一个示例：AmazonPython 软件开发工具包 (Boto3) 请求要在美国东部（弗吉尼亚北部）区域创建标记作业。所有红色参数都应替换为您的规范和资源。

```
response = client.create_labeling_job(
    LabelingJobName='example-multi-label-image-classification-labeling-job',
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://bucket/path/manifest-with-input-data.json'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/file-to-store-output-data',
```

```
        'KmsKeyId': 'string'
    },
    RoleArn='arn:aws:iam::*:role/*',
    LabelCategoryConfigS3Uri='s3://bucket/path/label-categories.json',
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:region:*:workteam/private-crowd/*',
        'UiConfig': {
            'UiTemplateS3Uri': 's3://bucket/path/worker-task-template.html'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda:us-east-1:432418664414:function:PRE-
ImageMultiClassMultiLabel',
        'TaskKeywords': [
            'Image Classification',
        ],
        'TaskTitle': 'Multi-label image classification task',
        'TaskDescription': 'Select all labels that apply to the images shown',
        'NumberOfHumanWorkersPerDataObject': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'MaxConcurrentTaskCount': 123,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-ImageMultiClassMultiLabel'
        },
    Tags=[
        {
            'Key': 'string',
            'Value': 'string'
        },
    ]
)
```

为多标签图像分类提供模板

如果要使用 API 创建标记作业，必须在 `UiTemplateS3Uri` 中提供工作人员任务模板。复制并修改以下模板。仅修改 `short-instructions`、`full-instructions` 和 `header`。

将此模板上传到 S3，并在 `UiTemplateS3Uri` 中为此文件提供 S3 URI。

多标签图像分类输出数据

创建多标签图像分类标签任务后，您的输出数据将位于`S3OutputPath`使用 API 时的参数或在输出数据集位置字段的Job 概述控制台的部分。

要详细了解 Ground Truth 生成的输出清单文件以及 Ground Truth 用于存储输出数据的文件结构，请参阅。输出数据 (p. 414).

要查看多标签图像分类标记作业的输出清单文件示例，请参阅多标签分类作业输出 (p. 418)。

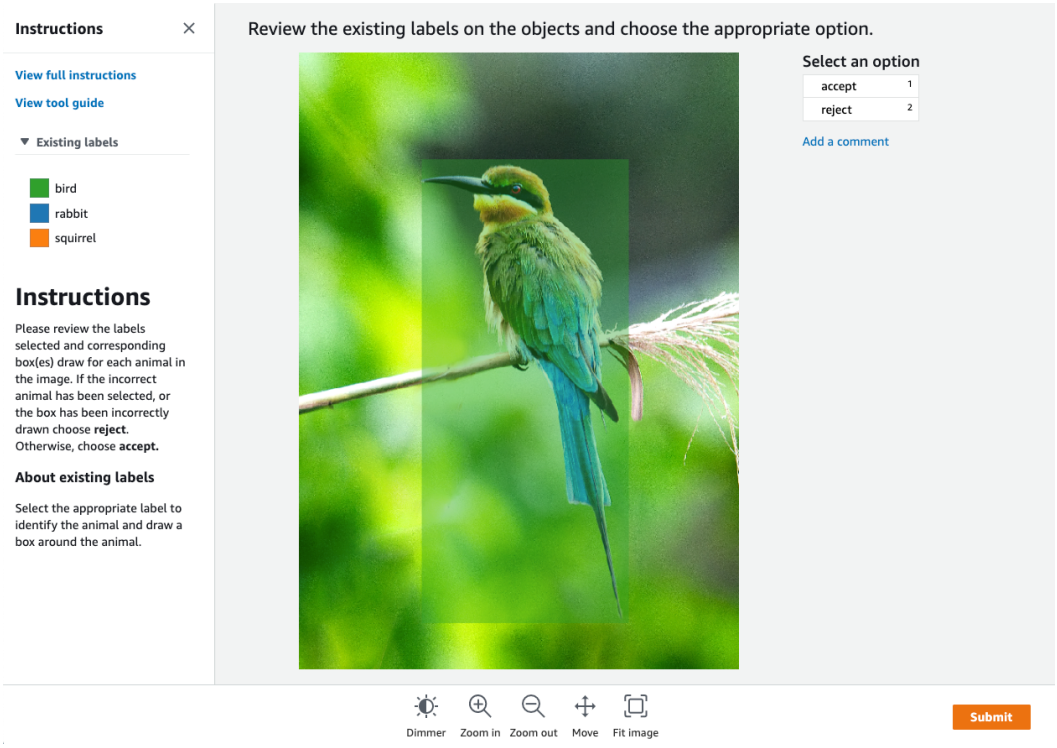
图像标签验证

为机器学习 (ML) 算法构建高度精确的训练数据集是一个迭代过程。通常情况下，您应当查看并持续调整标签，直到您满意它们准确表示基本实际情况或可在现实世界中直接观察到的内容为止。

您可以使用 Amazon SageMaker Ground Truth 图像标签验证任务指示工作人员查看数据集标签并提高标签准确性。工作人员可以指示现有标签是否正确或评估标签质量。他们也可以添加评论来解释其原因。Amazon SageMaker Ground Truth 支持对Bounding Box (p. 211)和图像语义分割 (p. 215)标签。

您可以使用 Amazon SageMaker 控制台的 “Ground Truth” 部分或CreateLabelingJoboperation.

TGround Truth 标记任务提供类似于以下内容的工作人员控制台。使用控制台创建标记作业时，可以修改显示的图像和内容。要了解如何使用 Ground Truth 控制台创建标记作业，请参阅创建标记作业（控制台）(p. 353).



您可以使用 SageMaker 控制台或 API 创建标签验证标记作业。要了解如何使用 Ground Truth API 操作创建标记作业`CreateLabelingJob`，请参阅创建标记作业 (API) (p. 356).

使用 Ground Truth 标记文本

使用 Ground Truth 发短信。选择以下内置任务类型之一以了解有关该任务类型的详细信息。每个页面都包含帮助您使用该任务类型创建标记作业的说明。

> Tip
>
> 要详细了解受支持的文件类型和输入数据配额，请参阅输入数据 (p. 378).

主题

- 命名实体识别 (p. 224)
- 文本分类（单标签）(p. 228)
- 文本分类（多标签）(p. 230)

命名实体识别

要从非结构化文本中提取信息并将其分类为预定义类别，请使用 Amazon SageMaker Ground Truth 命名实体识别 (NER) 标记任务。传统上，NER 涉及筛选文本数据来定位名词短语（称为命名实体），并使用标签对每个短语进行分类，例如，“人”、“组织”或“品牌”。您可以扩展此任务以标记更长的文本跨度，并使用您指定的预定义标签对这些序列进行分类。

当负责命名实体识别标签作业时，工作人员会将标签应用于较大文本块中的特定单词或短语。他们选择一个标签，然后通过使用光标突出显示标签所应用的文本部分来应用该标签。Ground Truth 命名实体识别工具支

持重叠注释、上下文中的标签选择和单个突出显示的多标签选择。此外，工作人员可以使用键盘快速选择标签。

您可以使用 Amazon SageMaker 控制台的 “Ground Truth” 部分创建命名实体识别标记作业，或使用`CreateLabelingJob`operation.

Important

如果手动创建输入清单文件，请使用`"source"`以识别您要标记的文本。有关更多信息，请参阅 输入数据 (p. 378)。

创建命名实体识别标记作业（控制台）

你可以按照中的说明执行创建标记作业（控制台） (p. 353)以了解如何在 SageMaker 控制台中创建命名实体识别标记作业。在步骤 10 中，选择Text来自 的任务类别下拉菜单，然后选择命名实体识别作为任务类型。

为标记任务提供类似于以下内容的工作人员 UI。使用控制台创建标记作业时，需要指定说明，以帮助工作人员完成工作人员可以从中选择的作业和标签。

创建命名实体识别标记作业 (API)

要创建命名实体识别标记作业，请使用 SageMaker API 操作`CreateLabelingJob`. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅的部分`CreateLabelingJob`.

请按照创建标记作业 (API) (p. 356)中的说明进行操作，并在配置请求时执行以下操作：

- 此任务类型的注释前 Lambda 函数以结尾。`PRE-NamedEntityRecognition`. 要查找您所在区域的注释前 Lambda ARN，请参阅PreHumanTaskLambdaArn.
- 此任务类型的注释合并 Lambda 函数以结尾。`ACS-NamedEntityRecognition`. 要查找您所在区域的注释合并 Lambda ARN，请参阅AnnotationConsolidationLambdaArn.
- 您必须为提供以下 ARN`HumanTaskUiArn`：

```
arn:aws:sagemaker:aws-region:394669845002:human-task-ui/NamedEntityRecognition
```

Replace*aws-region*与您用于创建标记作业的 Amazon 区域一起执行。例如，使用`us-west-1`如果您在美国西部（加利福尼亚北部）创建标记作业。

- 使用在标签类别配置文件中提供工作人员说明`instructions`参数。您可以在`shortInstruction`和`fullInstruction`字段之间没有不同。有关更多信息，请参阅 在标签类别配置文件中提供工作人员说明 (p. 227)。

```
"instructions": {"shortInstruction":"<h1>Add header</h1><p>Add Instructions</p>",
 "fullInstruction":"<p>Add additional instructions.</p>"}
```

以下是的示例：AmazonPython SDK (Boto3) 请求以在美国东部（弗吉尼亚北部）区域创建标记作业。所有红色参数都应替换为您的规范和资源。

```
response = client.create_labeling_job(
    LabelingJobName='example-ner-labeling-job',
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://bucket/path/manifest-with-input-data.json'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/file-to-store-output-data',
        'KmsKeyId': 'string'
    },
    RoleArn='arn:aws:iam::*:role/*',
    LabelCategoryConfigS3Uri='s3://bucket/path/label-categories.json',
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:region:*:workteam/private-crowd/*',
        'UiConfig': {
            'HumanTaskUiArn': 'arn:aws:sagemaker:us-east-1:394669845002:human-task-ui/
NamedEntityRecognition'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda:us-east-1:432418664414:function:PRE-
NamedEntityRecognition',
        'TaskKeywords': [
            'Named entity Recognition',
        ],
        'TaskTitle': 'Named entity Recognition task',
        'TaskDescription': 'Apply the labels provided to specific words or phrases within
 the larger text block.',
        'NumberOfHumanWorkersPerDataObject': 1,
        'TaskTimeLimitInSeconds': 28800,
        'TaskAvailabilityLifetimeInSeconds': 864000,
        'MaxConcurrentTaskCount': 1000,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-NamedEntityRecognition'
```

```
        },
    Tags=[
        {
            'Key': 'string',
            'Value': 'string'
        },
    ]
)
```

在标签类别配置文件中提供工作人员说明

您必须在标签类别配置文件中提供工作人员说明`LabelCategoryConfigS3Uri`中的参数`CreateLabelingJob`. 您可以使用这些说明提供有关希望工作人员执行的任务的详细信息，并帮助他们有效地使用该工具。

你使用提供简短和长的说明`shortInstruction`和`fullInstruction`中的`instructions`参数。要详细了解这些指令类型，请参阅创建说明页 (p. 351).

以下是标签类别配置文件的示例，其中包含可用于命名实体识别标记作业的说明。

```
{
  "document-version": "2018-11-28",
  "labels": [
    {
      "label": "label1",
      "shortDisplayName": "L1"
    },
    {
      "label": "label2",
      "shortDisplayName": "L2"
    },
    {
      "label": "label3",
      "shortDisplayName": "L3"
    },
    {
      "label": "label4",
      "shortDisplayName": "L4"
    },
    {
      "label": "label5",
      "shortDisplayName": "L5"
    }
  ],
  "instructions": {
    "shortInstruction": "<p>Enter description of the labels that workers have
                        to choose from</p><br><p>Add examples to help workers understand
 the label</p>",
    "fullInstruction": "<ol>
                        <li><strong>Read</strong> the text carefully.</li>
                        <li><strong>Highlight</strong> words, phrases, or sections of the
 text.</li>
                        <li><strong>Choose</strong> the label that best matches what you
 have highlighted.</li>
                        <li>To <strong>change</strong> a label, choose highlighted text and
 select a new label.</li>
                        <li>To <strong>remove</strong> a label from highlighted text,
 choose the X next to the
                        abbreviated label name on the highlighted text.</li>
                        <li>You can select all of a previously highlighted text, but not a
 portion of it.</li>
                        </ol>"
  }
```

```
}
```

命名实体识别输出数据

创建命名实体识别标签作业后，您的输出数据将位于`S3OutputPath`使用 API 时的参数或在输出数据集位置字段中的Job 概述控制台的部分。

要详细了解由 Ground Truth 生成的输出清单文件以及用于存储输出数据的文件结构，请参阅。输出数据 (p. 414).

文本分类（单标签）

要将文章和文本分类为预定义类别，请使用文本分类。例如，您可以使用文本分类来识别评论中传达的情绪或文本部分背后的情绪。您可以使用 Amazon SageMaker Ground Truth 文本分类可让员工按您定义的类别对文本进行排序

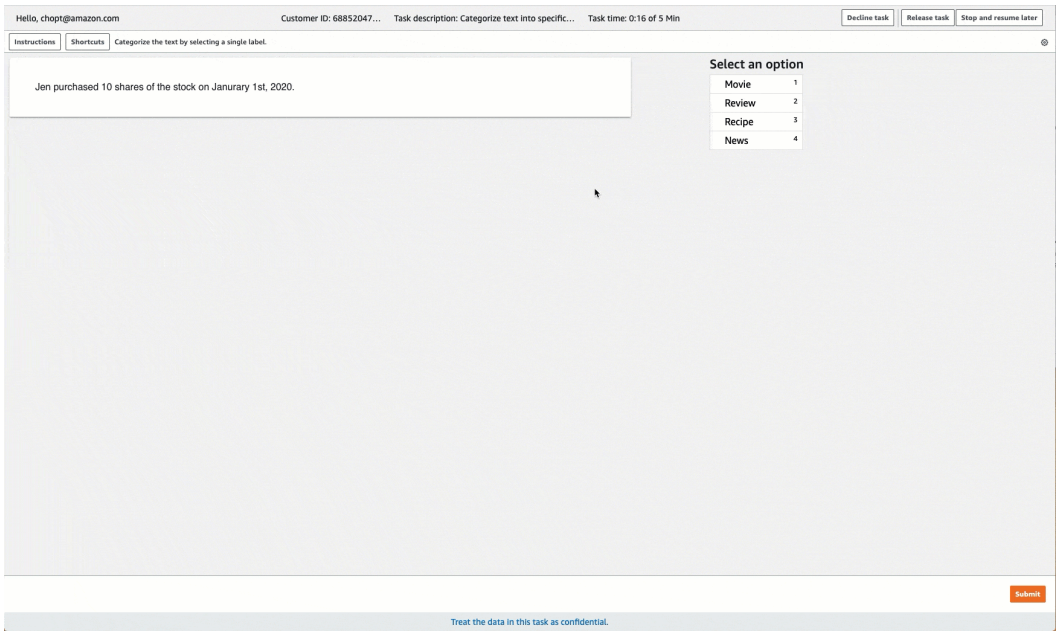
您可以使用 Amazon SageMaker 控制台的 “Ground Truth” 部分或`CreateLabelingJob`operation.

> **Important**
>
> 如果手动创建输入清单文件，请使用`"source"`以识别要标记的文本。有关更多信息，请参阅 输入数据 (p. 378)。

创建文本分类标记作业（控制台）

你可以按照中的说明创建标记作业（控制台） (p. 353)以了解如何在 SageMaker 控制台中创建文本分类标记作业。在步骤 10 中，选择Text来自 的任务类别下拉菜单，然后选择文本分类（单标签）作为任务类型。

Ground Truth 为标记任务提供类似于以下内容的工作人员 UI。使用控制台创建标记作业时，需要指定说明，以帮助工作人员完成工作人员可以从中选择的作业和标签。



创建文本分类标记作业 (API)

要创建文本分类标记作业，请使用 SageMaker API 操作。`CreateLabelingJob`. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅部分`CreateLabelingJob`.

请按照创建标记作业 (API) (p. 356)中的说明进行操作，并在配置请求时执行以下操作：

- 此任务类型的注释前 Lambda 以结尾。PRE-TextMultiClass. 要查找您的区域的注释前 Lambda ARN，请参阅PreHumanTaskLambdaArn.
- 此任务类型的注释合并 Lambda 函数以结尾ACS-TextMultiClass. 要查找您所在区域的注释合并 Lambda ARN，请参阅AnnotationConsolidationLambdaArn.

以下是一个示例AmazonPython 软件开发工具包 (Boto3) 请求以在美国东部（弗吉尼亚北部）区域创建标记作业。所有红色参数都应替换为您的规范和资源。

```
response = client.create_labeling_job(
    LabelingJobName='example-text-classification-labeling-job',
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://bucket/path/manifest-with-input-data.json'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/file-to-store-output-data',
        'KmsKeyId': 'string'
    },
    RoleArn='arn:aws:iam::*:role/*',
    LabelCategoryConfigS3Uri='s3://bucket/path/label-categories.json',
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:region:*:workteam/private-crowd/*',
        'UiConfig': {
            'UiTemplateS3Uri': 's3://bucket/path/worker-task-template.html'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda:us-east-1:432418664414:function:PRE-
TextMultiClass,
        'TaskKeywords': [
            Text classification',
        ],
        'TaskTitle': Text classification task',
        'TaskDescription': 'Carefully read and classify this text using the categories
 provided.',
        'NumberOfHumanWorkersPerDataObject': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'MaxConcurrentTaskCount': 123,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-TextMultiClass'
        },
    Tags=[
        {
            'Key': 'string',
            'Value': 'string'
        },
    ]
```

```
)
```

为文本分类标记作业提供模板

如果要使用 API 创建标记作业，必须在 `UiTemplateS3Uri` 中提供工作人员任务模板。复制并修改以下模板。仅修改 `short-instructions`、`full-instructions` 和 `header`。

将此模板上传到 S3，并在 `UiTemplateS3Uri` 中为此文件提供 S3 URI。

文本分类输出数据

创建文本分类标签作业后，您的输出数据将位于`S3OutputPath`使用 API 时的参数或在输出数据集位置字段的Job 概述控制台的部分。

要详细了解 Ground Truth 生成的输出清单文件以及用于存储输出数据的文件结构，请参阅输出数据 (p. 414).

要查看来自文本分类标记作业的输出清单文件示例，请参阅 分类作业输出 (p. 417)。

文本分类（多标签）

要将文章和文本分类为多个预定义类别，请使用多标签文本分类任务类型。例如，您可以使用此任务类型来识别在文本中传达的多种情感。

在处理多标签文本分类任务时，工作人员应选择所有适用的标签，且必须至少选择一个标签。使用此任务类型创建作业时，您最多可提供 50 个标签类别。

在没有任何标签适用时，Amazon SageMaker Ground Truth 不会针对未向其提供 “无” 类别。要向工作人员提供此选项，请在创建多标签文本分类作业时，包含类似于“无”或“其他”的标签。

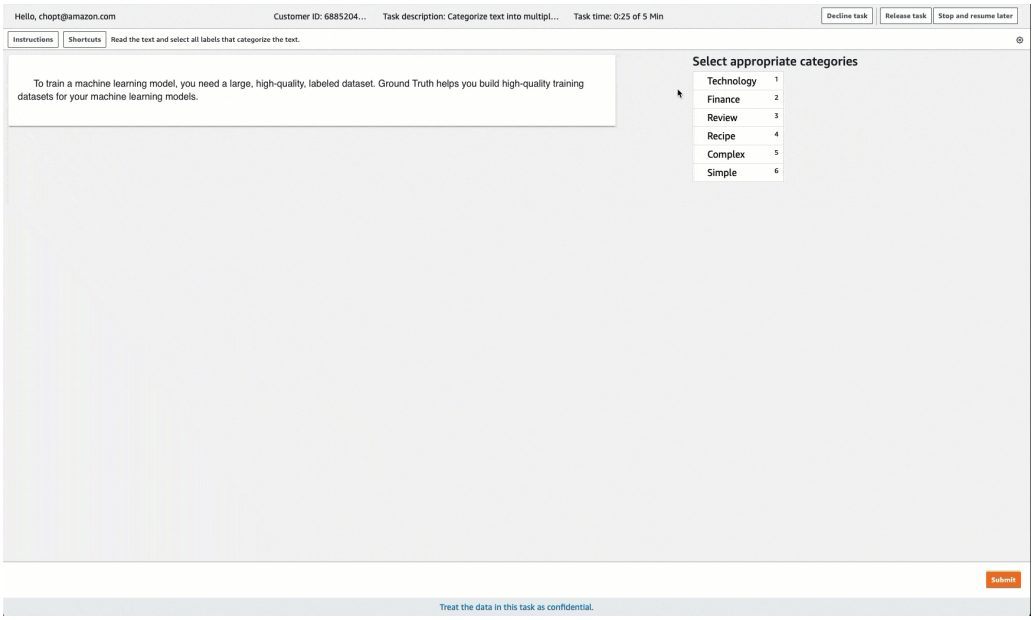
若要限制工作人员为每个文档或文本选择单个标签，请使用 文本分类（单标签） (p. 228) 任务类型。

> **Important**
>
> 如果手动创建输入清单文件，请使用`"source"`以识别要标记的文本。有关更多信息，请参阅 输入数据 (p. 378)。

创建多标签文本分类标记作业（控制台）

要按照中的说明进行操作创建标记作业（控制台） (p. 353)要了解如何在 Amazon SageMaker 控制台中创建多标签文本分类标记作业。在步骤 10 中，选择Text来自 的任务类别选择下拉菜单，然后选择文本分类（多标签）作为任务类型。

TGround Truth 提供类似于以下内容的工作人员 UI，以用于标记任务。使用控制台创建标记作业时，需要指定说明，以帮助工作人员完成工作人员可以从中选择的作业和标签。



创建多标签文本分类标记作业 (API)

要创建多标签文本分类标记作业，请使用 SageMaker API 操作。`CreateLabelingJob`. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅部分`CreateLabelingJob`.

请按照创建标记作业 (API) (p. 356)中的说明进行操作，并在配置请求时执行以下操作：

- 此任务类型的注释前 Lambda 函数以结尾`PRE-TextMultiClassMultiLabel`. 要查找您所在区域的注释前 Lambda ARN，请参阅PreHumanTaskLambdaArn.
- 此任务类型的注释合并 Lambda 函数以结尾`ACS-TextMultiClassMultiLabel`. 要查找您所在区域的注释合并 Lambda ARN，请参阅AnnotationConsolidationLambdaArn.

以下是一个示例。AmazonPython SDK (Boto3) 请求在美国东部（弗吉尼亚北部）区域创建标记作业。所有红色参数都应替换为您的规范和资源。

```
response = client.create_labeling_job(
    LabelingJobName='example-multi-label-text-classification-labeling-job',
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://bucket/path/manifest-with-input-data.json'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/file-to-store-output-data',
        'KmsKeyId': 'string'
    },
    RoleArn='arn:aws:iam::*:role/*',
    LabelCategoryConfigS3Uri='s3://bucket/path/label-categories.json',
```

```
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:region:*:workteam/private-crowd/*',
        'UiConfig': {
            'UiTemplateS3Uri': 's3://bucket/path/custom-worker-task-template.html'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda::function:PRE-TextMultiClassMultiLabel,
        'TaskKeywords': [
            'Text Classification',
        ],
        'TaskTitle': 'Multi-label text classification task',
        'TaskDescription': 'Select all labels that apply to the text shown',
        'NumberOfHumanWorkersPerDataObject': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'MaxConcurrentTaskCount': 123,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-TextMultiClassMultiLabel'
        },
    Tags=[
        {
            'Key': 'string',
            'Value': 'string'
        },
    ]
)
```

为多标签文本分类创建模板

如果要使用 API 创建标记作业，必须在 `UiTemplateS3Uri` 中提供工作人员任务模板。复制并修改以下模板。仅修改 `short-instructions`、`full-instructions` 和 `header`。

将此模板上传到 S3，并在 `UiTemplateS3Uri` 中为此文件提供 S3 URI。

要了解如何创建自定义模板，请参阅 创建自定义标记工作流 (p. 325)。

多标签文本分类输出数据

创建多标签文本分类标签任务后，您的输出数据将位于`S3OutputPath`使用 API 时的参数或输出数据集位置字段中的Job 概述控制台的部分。

要详细了解 Ground Truth 生成的输出清单文件以及 Ground Truth 用于存储输出数据的文件结构，请参阅输出数据 (p. 414).

要查看多标签文本分类标记作业的输出清单文件示例，请参阅多标签分类作业输出 (p. 418)。

为视频和视频帧添加标签

您可以使用 Ground Truth 对视频进行分类并使用三种内置视频任务类型之一对视频帧（从视频中提取的静止图像）进行注释。这些任务类型简化了使用 Amazon SageMaker 控制台、API 和语言特定的开发工具包创建视频和视频帧标记作业的过程。

- 视频剪辑分类 — 使员工能够将视频分类为您指定的类别。例如，您可以使用该任务类型让工作人员将视频分为体育、喜剧、音乐和教育等主题。要了解更多信息，请参阅 视频分类 (p. 233)。
- 视频帧标签作业 — 使工作人员能够使用边界框、折线、多边形或关键点注释工具对从视频中提取的视频帧进行注释。Ground Truth 提供两种内置任务类型来标记视频帧：

- 视频帧物体检测：使工作人员能够识别和定位视频帧中的对象。
- 视频帧对象跟踪：使工作人员能够跟踪对象在视频帧之间的移动。
- 视频帧调整工作：让工作人员调整以前的视频帧对象检测或对象跟踪标签作业中的标签、标签类别属性和帧属性。
- 视频帧验证工作：让工作人员验证以前的视频帧对象检测或对象跟踪标签作业中的标签、标签类别属性和帧属性。

如果您有视频文件，则可以使用 Ground Truth 自动帧提取工具从视频中提取视频帧。要了解更多信息，请参阅 视频帧输入数据 (p. 409)。

Tip

要详细了解支持的文件类型和输入数据配额，请参阅输入数据 (p. 378).

主题

- 视频分类 (p. 233)
- 标记视频帧 (p. 236)
- 工作人员说明 (p. 248)

视频分类

当您需要工作人员使用您指定的预定义标签对视频进行分类时，请使用 Amazon SageMaker Ground Truth 视频分类标记任 工作人员将看到视频，并要求工作人员为每个视频选择一个标签。

您可以使用 Amazon SageMaker 控制台的 “Ground Truth” 部分或CreateLabelingJoboperation.

您的视频文件必须以标记数据的工作团队使用的浏览器支持的格式进行编码。建议您使用 worker UI 预览验证输入清单文件中的所有视频文件格式是否正确显示。您可以使用工作人员说明将受支持的浏览器传达给工作 要查看支持的文件格式，请参阅支持的数据格式 (p. 380).

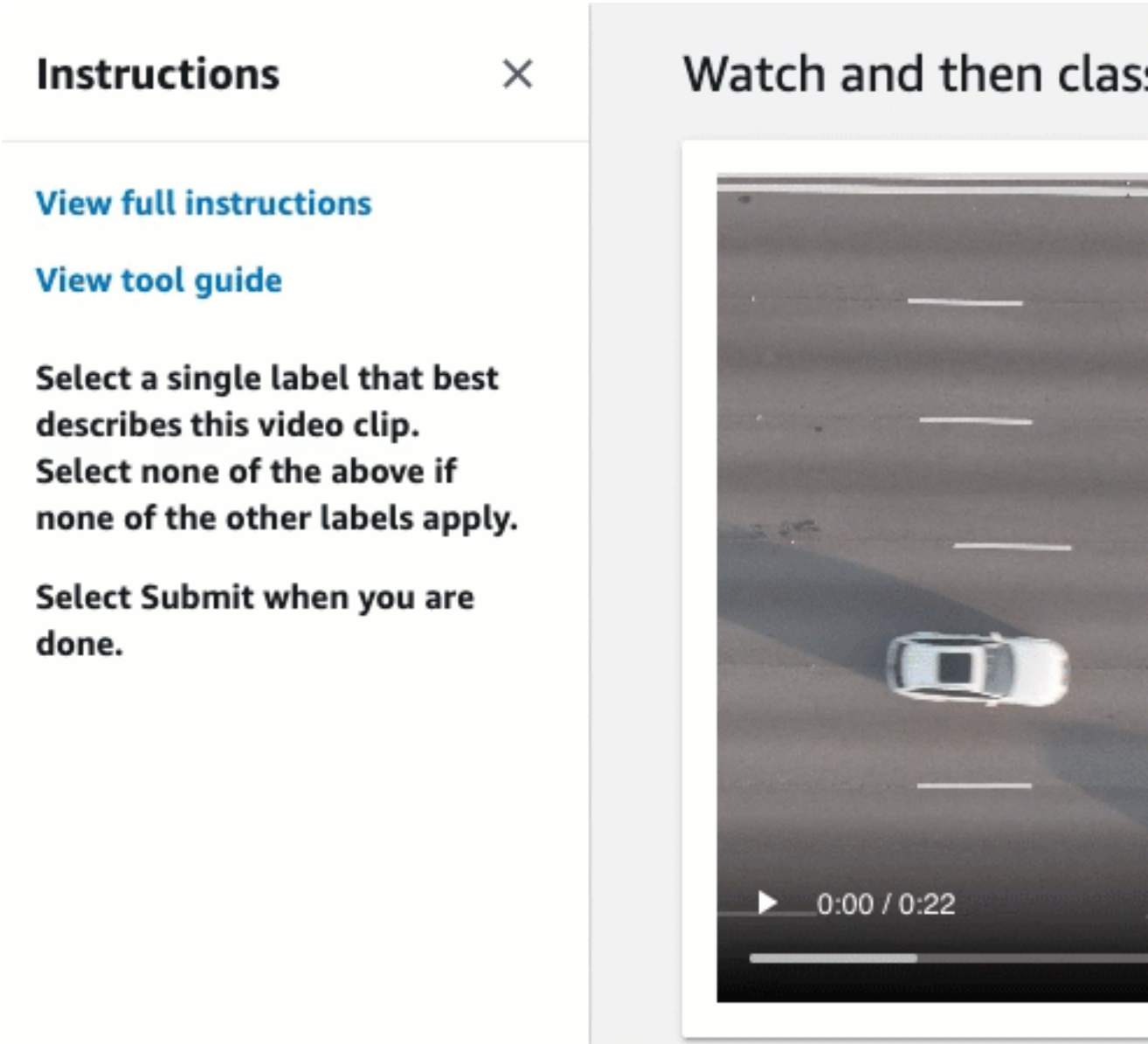
Important

对于此任务类型，如果您创建自己的清单文件，请使用`"source-ref"`以识别 Amazon S3 中您要标记的每个视频文件的位置。有关更多信息，请参阅 输入数据 (p. 378)。

创建视频分类标记 Job (控制台)

您可以按照中的说明进行操作创建标记作业（控制台） (p. 353)以了解如何在 SageMaker 控制台中创建视频分类标记作业。在步骤 10 中，选择视频来自 的任务类别下拉列表，然后选择视频分类作为任务类型。

Ground Truth 为标记任务提供类似于以下内容的工作人员 UI。在控制台中创建标记作业时，需要指定说明，以帮助工作人员完成工作人员可以从中选择的作业和标签。


Instructions
View full instructions
View tool guide
Select a single label that best describes this video clip. Select none of the above if none of the other labels apply.
Select Submit when you are done.
Watch and then class
0:00 / 0:22

创建视频分类标记 Job (API)

本节介绍了使用 SageMaker API 操作创建标记作业时需要了解的详细信息。`CreateLabelingJob`. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅的部分`CreateLabelingJob`.

请按照创建标记作业 (API) (p. 356)中的说明进行操作，并在配置请求时执行以下操作：

- 使用以下结尾的预注 Lambda 函数`PRE-VideoClassification`. 要查找您的区域的注释前 Lambda ARN，请参阅。PreHumanTaskLambdaArn.
- 使用注释合并 Lambda 函数，结尾为`ACS-VideoClassification`. 要查找您所在区域的注释合并 Lambda ARN，请参阅。AnnotationConsolidationLambdaArn.

以下是一个示例：AmazonPython SDK (Boto3) 请求在美国东部（弗吉尼亚北部）区域创建标记作业。

```
response = client.create_labeling_job(
    LabelingJobName='example-video-classification-labeling-job',
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://bucket/path/manifest-with-input-data.json'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/file-to-store-output-data',
        'KmsKeyId': 'string'
    },
    RoleArn='arn:aws:iam::*:role/*',
    LabelCategoryConfigS3Uri='s3://bucket/path/label-categories.json',
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:region:*:workteam/private-crowd/*',
        'UiConfig': {
            'UiTemplateS3Uri': 's3://bucket/path/worker-task-template.html'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda:us-east-1:432418664414:function:PRE-
VideoClassification',
        'TaskKeywords': [
            'Video Classification',
        ],
        'TaskTitle': 'Video classification task',
        'TaskDescription': 'Select a label to classify this video',
        'NumberOfHumanWorkersPerDataObject': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'MaxConcurrentTaskCount': 123,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-VideoClassification'
        },
    Tags=[
        {
```

```
            'Key': 'string',
            'Value': 'string'
        },
    ]
)
```

为视频分类提供模板

如果要使用 API 创建标记作业，必须在 `UiTemplateS3Uri` 中提供工作人员任务模板。通过修改以下模板复制并修改以下模板`short-instructions`、`full-instructions`, 和`header`. 将此模板上传到 Amazon S3，并在中为此文件提供 Amazon S3 URI。`UiTemplateS3Uri`.

视频分类输出数据

创建视频分类标签作业后，您的输出数据 Amazon S3 位于在`S3OutputPath`使用 API 时的参数或在输出数据集位置字段中的Job 概述控制台的部分。

要详细了解 Ground Truth 生成的输出清单文件以及 Ground Truth 用于存储输出数据的文件结构，请参阅。输出数据 (p. 414).

要查看视频分类标记作业的输出清单文件示例，请参阅。分类作业输出 (p. 417).

标记视频帧

您可以使用 Ground Truth 内置视频帧任务类型，让工作人员使用边界框、折线、多边形或关键点对视频帧进行注释。一个视频帧是从视频中提取的一系列图像。

如果没有视频帧，则可以提供视频文件（MP4 文件）并使用 Ground Truth 自动帧提取工具来提取视频帧。要了解更多信息，请参阅 提供视频文件 (p. 410)。

您可以使用以下内置视频任务类型，使用 Amazon SageMaker 控制台、API 和特定于语言的 SDK 创建视频帧标记作业。

- 视频帧对象检测— 如果您希望工作人员按视频帧序列标识和定位对象，请使用该任务类型。您可以提供类别列表，工作人员可以一次选择一个类别并注释该类别应用于所有帧的对象。例如，您可以使用该任务要求工作人员识别和本地化场景中的各种对象，例如汽车、自行车和行人。
- 视频帧对象跟踪— 如果您希望工作人员跟踪对象在视频帧序列中的移动情况，请使用该任务类型。当工作人员向单个帧添加注释时，该注释将与唯一的实例 ID 相关联。工作人员在所有其他帧中添加与相同 ID 关联的注释，以标识同一对象或人员。例如，工作人员可以通过在出现的每帧中绘制与车辆周围相同 ID 关联的边界框来跟踪车辆在一系列视频帧中的移动情况。

请使用以下主题以了解这些内置任务类型以及如何使用每种任务类型创建标记作业。请参阅任务类型 (p. 245)了解有关这些任务类型可用的注释工具（边界框、折线、多边形和关键点）的详细信息。

在创建标记作业之前，建议查看。视频帧标签 Job 概述 (p. 244).

主题

- 视频帧物体检测 (p. 236)
- 视频帧对象跟踪 (p. 240)
- 视频帧标签 Job 概述 (p. 244)

视频帧物体检测

您可以使用视频帧对象检测任务类型让工作人员使用边界框、折线、多边形或关键点识别和定位视频帧序列（从视频中提取的图像）中的对象注释工具. 您选择的工具定义了您创建的视频帧任务类型。例如，您可以使

用边界框视频帧对象检测任务类型工作人员识别和本地化一系列视频帧中的各种对象，例如汽车、自行车和行人。

您可以使用 Amazon SageMaker Ground Truth 控制台、SageMaker API 和特定于语言创建视频帧对象检测标签作业Amazon开发工具包。要了解更多信息，请参阅创建视频帧对象检测标记 Job (p. 237)然后选择您的首选方法。请参阅任务类型 (p. 245)以了解在创建标记作业时可以从中选择的注释工具的更多信息。

Ground Truth 提供了工作人员用户界面和工具来完成标签作业任务：预览工作人员 UI (p. 237).

您可以创建作业以使用视频对象检测调整任务类型以调整在视频对象检测标记作业中创建的注释。要了解更多信息，请参阅 创建视频帧物体检测调整或验证标签 Job (p. 240)。

预览工作人员 UI

Ground Truth 为工作人员提供了 Web 用户界面 (UI) 以完成视频帧对象检测注释任务。在控制台中创建标记作业时，您可以预览该工作人员 UI 并与之交互。如果您是新用户，建议您使用小型输入数据集通过控制台创建标记作业，从而预览工作人员 UI 并确保视频帧、标签和标签属性按预期方式显示。

UI 为工作人员提供了以下辅助标签工具来完成对象检测任务：

- 对于所有任务，工作人员可以使用复制到下一个和复制到所有要素，分别将注释复制到下一帧或所有后续帧。
- 对于包含边界框工具的任务，工作人员可以使用下一步预测在单个帧中绘制边界框的功能，然后让 Ground Truth 预测所有其他帧中具有相同标签的框的位置。然后，工作人员可以进行调整以正确预测的箱子位

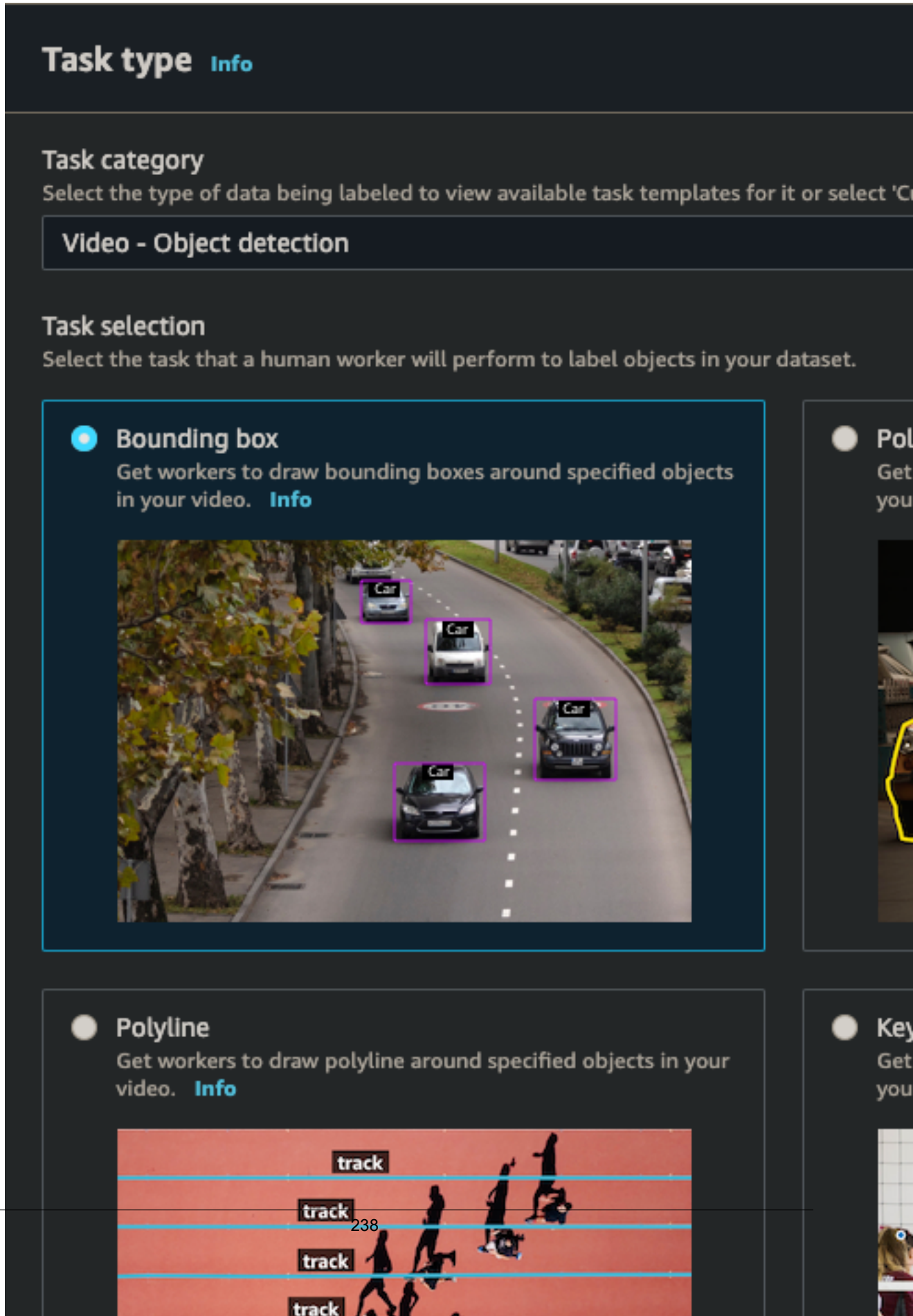
创建视频帧对象检测标记 Job

您可以使用 SageMaker 控制台或`CreateLabelingJob`API 操作。

本部分假定您已查看视频帧标签 Job 概述 (p. 244)并选择了您正在使用的输入数据集的类型和输入数据集连接。

创建标记作业（控制台）

您可以按照中的说明操作：创建标记作业（控制台） (p. 353)以了解如何在 SageMaker 控制台中创建视频帧对象跟踪作业。在步骤 10 中，选择视频-对象检测来自 的任务类别下拉列表。通过在中选择其中的一张卡片来选择你想要的任务类型任务选择.


Task type Info
Task category
Select the type of data being labeled to view available task templates for it or select 'C
Video - Object detection
Task selection
Select the task that a human worker will perform to label objects in your dataset.
Bounding box
Get workers to draw bounding boxes around specified objects in your video. Info
Car
Car
Car
Car
Pol
Get
you
Polyline
Get workers to draw polyline around specified objects in your video. Info
track
track
track
track
Key
Get
you

创建标记作业 (API)

您可以使用 SageMaker API 操作创建对象检测标签作业CreateLabelingJob. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅的部分CreateLabelingJob.

创建标记作业 (API) (p. 356)概述了 CreateLabelingJob 操作。请按照这些说明进行操作，并在配置请求时执行以下操作：

- 您必须为 HumanTaskUiArn 输入一个 ARN。使用 arn:aws:sagemaker:*<region>*:394669845002:human-task-ui/VideoObjectDetection。Replace*<region>*使用Amazon在其中创建标记作业的区域。

  请勿包含UiTemplateS3Uri参数。
- LabelAttributeName 必须以 -ref 结尾。例如：*video-od-labels*-ref。
- 输入清单文件必须是视频帧序列清单文件。您可以使用 SageMaker 控制台创建此清单文件，也可以手动创建该清单文件并将其上传到 Amazon S3。有关更多信息，请参阅 输入数据设置 (p. 410)。
- 您只能使用私人或供应商工作团队来创建视频帧对象检测标签作业。
- 您可以在标签类别配置文件中指定标签、标签类别和帧属性、任务类型和工作人员说明。使用指定任务类型（边界框、折线、多边形或关键点）annotationType在标签类别配置文件中。有关更多信息，请参阅创建具有标签类别和框架属性的标记类别配置文件 (p. 364)以了解如何创建该文件。
- 您需要为注释前和注释后 (ACS) Lambda 函数提供预定义的 ARN。这些 ARN 特定于您用于创建标记作业的 Amazon 区域。
  - 要查找注释前 Lambda ARN，请参阅。PreHumanTaskLambdaArn. 请使用您在其中创建标记作业的区域以查找以结尾的正确 ARNPRE-VideoObjectDetection.
  - 要查找注释后 Lambda ARN，请参阅。AnnotationConsolidationLambdaArn. 请使用您在其中创建标记作业的区域以查找以结尾的正确 ARNACS-VideoObjectDetection.
- NumberOfHumanWorkersPerDataObject 中指定的工作人员数必须为 1。
- 视频帧标记作业不支持自动数据标记。请勿在中指定参数值LabelingJobAlgorithmsConfig.
- 视频帧对象跟踪标记作业可能需要几小时才能完成。您可以在 TaskTimeLimitInSeconds 中为这些标记作业指定更长的时间限制（最多 7 天或 604,800 秒）。

以下是一个示例：AmazonPython 软件开发工具包 (Boto3) 请求以在美国东部（弗吉尼亚北部）区域创建标记作业。

```
response = client.create_labeling_job(
    LabelingJobName='example-video-od-labeling-job',
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://DOC-EXAMPLE-BUCKET/path/video-frame-sequence-input-
manifest.json'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://DOC-EXAMPLE-BUCKET/prefix/file-to-store-output-data',
        'KmsKeyId': 'string'
    },
    RoleArn='arn:aws:iam::*:role/*',
    LabelCategoryConfigS3Uri='s3://bucket/prefix/label-categories.json',
```

```
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:us-east-1:*:workteam/private-crowd/*',
        'UiConfig': {
            'HumanTaskUiArn: 'arn:aws:sagemaker:us-east-1:394669845002:human-task-ui/
VideoObjectDetection'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda:us-east-1:432418664414:function:PRE-
VideoObjectDetection',
        'TaskKeywords': [
            'Video Frame Object Detection',
        ],
        'TaskTitle': 'Video frame object detection task',
        'TaskDescription': 'Classify and identify the location of objects and people in
 video frames',
        'NumberOfHumanWorkersPerDataObject': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'MaxConcurrentTaskCount': 123,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-VideoObjectDetection'
        },
    Tags=[
        {
            'Key': 'string',
            'Value': 'string'
        },
    ]
)
```

创建视频帧物体检测调整或验证标签 Job

您可以使用 Ground Truth 控制台创建调整和验证标记作业，或者`CreateLabelingJob`API。要了解有关调整和验证标签作业的更多信息，以及了解如何创建任务，请参阅验证和调整标签 (p. 318).

输出数据格式

在创建视频帧对象检测标记作业时，任务将发送给工作人员。在这些工作人员完成其任务时，标签将写入到在创建标记作业时指定的 Amazon S3 输出位置。要了解视频帧对象检测输出数据格式，请参阅。视频帧物体检测输出 (p. 425). 如果您是 Ground Tiruth 的新用户，请参阅。输出数据 (p. 414)以了解有关 Ground Truth 输出数据格式的更多信息。

视频帧对象跟踪

您可以使用视频帧对象跟踪任务类型让工作人员使用边界框、折线、多边形或关键点来跟踪视频帧序列（从视频中提取的图像）中的对象移动注释工具. 您选择的工具定义了您创建的视频帧任务类型。例如，您可以使用边界框视频帧对象跟踪任务类型以请求工作人员通过在其周围绘制框以跟踪它们的移动情况，从而跟踪它们的移动情况。

您提供类别列表，工作人员添加到视频帧的每个注释都被标识为实例使用实例 ID 的该类别。例如，如果您提供了标签类别汽车，则工作人员注释的第一辆汽车将具有实例 ID 卡：1。工作人员注释的第二辆汽车将具有实例 ID 卡：2。为了跟踪对象的移动，工作人员将与相同实例 ID 关联的注释添加到所有帧中的对象中。

您可以使用 Amazon SageMaker Ground Truth 控制台、SageMaker API 和特定于语言创建视频帧对象跟踪标签作业Amazon开发工具包。要了解更多信息，请参阅创建视频帧对象检测标记 Job (p. 237)然后选择您的首选方法。请参阅任务类型 (p. 245)以了解在创建标记作业时可以从中选择的注释工具的更多信息。

Ground Truth 提供了工作人员用户界面和工具来完成标签作业任务：预览工作人员 UI (p. 237).

您可以创建作业以使用视频对象检测调整任务类型以调整在视频对象检测标记作业中创建的注释。要了解更多信息，请参阅 创建视频帧物体检测调整或验证标签 Job (p. 240)。

预览工作人员 UI

Ground Truth 为工作人员提供了 Web 用户界面 (UI) 以完成视频帧对象跟踪注释任务。在控制台中创建标记作业时，您可以预览该工作人员 UI 并与之交互。如果您是新用户，建议您使用小型输入数据集通过控制台创建标记作业，从而预览工作人员 UI 并确保视频帧、标签和标签属性按预期方式显示。

UI 为工作人员提供了以下辅助标签工具来完成对象跟踪任务：

- 对于所有任务，工作人员可以使用复制到下一个和复制到所有功能，用于将具有相同唯一 ID 的注释分别复制到下一帧或所有后续帧。
- 对于包含边界框工具的任务，工作人员可以使用下一步预测在单个帧中绘制边界框的功能，然后让 Ground Truth 预测所有其他帧中具有相同唯一 ID 的框的位置。然后，工作人员可以进行调整以正确预测的箱子位

创建视频帧对象跟踪标记 Job

您可以使用 SageMaker 控制台或`CreateLabelingJob`API 操作。

本部分假定您已查看视频帧标签 Job 概述 (p. 244)并选择了您正在使用的输入数据集的类型和输入数据集连接。

创建标记作业（控制台）

您可以按照中的说明操作：创建标记作业（控制台） (p. 353)以了解如何在 SageMaker 控制台中创建视频帧对象跟踪作业。在步骤 10 中，选择视频-对象跟踪来自 的任务类别下拉列表。通过在中选择其中的一张卡片来选择你想要的任务类型任务选择.

Task type Info

Task category

Select the type of data being labeled to view available task templates for it or select 'C

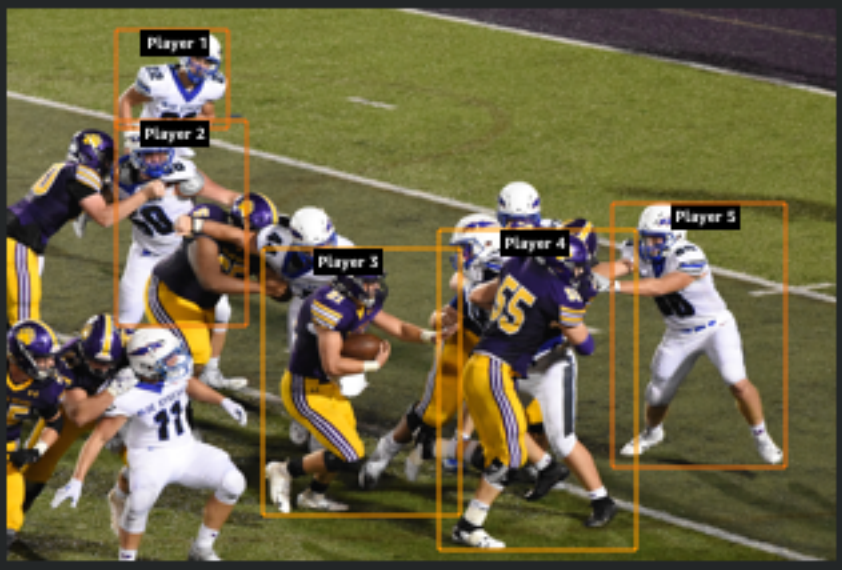
Video - Object tracking

Task selection

Select the task that a human worker will perform to label objects in your dataset.

Bounding box

Get workers to track specific instances of objects in your video across multiple frames in your bounding boxes. Info

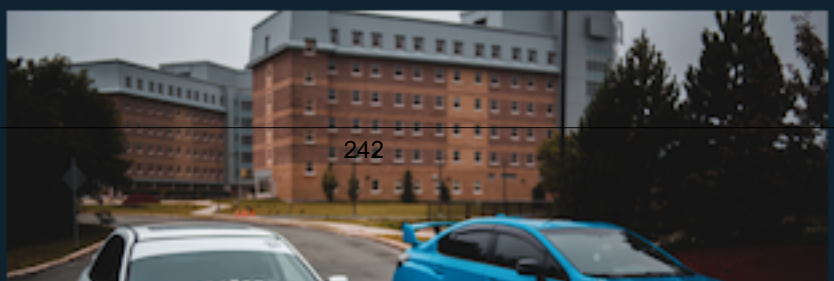


Pol

Get
vide

Polyline

Get workers to track specific instances of objects in your video across multiple frames in your polylines. Info



Key

Get
you

创建标记作业 (API)

您可以使用 SageMaker API 操作创建对象跟踪标签作业CreateLabelingJob. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅的部分CreateLabelingJob.

创建标记作业 (API) (p. 356)概述了 CreateLabelingJob 操作。请按照这些说明进行操作，并在配置请求时执行以下操作：

- 您必须为 HumanTaskUiArn 输入一个 ARN。使用 arn:aws:sagemaker:*<region>*:394669845002:human-task-ui/VideoObjectTracking。Replace*<region>*使用Amazon在其中创建标记作业的区域。

  请勿包含UiTemplateS3Uri参数。
- LabelAttributeName 必须以 -ref 结尾。例如：*ot-labels*-ref。
- 输入清单文件必须是视频帧序列清单文件。您可以使用 SageMaker 控制台创建此清单文件，也可以手动创建该清单文件并将其上传到 Amazon S3。有关更多信息，请参阅 输入数据设置 (p. 410)。如果您创建流式标签作业，则输入清单文件是可选的。
- 您只能使用私人或供应商工作团队来创建视频帧对象检测标签作业。
- 您可以在标签类别配置文件中指定标签、标签类别和帧属性、任务类型和工作人员说明。使用指定任务类型（边界框、折线、多边形或关键点）annotationType在标签类别配置文件中。有关更多信息，请参阅创建具有标签类别和框架属性的标记类别配置文件 (p. 364)以了解如何创建该文件。
- 您需要为注释前和注释后 (ACS) Lambda 函数提供预定义的 ARN。这些 ARN 特定于您用于创建标记作业的 Amazon 区域。
  - 要查找注释前 Lambda ARN，请参阅。PreHumanTaskLambdaArn. 请使用您在其中创建标记作业的区域以查找以结尾的正确 ARNPRE-VideoObjectTracking.
  - 要查找注释后 Lambda ARN，请参阅。AnnotationConsolidationLambdaArn. 请使用您在其中创建标记作业的区域以查找以结尾的正确 ARNACS-VideoObjectTracking.
- NumberOfHumanWorkersPerDataObject 中指定的工作人员数必须为 1。
- 视频帧标记作业不支持自动数据标记。请勿在中指定参数值LabelingJobAlgorithmsConfig.
- 视频帧对象跟踪标记作业可能需要几小时才能完成。您可以在 TaskTimeLimitInSeconds 中为这些标记作业指定更长的时间限制（最多 7 天或 604,800 秒）。

以下是一个示例：AmazonPython 软件开发工具包 (Boto3) 请求以在美国东部（弗吉尼亚北部）区域创建标记作业。

```
response = client.create_labeling_job(
    LabelingJobName='example-video-ot-labeling-job,
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://DOC-EXAMPLE-BUCKET/path/video-frame-sequence-input-
manifest.json'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://DOC-EXAMPLE-BUCKET/prefix/file-to-store-output-data',
        'KmsKeyId': 'string'
    },
    RoleArn='arn:aws:iam::*:role/*,
```

```
    LabelCategoryConfigS3Uri='s3://bucket/prefix/label-categories.json',
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:us-east-1:*:workteam/private-crowd/*',
        'UiConfig': {
            'HumanTaskUiArn: 'arn:aws:sagemaker:us-east-1:394669845002:human-task-ui/
VideoObjectTracking'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda:us-east-1:432418664414:function:PRE-
VideoObjectTracking',
        'TaskKeywords': [
            'Video Frame Object Tracking,
        ],
        'TaskTitle': 'Video frame object tracking task',
        'TaskDescription': Tracking the location of objects and people across video
 frames',
        'NumberOfHumanWorkersPerDataObject': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'MaxConcurrentTaskCount': 123,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-VideoObjectTracking'
        },
    Tags=[
        {
            'Key': 'string',
            'Value': 'string'
        },
    ]
)
```

创建视频帧对象跟踪调整或验证标签 Job

您可以使用 Ground Truth 控制台创建调整和验证标记作业，或`CreateLabelingJob`API。要了解有关调整和验证标签作业的更多信息，以及了解如何创建任务，请参阅验证和调整标签 (p. 318).

输出数据格式

在创建视频帧对象跟踪标记作业时，任务将发送给工作人员。在这些工作人员完成其任务时，标签将写入到在创建标记作业时指定的 Amazon S3 输出位置。要了解视频帧对象跟踪输出数据格式，请参阅。视频帧对象跟踪输出 (p. 427). 如果您是 Ground Tiruth 的新用户，请参阅。输出数据 (p. 414)以了解有关 Ground Truth 输出数据格式的更多信息。

视频帧标签 Job 概述

使用此页可以了解对象检测和对象跟踪视频帧标记作业。此页面上的信息适用于这两种内置任务类型。

由于以下原因，视频帧标签作业是唯一的：

- 您可以提供准备注释的数据对象（视频帧），也可以提供视频文件并让 Ground Truth 自动提取视频帧。
- 员工有能力随时保存工作。
- 您无法使用Amazon Mechanical Turk工作人员来完成标签任务。
- Ground Truth 提供了一个工作人员用户界面以及辅助和基本的标签工具，以帮助员工完成任务。您无需提供工作人员任务模板。

可以使用以下主题了解更多信息。

主题

- 输入数据 (p. 245)
- 作业完成时间 (p. 245)
- 任务类型 (p. 245)
- 人力 (p. 246)
- 工作人员用户界面 (UI) (p. 246)
- 视频帧 Job 权限要求 (p. 247)

输入数据

视频帧标签作业使用序列的视频帧。单个序列是从单个视频中提取的一系列图像。您可以提供自己的视频帧序列，也可以让 Ground Truth 自动从视频文件中提取视频帧序列。要了解更多信息，请参阅 提供视频文件 (p. 410)。

Ground Truth 使用序列文件在一个序列中识别所有图像。要包含在单个标签作业中的所有序列都会在输入清单文件中标识。每个序列都用于创建单个工作人员任务。您可以使用 Ground Truth 自动数据设置自动创建序列文件和输入清单文件。要了解更多信息，请参阅 自动视频帧输入数据设置 (p. 411)。

要了解如何手动创建序列文件和输入清单文件，请参阅创建视频帧输入清单文件 (p. 413).

作业完成时间

完成视频和视频帧标记作业可能需要几小时的时间。您可以在创建标记作业时设置工作人员可处理每个任务的总时间。您可以为工作人员处理任务设置的最长时间为 7 天。默认值为 3 天。

我们强烈建议您创建工作人员可在 12 小时内完成的任务。在处理任务时，工作人员必须将工作人员 UI 保持打开状态。他们可以随时保存工作，Ground Truth 每 15 分钟保存一次他们的工作。

使用 SageMaker 时`CreateLabelingJob`在 API 操作中设置工作人员可处理任务的总时间`TaskTimeLimitInSeconds`的参数`HumanTaskConfig`.

在控制台中创建标记作业时，您可以在选择人力类型和工作组时指定该时间限制。

任务类型

创建视频对象跟踪或视频对象检测标注作业时，您可以指定希望工作人员在处理标注任务时创建的注释类型。注释类型决定了 Ground Truth 返回的输出数据的类型，并定义任务类型为了您的标记作业。

如果您使用 API 操作创建标记作业，请参阅`CreateLabelingJob`，您可以使用标签类别配置文件参数指定任务类型`annotationType`. 要了解更多信息，请参阅 创建具有标签类别和框架属性的标记类别配置文件 (p. 364)。

以下任务类型可用于视频对象跟踪或视频对象检测标签作业：

- 边界框— 为工作人员提供了创建边界框注释的工具。边界框是一个框，工作人员在对象周围绘制，以识别该对象在框架中的像素位置和标签。
- 多边形— 为员工提供了创建折线注释的工具。折线由一系列有序的 x, y 坐标定义。添加到折线的每个点都将一条线连接到前一个点。折线不必关闭（起点和终点不一定相同），并且对线之间形成的角度没有限制。
- 面— 为员工提供了创建多边形注释的工具。多边形是由一系列有序 x, y 坐标定义的封闭形状。添加到多边形的每个点都通过一条线连接到前一个点，并且对线之间形成的角度没有限制。多边形的两条线（边）不能交叉。多边形的起点和终点必须相同。
- 关键点— 为员工提供了创建关键点注释的工具。关键点是与视频帧中的 x, y 坐标关联的单个点。

人力

在创建视频帧标记作业时，您需要指定一个工作组以完成注释任务。您可以从自己的工作人员的私有人力中选择一个工作组，或者从在 Amazon Web Services Marketplace 上选择的供应商人力中选择一个工作组。您不能使用 Amazon Mechanical Turk 工作人员进行视频帧标签工作。

要了解供应商人力的更多信息，请参阅管理供应商人力 (p. 466).

要了解如何创建和管理私有人力，请参阅使用私有人力 (p. 467)。

工作人员用户界面 (UI)

Ground Truth 提供了工作人员用户界面 (UI)、工具和辅助标记功能，以帮助工作人员完成视频标记任务。在控制台中创建标记作业时，您可以预览工作人员 UI。

在使用 API 操作创建标记作业时`CreateLabelingJob`，您必须在参数中提供 Ground Truth 提供的 ARN`HumanTaskUiArn`以便为任务类型指定工作器用户界面。您可以使用`HumanTaskUiArn`与 SageMaker 一起使用`RenderUiTemplate`预览工作人员 UI 的 API 操作。

您可以提供工作人员说明、标签以及属性（可选），以供工作人员提供有关标签和视频帧的更多信息。这些属性分别称为标签类别属性和框架属性。它们都显示在工作人员 UI 中。

标签类别和框架属性

在创建视频对象跟踪或视频对象检测标记作业时，您可以添加一个或多个标签类别属性和帧属性：

- 标签类别属性— 选项（字符串）、自由格式文本框或与一个或多个标签关联的数值字段的列表。工作人员使用它来提供有关标签的元数据。
- “帧” 属性— 发送工作人员进行注释的每个视频帧上显示的选项（字符串）、自由格式文本框或数字字段的列表。工作人员使用它来提供有关视频帧的元数据。

此外，您可以使用标签和帧属性让工作人员在视频帧标签验证作业中验证标签。

可以使用以下部分了解这些属性的更多信息。要了解如何向标注作业添加标签类别和框架属性，请使用创建标记 Job章节内容“任务类型” 页 (p. 236)您选择的。

帧级属性

添加帧属性以使工作人员能够提供有关单个视频帧的更多信息。您添加的每个帧属性都会显示在所有帧上。

例如，您可以添加数字帧属性，让工作人员识别他们在特定帧中看到的对象数量。

在另一个示例中，您可能想要提供一个自由格式文本框，让员工能够对问题提供答案。

创建标签验证作业时，您可以添加一个或多个帧属性，要求工作人员就视频帧中的所有标签提供反馈。

工作人员说明

您可以提供工作人员说明以帮助工作人员完成视频帧标记任务。在编写说明时，您可能想要介绍以下主题：

- 注释对象时的最佳实践和要避免的事项。
- 提供的标签类别属性（对于对象检测和对象跟踪任务）以及如何使用它们的方法。
- 如何使用键盘快捷键在标记标记时节省时间。

在创建标记作业时，您可以使用 SageMaker 控制台添加工作人员说明。如果使用 API 操作 `CreateLabelingJob` 创建标记作业，您可以在标签类别配置文件中指定工作人员说明。

除了您的说明以外，Ground Truth 还提供一个链接以帮助工作人员导航和使用工作人员门户。请在工作人员说明 (p. 248)中选择任务类型以查看这些说明。

拒绝任务

工作人员能够拒绝任务。

如果说明不清楚、输入数据不正确显示或遇到任务的其他问题，工作人员会拒绝任务。如果每个数据集对象的工作人员数量 (`NumberOfHumanWorkersPerDataObject`) 拒绝任务，数据对象被标记为已过期且不会发送给其他工作人员。

视频帧 Job 权限要求

当您创建视频帧标签作业时，除了中的权限要求之外分配 IAM 权限以使用 Ground Truth (p. 447)，您必须将一个 CORS 策略添加到包含输入清单文件的 S3 存储桶中。

将 CORS 权限策略添加到 S3 存储桶

在创建视频帧标记作业时，您可以在 S3 中指定输入数据和清单文件所在的存储桶以及存储输出数据的存储桶。这些存储桶可能是相同的。您必须将以下跨域资源共享 (CORS) 策略附加到输入和输出存储桶中。如果您使用 Amazon S3 控制台将策略添加到存储桶中，您必须使用 JSON 格式。

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "GET",
            "HEAD",
            "PUT"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [
            "Access-Control-Allow-Origin"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<CORSRule>
    <AllowedOrigin>*</AllowedOrigin>
    <AllowedMethod>GET</AllowedMethod>
    <AllowedMethod>HEAD</AllowedMethod>
    <AllowedMethod>PUT</AllowedMethod>
    <MaxAgeSeconds>3000</MaxAgeSeconds>
    <ExposeHeader>Access-Control-Allow-Origin</ExposeHeader>
    <AllowedHeader>*</AllowedHeader>
</CORSRule>
</CORSConfiguration>
```

要了解如何将 CORS 策略添加到 S3 存储桶，请参阅如何通过 CORS 添加跨域资源共享？在《亚马逊简单存储服务用户指南》中。

工作人员说明

本主题概述 Ground Truth 可用于完成视频帧标记任务的工作人员门户和工具。首先，从主题中选择您处理的任务类型。

> Important
>
> 建议您使用谷歌 Chrome 浏览器或 Firefox 网络浏览器完成任务。

对于调整作业，请选择生成要调整的标签的原始标记作业任务类型。根据需要，查看并调整任务中的标签。

主题

- 处理视频帧对象跟踪任务 (p. 248)
- 处理视频帧对象检测任务 (p. 254)

处理视频帧对象跟踪任务

视频帧对象跟踪任务要求您跟踪对象在视频帧之间的移动。视频帧是来自视频场景的静态图像。

您可以使用工作器 UI 在视频帧之间导航，并使用提供的工具来识别唯一对象并跟踪它们从一个到另一个对象的移动。使用此页可以了解如何导航您的工作人员 UI、使用提供的工具以及完成任务。

建议您使用谷歌 Chrome 浏览器或 Firefox 网络浏览器完成任务。

> Important
>
> 如果在打开任务时看到注释已添加到一个或多个视频帧中，请调整这些注释并根据需要添加其他注释。

主题

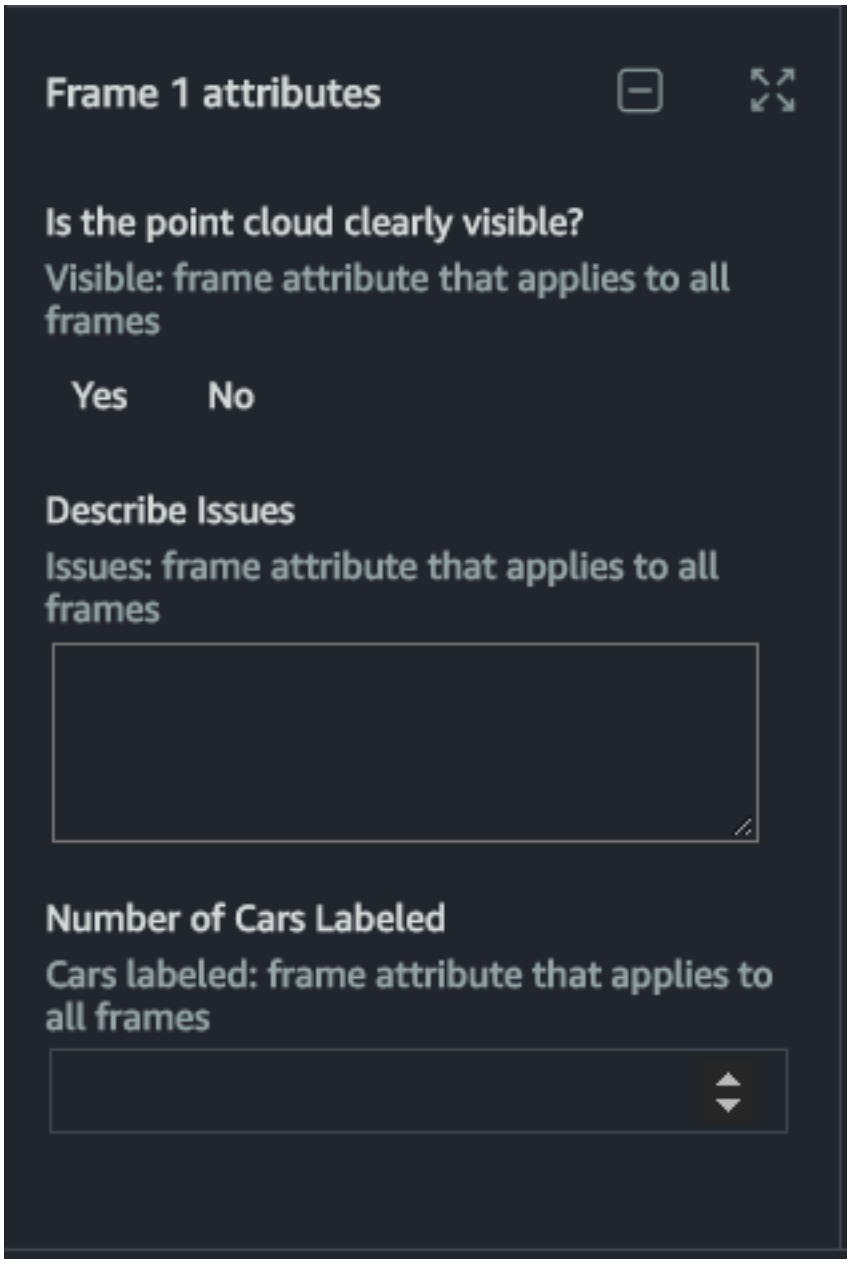
- 您的任务 (p. 248)
- 浏览 UI (p. 250)
- 批量编辑标签和框架属性 (p. 250)
- 工具指南 (p. 250)
- 图标指南 (p. 252)
- Shortcuts (p. 254)
- 释放、停止和恢复以及拒绝任务 (p. 254)
- 保存您的工作并提交 (p. 254)

您的任务

在处理视频帧对象跟踪任务时，您需要从标签类别开始注释工作人员门户右侧的菜单。在选择类别后，请使用提供的工具对该类别适用的对象进行注释。此注释将与只能用于该对象的唯一标签 ID 相关联。使用此相同的标签 ID 可以在其中显示的所有视频帧中为同一对象创建其他注释。请参阅工具指南 (p. 250)以了解有关提供的工具的更多信息。

添加标签后，您可能会看到标签旁边的向下箭头标签。选择此箭头，然后为您看到的每个标签属性选择一个选项，以提供有关该标签的详细信息。

你可能会在标签。这些属性将出现在任务的每个帧中。使用这些属性提示输入有关每个帧的其他信息。



添加标签后，可以使用标签旁边的向下箭头快速添加和编辑标签类别属性值标签。如果您在中选择标签旁边的铅笔图标。标签中的菜单编辑实例菜单将出现。您可以使用此菜单编辑标签 ID、标签类别和标签类别属性。

要编辑注释，请在中选择要编辑的注释的标签。标签菜单或在框架中选择注释。当您编辑或删除注释时，该操作只会在单个框架中修改注释。

如果您正在处理包含边界框工具的任务，请使用预测下一个图标来预测下一帧中在框架中绘制的所有边界框的位置。如果选择单个框然后选择预测下一个图标，则在下一帧中只会预测该框。如果您没有向当前帧添加任何框，则将收到错误。在使用该功能之前，您必须在帧中添加至少一个框。

使用预测下一个图标后，请查看下一帧中每个箱子的位置，并根据需要调整箱子的位置和大小。

对于所有其他工具，您可以使用复制到下一个和复制到所有分别将注释复制到下一个或所有帧的工具。

浏览 UI

您可以使用 UI 左下角的导航栏在视频帧之间导航。

使用播放按钮自动在整个帧序列中移动。

使用下一帧和上一帧按钮一次向前或向后移动一帧。您还可以输入帧编号以导航到该帧。

您可以放大和缩小所有视频帧。放大视频帧后，可以使用移动图标在该帧中四处移动。在单个视频帧中通过缩放和移动在单个视频帧中设置新视图时，所有视频帧都设置为同一视图。您可以使用适合屏幕图标将所有视频帧重置为原始视图。有关其他视图选项，请参阅图标指南 (p. 252).

当您位于工作人员 UI 时，您会看到以下菜单：

- 说明— 开始执行任务之前，请查看这些说明。此外，选择更多说明然后查看这些说明。
- Shortcuts— 使用该菜单可以使用该菜单查看键盘快捷键，它们可用于导航视频帧和使用提供的工具。
- 帮助— 使用此选项返回此文档。

批量编辑标签和框架属性

您可以批量编辑标签属性和框架属性（属性）。

批量编辑属性时，您可以指定要将编辑应用到的一个或多个帧范围。您选择的属性将在该范围内的所有帧中进行编辑，包括指定的起始帧和结束帧。批量编辑标签属性时，您指定的范围必须包含标签属性附加到的标签。如果您指定的帧不包含该标签，则将收到错误。

要批量编辑属性，你必须首先指定属性的所需值。例如，如果您希望从更改属性是到否，你必须选择否，然后执行批量编辑。

还可以为尚未填充的属性指定新值，然后使用批量编辑功能在多个帧中填充该值。为此，请为属性选择所需的值并完成以下过程。

要批量编辑标签或属性：

1. 使用鼠标右键单击要批量编辑的属性。
2. 指定要使用短划线应用批量编辑的帧范围 (-) 在文本框中显示。例如，如果要将编辑应用于第 1 至 10 帧，请输入`1-10`. 如果你想将编辑应用于第二到五帧，八到十和二十输入`2-5,8-10,20`.
3. 选择 Confirm（确认）。

如果收到错误消息，请验证您输入的范围是否有效，并且指定的所有帧中都存在与正在编辑的 label 属性关联的标签（如果适用）。

您可以使用复制到以前的帧和复制到下一帧中的选项标签屏幕顶部的菜单。

工具指南

您的任务将包括一个或多个工具。提供的工具指定了您将创建用于识别和跟踪对象的注释类型。请使用下表以了解有关提供的每个工具的更多信息。

工具	图标	操作	描述
边界框		添加边界框注释。	选择该图标以添加边界框。您添加的每个边框与您从 Label (标签) 类别下拉菜单中选择的类别相关联。选择边界框或其关联的标签进行调整。

工具	图标	操作	描述
边界框		在下一帧中预测边界框。	选择一个边框，然后选择此图标以预测下一帧中该框的位置。您可以连续多次选择图标以自动检测多个帧中框的位置。例如，选择此图标 5 次以预测接下来 5 帧中边界框的位置。
keypoint		添加关键点注释。	选择该图标以添加关键点。单击图像的对象可将关键点放置在该位置。 您添加的每个关键点与您从 Label (标签) 类别下拉菜单中选择的类别相关联。选择关键点或其关联的标签进行调整。
多边形		添加折线注释。	选择该图标以添加折线。要添加折线，请连续单击感兴趣的对象以添加新点。要停止绘制折线，请选择第二次放置的最后一个点（此点将为绿色），或者按Enter在键盘上。 添加到折线的每个点都将一条线连接到前一个点。折线不必关闭（起点和终点不一定相同），并且对线之间形成的角度没有限制。 您添加的每个折线与您从 Label 类别下拉菜单中选择的类别相关联。选择折线或其关联的标签进行调整。

工具	图标	操作	描述
面		添加多边形注释。	选择该图标以添加多边形。要添加多边形，请连续单击感兴趣的对象以添加新点。要停止绘制多边形，请选择起点（此点将为绿色）。 多边形是由您放置的一系列点定义的封闭形状。添加到多边形的每个点都通过一条线连接到前一个点，并且对线之间形成的角度没有限制。起点和终点必须相同。 您添加的每个多边形与您从 Label (标签) 类别下拉菜单中选择的类别相关联。选择多边形或其关联的标签进行调整。
复制到下一个		将注释复制到下一帧。	如果在当前框架中选择了一个或多个批注，则这些批注将复制到下一帧中。如果未选择批注，则当前框架中的所有批注都将复制到下一帧中。
复制到所有		将注释复制到所有后续帧中。	如果在当前框架中选择了一个或多个批注，则这些批注将复制到所有后续帧中。如果未选择注释，则当前框架中的所有注释都将复制到后续所有帧中。

图标指南

可以使用该表以了解您在 UI 中看到的图标。您可以使用中的键盘快捷键自动选择其中的一些图标Shortcuts。

图标	操作	描述
	亮度	选择此图标可调整所有视频帧的亮度。

图标	操作	描述
	对比	选择此图标可调整所有视频帧的对比度。
	放大	选择此图标可放大所有视频帧。
	缩小	选择此图标可缩小所有视频帧。
	移动屏幕	放大视频帧后，选择此图标可在该视频帧中四处移动。您可以使用鼠标在视频帧四处移动，方法是单击并向想要移动的方向拖动该帧。这将改变所有视图框中的视图。
	适合屏幕	将所有视频帧重置为原始位置。
	撤消	撤消操作。您可以使用此图标移除刚添加的边界框，或撤消对边界框所做的调整。
	重做	重做使用撤消图标撤消的操作。
	删除标签	删除标签。这将在单个框架中删除与标签关联的边界框。
	显示或隐藏标签	选择此图标可显示已隐藏的标签。如果此图标有斜线，请选择它以隐藏标签。
	编辑标签	选择此图标可打开编辑实例。使用此菜单可以编辑标签类别、ID 以及添加或编辑标签属性。

Shortcuts

中列出的键盘快捷键Shortcuts菜单可以帮助您快速选择图标、撤消和重做批注，并使用工具添加和编辑注释。例如，一旦你添加了边界框，你就可以使用P以便在后续帧中快速预测该盒子的位置。

在开始执行任务之前，建议您查看Shortcuts菜单并熟悉这些命令。

释放、停止和恢复以及拒绝任务

当你打开标签任务时，右上角的三个按钮允许你拒绝任务（拒绝任务），释放它（发布任务），稍后再停止和恢复它（稍后停止和恢复)。以下列表描述了选择这些选项之一时会发生什么情况：

- 拒绝任务：只有在任务有问题（例如视频帧图像不清楚或 UI 出现问题）时，才应拒绝任务。如果您拒绝任务，您将无法返回该任务。
- 发布任务：使用此选项可以释放任务并允许其他人处理任务。当你发布任务时，你会放松该任务完成的所有工作，你的团队中的其他工作人员可以接受它。如果有足够的工作人员接受该任务，您可能无法返回任务。当你选择此按钮然后选择确认，您将返回到工作人员门户网站。如果任务仍可用，则其状态将为Available. 如果其他员工拿起它，它将从您的门户中消失。
- 稍后停止和恢复：您可以使用稍后停止和恢复按钮停止工作并稍后返回任务。您应使用Save（保存）选择按钮保存您的工作稍后停止和恢复. 当你选择此按钮然后选择确认，您将返回到工作人员门户，任务状态为Stopped. 你可以选择相同的任务来恢复对它的工作。

  请注意，创建标签任务的人员指定了一个时间限制，在此限制内所有任务都需要完成。如果您没有在该时限内返回并完成此任务，则该任务将过期并且不会提交您的工作。有关详情，请联系您的管理员。

保存您的工作并提交

你应该定期使用Save（保存）按钮。每 15 分钟自动保存您的工作。

在打开任务时，您必须先在其中完成您的工作，然后再按 Submit (提交)。

处理视频帧对象检测任务

视频帧对象检测任务要求您使用注释对象在视频帧中的位置进行分类和识别。视频帧是来自视频场景的静态图像。

您可以使用工作器 UI 在视频帧之间导航，然后创建注释以识别感兴趣的对象。使用该页面上的部分以了解如何导航工作人员用户界面、使用提供的工具和完成任务。

建议您使用谷歌 Chrome 浏览器完成任务。

> **Important**
>
> 如果在打开任务时看到注释已添加到一个或多个视频帧中，请调整这些注释并根据需要添加其他注释。

主题

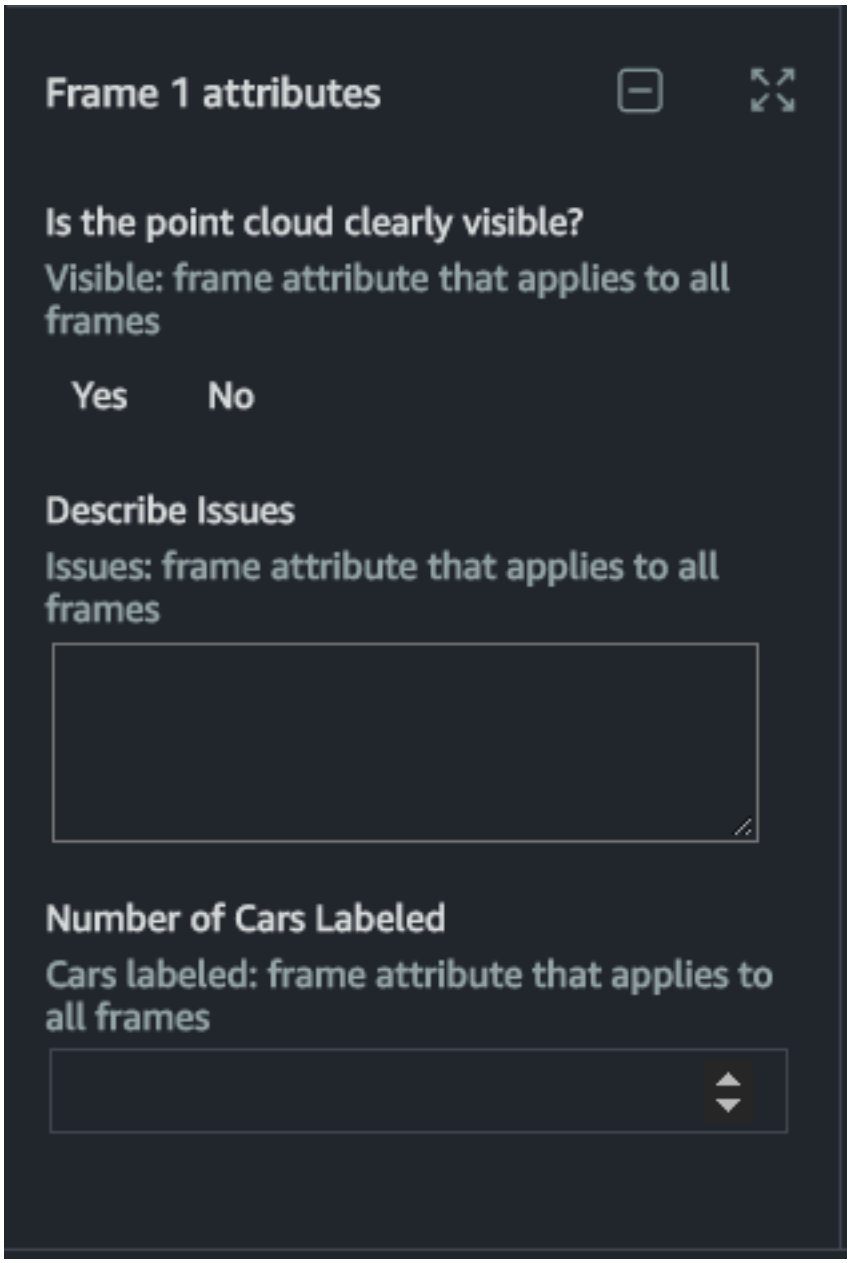
- 您的任务 (p. 255)
- 浏览 UI (p. 256)
- 批量编辑标签和框架属性 (p. 256)
- 工具指南 (p. 256)
- UI 图标指南 (p. 258)
- Shortcuts (p. 259)
- 释放、停止和恢复以及拒绝任务 (p. 259)
- 保存您的工作并提交 (p. 260)

您的任务

在处理视频帧对象检测任务时，您需要从标签类别开始注释工作人员门户右侧的菜单。选择类别后，围绕此类别适用的对象绘制注释。要详细了解你在工作人员 UI 中看到的工具，请参阅工具指南 (p. 256).

添加标签后，您可能会看到标签旁边的向下箭头标签。选择此箭头，然后为您看到的每个标签属性选择一个选项，以提供有关该标签的详细信息。

你可能会在标签。这些属性将出现在任务的每个帧中。使用这些属性提示输入有关每个帧的其他信息。



要编辑注释，请在中选择要编辑的注释的标签。标签菜单或在框架中选择注释。当您编辑或删除注释时，该操作只会在单个框架中修改注释。

如果您正在处理包含边界框工具的任务，请使用预测下一个图标来预测下一帧中在框架中绘制的所有边界框的位置。如果选择单个框然后选择预测下一个图标，则在下一帧中只会预测该框。如果您没有向当前帧添加任何框，则将收到错误。在使用该功能之前，您必须在帧中添加至少一个框。

Note

预测下一个要素不会覆盖手动创建的注释。它只会添加注释。如果接下来使用预测，并因此在单个对象周围有多个边界框，请删除除一个框以外的所有边界框。每个对象只能用一个框来识别。

使用预测下一个图标后，请查看下一帧中每个箱子的位置，并根据需要调整箱子的位置和大小。

对于所有其他工具，您可以使用复制到下一个和复制到所有分别将注释复制到下一个或所有帧的工具。

浏览 UI

您可以使用 UI 左下角的导航栏在视频帧之间导航。

使用播放按钮自动播放多个帧。

使用下一帧和上一帧按钮一次向前或向后移动一帧。您还可以输入帧编号以导航到该帧。

您可以放大和缩小所有视频帧。放大视频帧后，可以使用移动图标在该帧中四处移动。当您通过缩放和移动在单个视频帧中导航到该视图中的新视图时，所有视频帧都设置为同一视图。您可以使用适合屏幕图标将所有视频帧重置为原始视图。要了解更多信息，请参阅 UI 图标指南 (p. 258)。

当您位于工作人员 UI 时，您会看到以下菜单：

- 说明— 开始执行任务之前，请查看这些说明。此外，选择更多说明然后查看这些说明。
- Shortcuts— 使用该菜单可以使用该菜单查看键盘快捷键，它们可用于导航视频帧和使用提供的注释工具。
- 帮助— 使用此选项返回此文档。

如果您

批量编辑标签和框架属性

您可以批量编辑标签属性和框架属性（属性）。

批量编辑属性时，您可以指定要将编辑应用到的一个或多个帧范围。您选择的属性将在该范围内的所有帧中进行编辑，包括指定的起始帧和结束帧。批量编辑标签属性时，您指定的范围必须包含标签属性附加到的标签。如果您指定的帧不包含该标签，则将收到错误。

要批量编辑属性，你必须首先指定属性的所需值。例如，如果您希望从更改属性是到否，你必须选择否，然后执行批量编辑。

还可以为尚未填充的属性指定新值，然后使用批量编辑功能在多个帧中填充该值。为此，请为属性选择所需的值并完成以下过程。

要批量编辑标签或属性：

1. 使用鼠标右键单击要批量编辑的属性。
2. 指定要使用短划线应用批量编辑的帧范围 (-) 在文本框中显示。例如，如果要将编辑应用于第 1 至 10 帧，请输入`1-10`. 如果你想将编辑应用于第二到五帧，八到十和二十输入`2-5,8-10,20`.
3. 选择 Confirm（确认）。

如果收到错误消息，请验证您输入的范围是否有效，并且指定的所有帧中都存在与正在编辑的 label 属性关联的标签（如果适用）。

您可以使用复制到以前的帧和复制到下一帧中的选项标签屏幕顶部的菜单。

工具指南

您的任务将包括一个或多个工具。提供的工具指定了您将创建用于识别和标记对象的注释类型。使用下表详细了解您可能在工作人员 UI 中看到的一个或多个工具。

工具	图标	操作	描述
边界框		添加边界框注释。	选择该图标以添加边界框。您添加的每个边框与您从 Label (标签) 类别下拉菜单中选择的类别相关联。选择边界框或其关联的标签进行调整。
预测下一步		在下一帧中预测边界框。	选择一个边框，然后选择此图标以预测下一帧中该框的位置。您可以连续多次选择图标以自动检测多个帧中框的位置。例如，选择此图标 5 次以预测接下来 5 帧中边界框的位置。
keypoint		添加关键点注释。	选择该图标以添加关键点。单击图像的对象可将关键点放置在该位置。 您添加的每个关键点与您从 Label (标签) 类别下拉菜单中选择的类别相关联。选择关键点或其关联的标签进行调整。
多边形		添加折线注释。	选择该图标以添加折线。要添加折线，请连续单击感兴趣的对象以添加新点。要停止绘制折线，请选择第二次放置的最后一个点（此点将为绿色），或者按Enter在键盘上。 添加到折线的每个点都将一条线连接到前一个点。折线不必关闭（起点和终点不一定相同），并且对线之间形成的角度没有限制。 您添加的每个折线与您从 Label 类别下拉菜单中选择的类别相关联。选择折线或其关联的标签进行调整。
面		添加多边形注释。	选择该图标以添加多边形。要添加多边形，请连续单击感兴趣的对象以添加新点。要停止绘制多边形，请选择起点（此点将为绿色）。

工具	图标	操作	描述
			多边形是由您放置的一系列点定义的封闭形状。添加到多边形的每个点都通过一条线连接到前一个点，并且对线之间形成的角度没有限制。多边形的两条线（边）不能交叉。如果一条线违反此条件，将变为红色。起点和终点必须相同。 您添加的每个多边形与您从 Label (标签) 类别下拉菜单中选择的类别相关联。选择多边形或其关联的标签进行调整。
复制到下一个		将注释复制到下一帧。	如果在当前框架中选择了一个或多个批注，则这些批注将复制到下一帧中。如果未选择批注，则当前框架中的所有批注都将复制到下一帧中。
复制到所有		将注释复制到所有后续帧中。	如果在当前框架中选择了一个或多个批注，则这些批注将复制到所有后续帧中。如果未选择注释，则当前框架中的所有注释都将复制到后续所有帧中。

UI 图标指南

可以使用该表以了解您在工作人员任务门户中看到的图标。您可以使用中的键盘快捷键自动选择这些图标Shortcuts。

图标		描述
	亮度	选择此图标可调整所有视频帧的亮度。
	对比	选择此图标可调整所有视频帧的对比度。

图标		描述
	放大	选择此图标可放大所有视频帧。
	缩小	选择此图标可缩小所有视频帧。
	移动屏幕	放大视频帧后，选择此图标可在该视频帧中四处移动。您可以使用鼠标在视频帧中四处移动，方法是按照想要移动的方向单击并拖动该帧。这将改变所有视图框中的视图。
	适合屏幕	将所有视频帧重置为原始位置。
	撤消	撤消操作。您可以使用此图标移除刚添加的边界框，或撤消对边界框所做的调整。
	重做	重做使用撤消图标撤消的操作。
	删除标签	删除标签。这将在单个框架中删除与标签关联的边界框。
	显示或隐藏标签	选择此图标可显示已隐藏的标签。如果此图标有斜线，请选择它以隐藏标签。

Shortcuts

中列出的键盘快捷键Shortcuts菜单可以帮助您快速选择图标、撤消和重做批注，并使用工具添加和编辑注释。例如，一旦你添加了边界框，你就可以使用P以便在后续帧中快速预测该盒子的位置。

在开始执行任务之前，建议您查看Shortcuts菜单并熟悉这些命令。

释放、停止和恢复以及拒绝任务

当你打开标签任务时，右上角的三个按钮允许你拒绝任务（拒绝任务），释放它（发布任务），稍后再停止和恢复它（稍后停止和恢复)。以下列表描述了选择这些选项之一时会发生什么情况：

- 拒绝任务：只有在任务有问题（例如视频帧图像不清楚或 UI 出现问题）时，才应拒绝任务。如果您拒绝任务，您将无法返回该任务。
- 发布任务：使用此选项可以释放任务并允许其他人处理任务。当你发布任务时，你会放松该任务完成的所有工作，你的团队中的其他工作人员可以接受它。如果有足够的工作人员接受该任务，您可能无法返回任务。当你选择此按钮然后选择确认，您将返回到工作人员门户网站。如果任务仍可用，则其状态将为Available. 如果其他员工拿起它，它将从您的门户中消失。
- 稍后停止和恢复：您可以使用稍后停止和恢复按钮停止工作并稍后返回任务。您应使用Save（保存）选择按钮保存您的工作稍后停止和恢复. 当你选择此按钮然后选择确认，您将返回到工作人员门户，任务状态为Stopped. 你可以选择相同的任务来恢复对它的工作。

  请注意，创建标签任务的人员指定了一个时间限制，在此限制内所有任务都需要完成。如果您没有在该时限内返回并完成此任务，则该任务将过期并且不会提交您的工作。有关详情，请联系您的管理员。

保存您的工作并提交

您应该定期保存您的工作。Ground Truth 每 15 分钟自动保存一次您的工作。

在打开任务时，您必须先在其中完成您的工作，然后再按提交.

使用 Ground Truth 标记 3D 点云

创建 3D 点云标记作业，以让工作人员在 3D 点云中标记对象，这些点云是从 3D 传感器（例如 光检测和测距 (LiDAR) 传感器）和深度摄像机中生成的，或者是从 3D 重建中生成的（通过拼接无人机等代理捕获的图像）。

3D 点云

点云由包含点的三维 (3D) 可视数据组成。每个点是使用三个坐标（通常为 `x`、`y` 和 `z`）描述的。要在点云中添加颜色或点强度变化，可以使用其他属性描述点，例如使用 `i` 表示强度或使用红色 (`r`)、绿色 (`g`) 和蓝色 (`b`) 8 位颜色通道值。在创建 Ground Truth 3D 点云标记作业时，您可以提供点云以及传感器融合数据（可选）。

下图显示了由 Ground Truth 渲染并在语义分割工作人员 UI 中显示的单个 3D 点云场景。

Instructions
Shortcuts
View
3D Point Cloud
Paint
Help
Point cloud View

LiDAR

光检测和测距 (LiDAR) 传感器是一种常见类型的传感器，用于收集在生成点云数据时使用的测量值。LiDAR 是一种遥感方法，它使用脉冲激光形式的光以测量物体与传感器之间的距离。您可以使用中所述的原始数据格式，为 Ground Truth 3D 点云标记作业提供从 LiDAR 传感器中生成的 3D 点云数据。接受的原始 3D 数据格式 (p. 388).

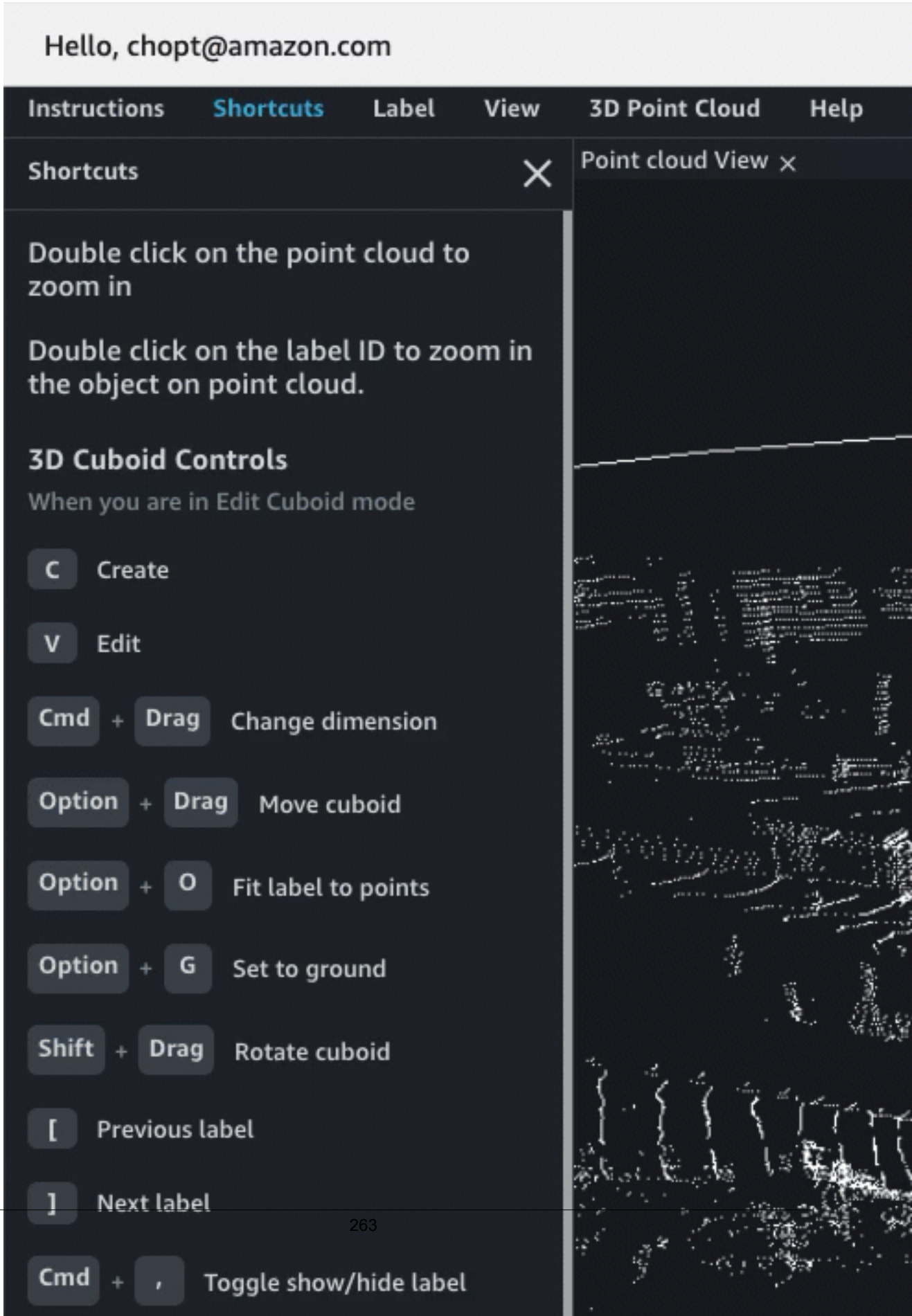
传感器融合

Ground Truth 点云标记作业包括传感器融合功能，该功能为所有任务类型提供摄像机传感器融合支持。一些传感器配备了多个 LiDAR 设备和视频摄像机，以捕获图像并将其与 LiDAR 帧相关联。为了帮助注释者高度自信地直观完成任务，您可以使用 Ground Truth 传感器融合功能通过 3D 扫描仪（例如 LiDAR）外部矩阵以及摄像机外部和内部矩阵将注释（标签）从 3D 点云投影到 2D 摄像机图像，反之亦然。要了解更多信息，请参阅 传感器融合 (p. 402)。

标记 3D 点云

Ground Truth 提供了工作人员用于标记或注释3D 点云。在使用对象检测或语义分割任务类型时，工作人员可以注释单个点云帧。在使用对象跟踪时，工作人员注释一个帧序列。您可以使用对象跟踪以跟踪对象在序列中的所有帧上的移动情况。

以下内容说明了工作人员如何使用 Ground Truth 工作人员门户和工具，为对象检测任务注释 3D 点云。有关其他任务类型的类似可视示例，请参阅 3D 点云任务类型 (p. 264)。


Hello, chopt@amazon.com
Instructions
Shortcuts
Label
View
3D Point Cloud
Help
Shortcuts
Point cloud View
Double click on the point cloud to zoom in
Double click on the label ID to zoom in the object on point cloud.
3D Cuboid Controls
When you are in Edit Cuboid mode
C Create
V Edit
Cmd + Drag Change dimension
Option + Drag Move cuboid
Option + O Fit label to points
Option + G Set to ground
Shift + Drag Rotate cuboid
[Previous label
] Next label
Cmd + , Toggle show/hide label

用于点云注释的辅助标记工具

Ground Truth 提供了辅助标记工具，以帮助工作人员快速准确地完成点云注释任务。有关在工作人员 UI 中包含的每种任务类型的辅助标记工具的详细信息，请选择一种任务类型 (p. 264)，然后参阅该页面的查看工作人员任务界面部分。

后续步骤

在使用 Ground Truth 点云标记作业时，您可以创建 6 种类型的任务。可以使用3D 点云任务类型 (p. 264)中的主题以了解这些任务类型的更多信息，并了解如何使用所选的任务类型创建标记作业。

3D 点云标记作业不同于其他 Ground Truth 标记模式。在创建标记作业之前，我们建议您阅读3D 点云标记作业概述 (p. 286)。此外，请在中查看输入数据配额3D 点云和视频帧标记 Job 配额 (p. 386).

有关使用 SageMaker API 的端到端演示Amazon有关创建 3D 点云标记作业的 Python SDK（boo 3），请参阅。创建 3D 点云标签-Job.iPynb中的SageMaker 示例笔记本标签.

> Important
>
> 如果您使用在 2020 年 6 月 5 日之前创建的笔记本实例运行该笔记本，您必须停止并重新启动该笔记本实例，笔记本才能正常工作。

主题

- 3D 点云任务类型 (p. 264)
- 3D 点云标记作业概述 (p. 286)
- 工作人员说明 (p. 290)

3D 点云任务类型

您可以将 Ground Truth 3D 点云标记模式用于多种使用案例。以下列表简要介绍了每种 3D 点云任务类型。有关如何使用特定任务类型创建标记作业的其他详细信息和说明，请选择任务类型名称以查看其任务类型页面。

- 3D 点云物体检测— 当您希望工作人员在对象周围添加 3D 长方体并使其适合这些对象，从而在 3D 点云中查找对象和进行分类，请使用该任务类型。
- 3D 点云对象跟踪— 当您希望工作人员在对象周围添加 3D 长方体并使其适合这些对象，从而跟踪它们在 3D 点云帧序列中的移动情况，请使用该任务类型。例如，您可以使用该任务类型要求工作人员跟踪车辆在多个点云帧上的移动情况。
- 3D 点云语义分割— 当您希望工作人员使用不同的颜色（每种颜色分配给您指定的类之一）在 3D 点云中绘制对象以创建点级语义分割掩膜，请使用该任务类型。
- 3D 点云调整任务类型 — 以上每种任务类型都有关联调整您可以使用该任务类型审计和调整从 3D 点云标记作业中生成的注释。请参阅关联的类型的任务类型页面，以了解如何为该任务创建调整标记作业。

3D 点云对象检测

当您希望工作人员在对象周围绘制 3D 长方体以对 3D 点云中的对象进行分类时，请使用该任务类型。例如，您可以使用该任务类型要求工作人员识别点云中的不同类型的对象，例如汽车、自行车和行人。

对于此任务类型，数据对象该工作人员标签是单个点云帧。GGround Truth 使用您提供的点云数据渲染 3D 点云。您也可以提供摄像机数据，以便为工作人员提供有关帧中的场景的更多视觉信息，并帮助工作人员在对象周围绘制 3D 长方体。

Gground Truth (地面) 为工作人员提供了一些工具，以便在 3D 场景和投影的侧视图（俯视图、侧视图和后视图）上的三个维度中以 9 个移动度 (x, y, z, rx, ry, rz, rz, rz, rz, rz, rz, rz, rz, rz, rz, rz, 如果您提供传感器融合

信息（例如摄像机数据），在工作人员添加长方体以标识 3D 点云中的对象时，将在 2D 图像中显示该长方体并且可以进行修改。在添加长方体后，在 2D 或 3D 场景中对该长方体所做的所有编辑将投影到另一个视图中。

您可以创建一个作业，以使用 3D 点云对象检测调整任务类型调整在 3D 点云对象检测标记作业中创建的注释。

如果您是 GGround Truth 3D 点云标记模式的新用户，我们建议您查阅。3D 点云标记作业概述 (p. 286). 该标记模式与其他 Ground Truth 际任务类型不同，该页面概述了在创建 3D 点云标记作业时应注意的重要详细信息。

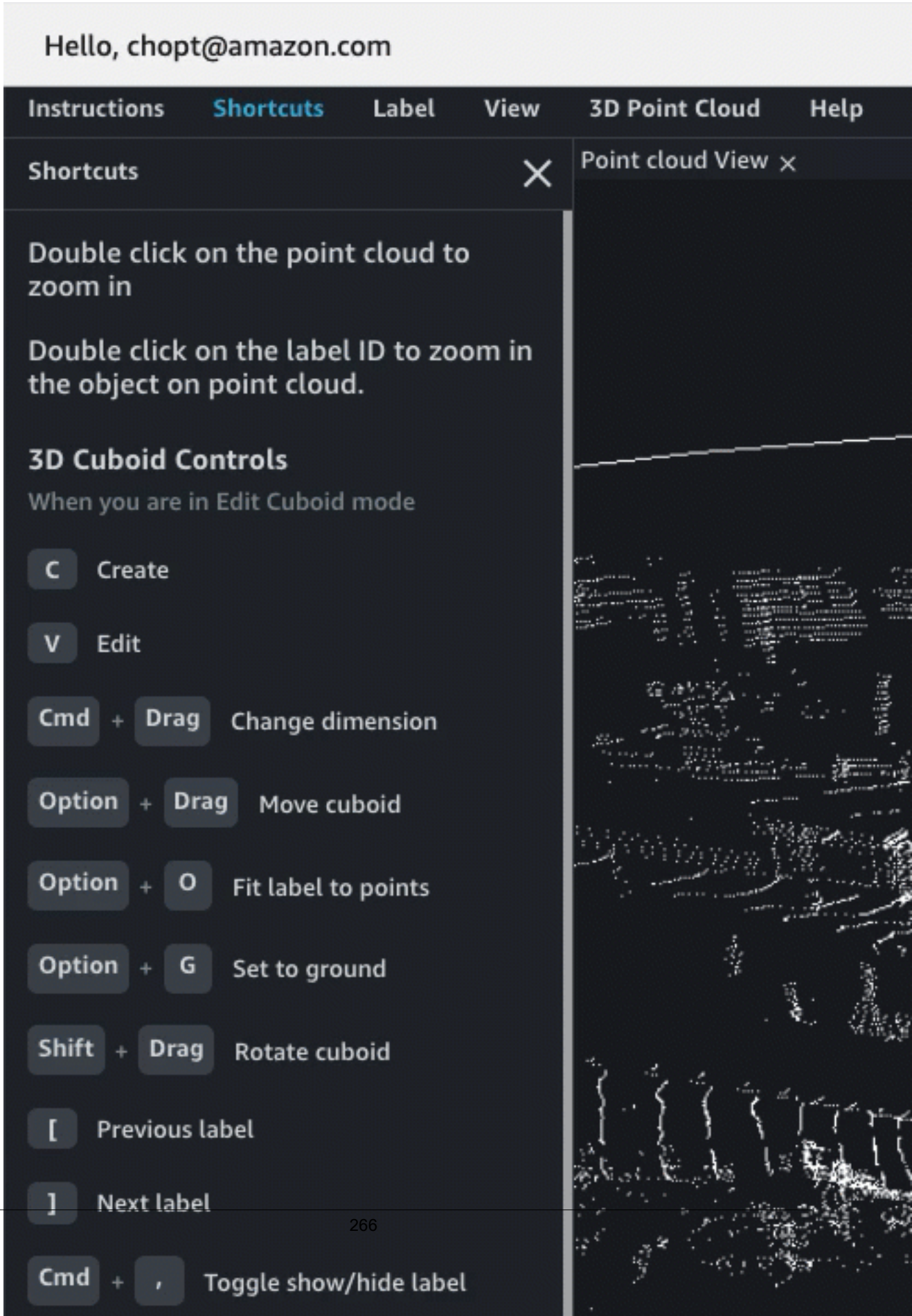
主题

- 查看工作人员任务界面 (p. 265)
- 创建 3D 点云对象检测标记作业 (p. 269)
- 创建 3D 点云对象检测调整或验证标记 Job (p. 270)
- 输出数据格式 (p. 270)

查看工作人员任务界面

Gground Truth 为工作人员提供了 Web 门户和工具以完成 3D 点云对象检测注释任务。在创建标记作业 ARN，您 Ground Truth 以在`HumanTaskUiArn`参数。在控制台中使用该任务类型创建标记作业时，将自动使用该工作人员 UI。在控制台中创建标记作业时，您可以预览该工作人员 UI 并与之交互。如果您是新用户，建议您使用控制台创建标记作业，以确保标签属性、点云帧和图像（如果适用）按预期方式显示。

以下是 3D 点云对象检测工作人员任务界面的 GIF。如果您在世界坐标系中提供用于传感器融合的摄像机数据，图像将与点云帧中的场景进行匹配。这些图像显示在工作人员门户中，如以下 GIF 中所示。

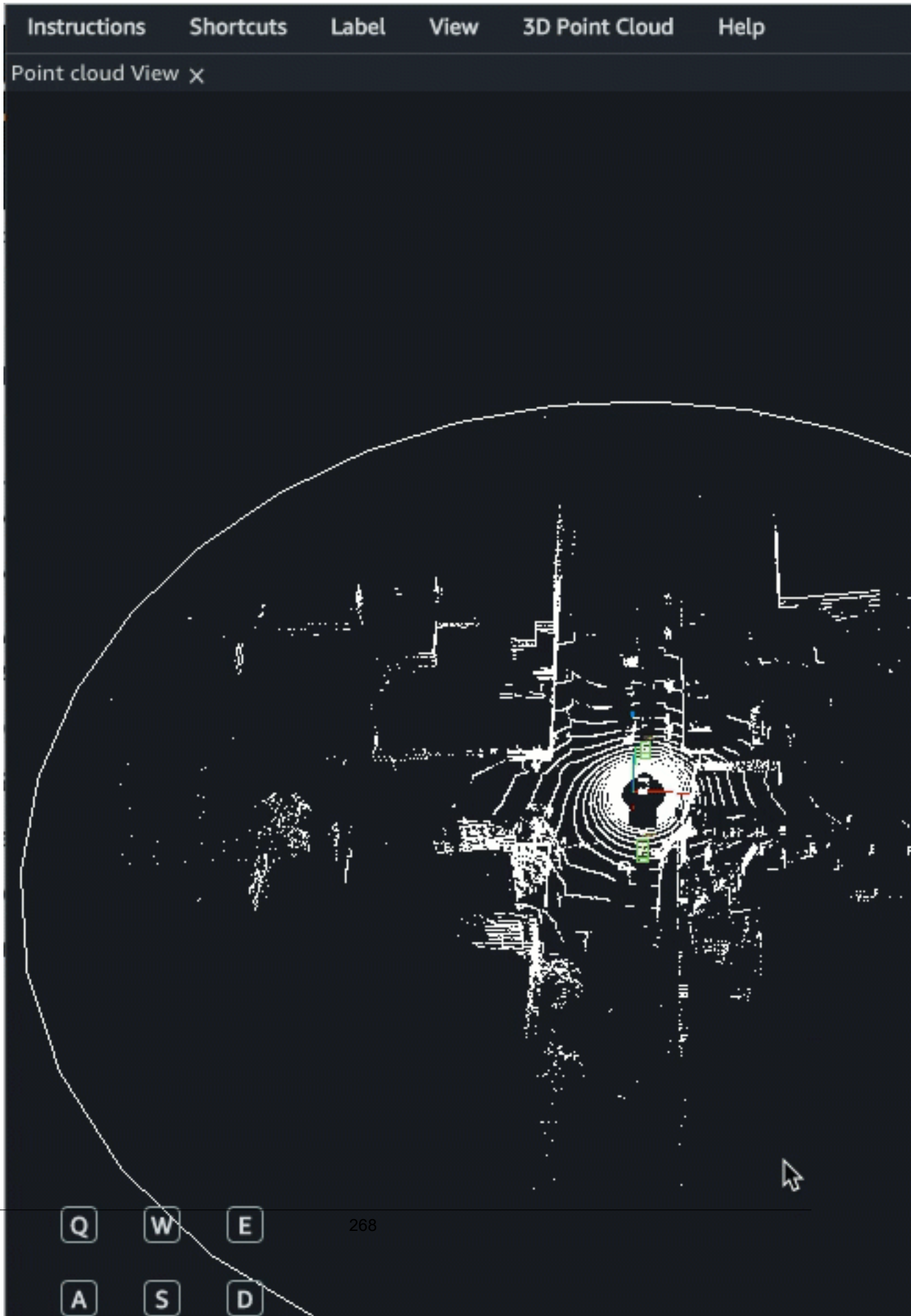

Hello, chopt@amazon.com
Instructions
Shortcuts
Label
View
3D Point Cloud
Help
Shortcuts
Point cloud View
Double click on the point cloud to zoom in
Double click on the label ID to zoom in the object on point cloud.
3D Cuboid Controls
When you are in Edit Cuboid mode
C Create
V Edit
Cmd + Drag Change dimension
Option + Drag Move cuboid
Option + O Fit label to points
Option + G Set to ground
Shift + Drag Rotate cuboid
[Previous label
] Next label
Cmd + , Toggle show/hide label

工作人员可以使用键盘和鼠标以在 3D 场景中导航。他们可以：

- 双击点云中的特定对象以将其放大。
- 使用鼠标滚轮或触控板以放大和缩小点云。
- 同时使用键盘箭头键和 Q、E、A 和 D 键以向上、向下、向左和向右移动。使用键盘 W 和 S 键以放大和缩小。

在工作人员将长方体放置在 3D 场景后，将显示侧视图，其中具有三个投影的侧视图：俯视图、侧视图和后视图。这些侧视图显示放置的长方体内部和周围的点，并帮助工作人员优化该区域中的长方体边界。工作人员可以使用鼠标以放大和缩小每个侧视图。

以下视频说明了在 3D 点云和在侧视图中的移动情况。


Instructions
Shortcuts
Label
View
3D Point Cloud
Help
Point cloud View
Q
W
E
A
S
D

工作人员 UI 的 View (视图) 菜单中提供了其他视图选项和功能。有关工作人员 UI 的全面概述，请参阅工作人员说明页面。

辅助标记工具

使用机器学习和计算机视觉驱动的辅助标记工具执行 3D 点云对象跟踪任务，以帮助工作人员快速准确 Ground Truth 注释 3D 点云。以下辅助标记工具可用于该任务类型：

- 捕捉— 工作人员可以在对象周围添加一个长方体，并使用键盘快捷键或菜单选项让 Gground Truth 的自动适合工具将长方体紧靠对象周围。
- 设置为地面— 在工作人员将长方体添加到 3D 场景后，工作人员可以自动将长方体贴靠到地面上。例如，工作人员可以使用该功能将长方体贴靠到场景中的道路或人行道。
- 多视图标签在工作人员将 3D 长方体添加到 3D 场景后，侧面板将显示前面、侧面和顶部透视图，以帮助工作人员调整紧靠对象周围的长方体。在所有这些视图中，长方体包含一个箭头以指示对象的方向或方位。在工作人员调整长方体时，调整内容将实时显示在所有视图（即，3D 视图、俯视图、侧视图和前视图）上。
- 传感器融合— 如果您提供用于传感器融合的数据，工作人员可以在 3D 场景和 2D 图像中调整注释，注释将实时投影到另一个视图中。此外，工作人员可以选择查看摄像机朝向的方向和摄像机截头锥体。
- 查看选项— 使工作人员能够轻松隐藏或查看长方体、标签文本、地面网格以及其他点属性，例如颜色或强度。工作人员还可以在透视投影和正交投影之间进行选择。

创建 3D 点云对象检测标记作业

您可以使用 SageMaker 控制台或 API 操作创建 3D 点云标记作业。`CreateLabelingJob`. 要为该任务类型创建标记作业，您需要具有以下内容：

- 单帧输入清单文件。要了解如何创建这种类型的清单文件，请参阅创建点云帧输入清单文件 (p. 390)。如果您是 GGround Truth 3D 点云标记模式的新用户，您可能还需要查阅。接受的原始 3D 数据格式 (p. 388).
- 来自私有或供应商人力的工作组。您不能使用 Amazon Mechanical Turk 进行视频帧标签工作。要了解如何创建人力和工作组，请参阅创建和管理人力 (p. 463)。

此外，请确保您已查阅分配 IAM 权限以使用 Ground Truth (p. 447)并满足相应的要求。

可以使用以下几节之一以了解如何使用控制台或 API 创建标记作业。

创建标记作业（控制台）

您可以按照中的说明执行创建标记作业（控制台） (p. 353)以了解如何在 SageMaker 控制台中创建 3D 点云对象检测标记作业。在创建标记作业时，请注意以下事项：

- 输入清单文件必须是单帧清单文件。有关更多信息，请参阅 创建点云帧输入清单文件 (p. 390)。
- （可选）您可以提供标签类别和帧属性。工作人员可以将其中的一个或多个属性分配给注释，以提供有关该对象的更多信息。例如，您可能希望使用 occluded 属性，以使工作人员确定对象何时被部分遮挡。
- 3D 点云标记任务不支持自动数据标记和注释合并。
- 3D 点云对象检测标记作业可能需要几小时才能完成。在选择工作组时，您可以为这些标记作业指定更长的时间限制（最多 7 天或 604,800 秒）。

创建标记作业 (API)

本节介绍了使用 SageMaker API 操作创建标记作业时需要了解的详细信息。`CreateLabelingJob`. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅部分`CreateLabelingJob`.

创建标记作业 (API) (p. 356)概述了 CreateLabelingJob 操作。请按照这些说明进行操作，并在配置请求时执行以下操作：

- 您必须为 HumanTaskUiArn 输入一个 ARN。使用 arn:aws:sagemaker:*<region>*:394669845002:human-task-ui/PointCloudObjectDetection。Replace*<region>*使用Amazon在其中创建标记作业的区域。

  不应具有 UiTemplateS3Uri 参数的条目。
- 输入清单文件必须是单帧清单文件。有关更多信息，请参阅 创建点云帧输入清单文件 (p. 390)。
- 您可以在标签类别配置文件中指定标签、标签类别和帧属性以及工作人员说明。要了解如何创建该文件，请参阅创建具有标签类别和框架属性的标记类别配置文件 (p. 364)。
- 您需要为注释前和注释后 (ACS) Lambda 函数提供预定义的 ARN。这些 ARN 特定于您用于创建标记作业的 Amazon 区域。
  - 要查找注释前 Lambda ARN，请参阅PreHumanTaskLambdaArn. 请使用您在其中创建标记作业的区域以查找正确的 ARN。例如，如果您在 us-east-1 中创建标记作业，则 ARN 为 arn:aws:lambda:us-east-1:432418664414:function:PRE-3DPointCloudObjectDetection。
  - 要查找注释后 Lambda ARN，请参阅AnnotationConsolidationLambdaArn. 请使用您在其中创建标记作业的区域以查找正确的 ARN。例如，如果您在 us-east-1 中创建标记作业，则 ARN 为 arn:aws:lambda:us-east-1:432418664414:function:ACS-3DPointCloudObjectDetection。
- NumberOfHumanWorkersPerDataObject 中指定的工作人员数必须为 1。
- 3D 点云标记作业不支持自动数据标记。您不应在 LabelingJobAlgorithmsConfig 中指定参数值。
- 3D 点云对象检测标记作业可能需要几小时才能完成。您可以在 TaskTimeLimitInSeconds 中为这些标记作业指定更长的时间限制（最多 7 天或 604,800 秒）。

创建 3D 点云对象检测调整或验证标记 Job

您可以使用基本实际情况控制台创建调整或验证标记作业。CreateLabelingJobAPI。要了解有关调整和验证标签作业的更多信息，以及了解如何创建任务，请参阅验证和调整标签 (p. 318).

创建调整标注作业时，标签作业的输入数据可以包括以前的标签或外部来源的标签以及偏航、倾斜和滚动测量值。在调整作业中，音调和滚动将在工作人员界面中可视化，但无法修改。偏航是可调的。

Ground Truth 使用 Tait-Bryan 角度和以下内在旋转来在工作人员 UI 中可视化偏航、俯仰和滚动。首先，旋转根据 z 轴（偏航）应用于车辆。接下来，旋转的车辆根据内在的 y 轴（俯仰）旋转。最后，车辆根据内在的 x”-轴（滚动）旋转。

输出数据格式

在创建 3D 点云对象检测标记作业时，任务将发送给工作人员。在这些工作人员完成其任务时，标签将写入到在创建标记作业时指定的 Amazon S3 存储桶中。输出数据格式决定了在标记任务状态时在 Amazon S3 存储桶中看到的内容（ LabelingJobStatus) 是Completed.

如果您是基本实际情况的新用户，请参阅输出数据 (p. 414)以了解有关基本实际输出数据格式的更多信息。要了解 3D 点云对象检测输出数据格式，请参阅 3D 点云对象检测输出 (p. 430)。

3D 点云对象跟踪

如果您希望工作人员在对象周围添加 3D 长方体并使其适合这些对象，从而跟踪它们在 3D 点云帧中的移动情况，请使用该任务类型。例如，您可以使用该任务类型要求工作人员跟踪车辆在多个点云帧上的移动情况。

对于该任务类型，工作人员标记的数据对象是一个点云帧序列。一个顺序定义为点云帧的时间序列。Ground Truth (基本实际) 使用您提供的序列渲染一系列 3D 点云可视化内容，工作人员可以在工作人员任务界面中在这些 3D 点云帧之间切换。

GGround Truth (x, y, z, rx, ry, l, w, h) 为工作人员提供了一些工具，以便在 3D 场景和投影的侧视图（俯视图、侧视图和后视图）上的三个维度中以 9 个旋转度 (x, y, z, rx, ry, l, w, h) 注释对象。当工作人员在对象周围绘制长方体时，将为该长方体分配唯一的 ID，例如序列中的一辆汽车为 `Car:1`，另一辆汽车为 `Car:2`。工作人员使用该 ID 在多个帧中标记相同的对象。

您也可以提供摄像机数据，以便为工作人员提供有关帧中的场景的更多视觉信息，并帮助工作人员在对象周围绘制 3D 长方体。在工作人员添加 3D 长方体以标识 2D 图像或 3D 点云中的对象时，长方体将显示在另一个视图中。

您可以使用 3D 点云对象跟踪调整任务类型以调整在 3D 点云对象检测标记作业中创建的注释。

如果您是基本 Ground Truth 际 3D 点云标记模式的新用户，我们建议您查阅。3D 点云标记作业概述 (p. 286). 该标记模式与其他 Ground Truth 际任务类型不同，该页面概述了在创建 3D 点云标记作业时应注意的重要详细信息。

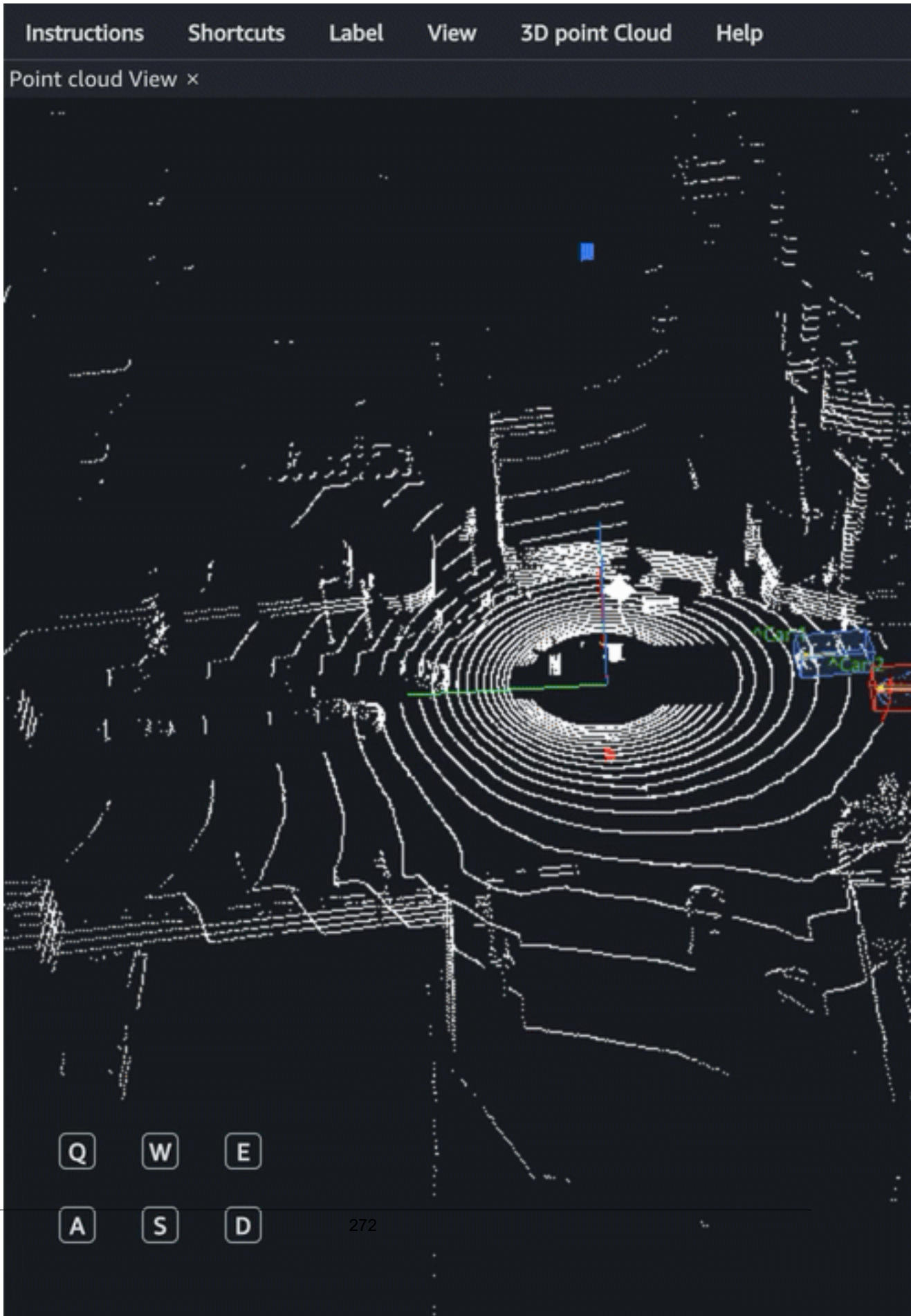
主题

- 查看工作人员任务界面 (p. 271)
- 创建 3D 点云对象跟踪标记作业 (p. 277)
- 创建 3D 点云对象跟踪调整或验证标记 Job (p. 278)
- 输出数据格式 (p. 279)

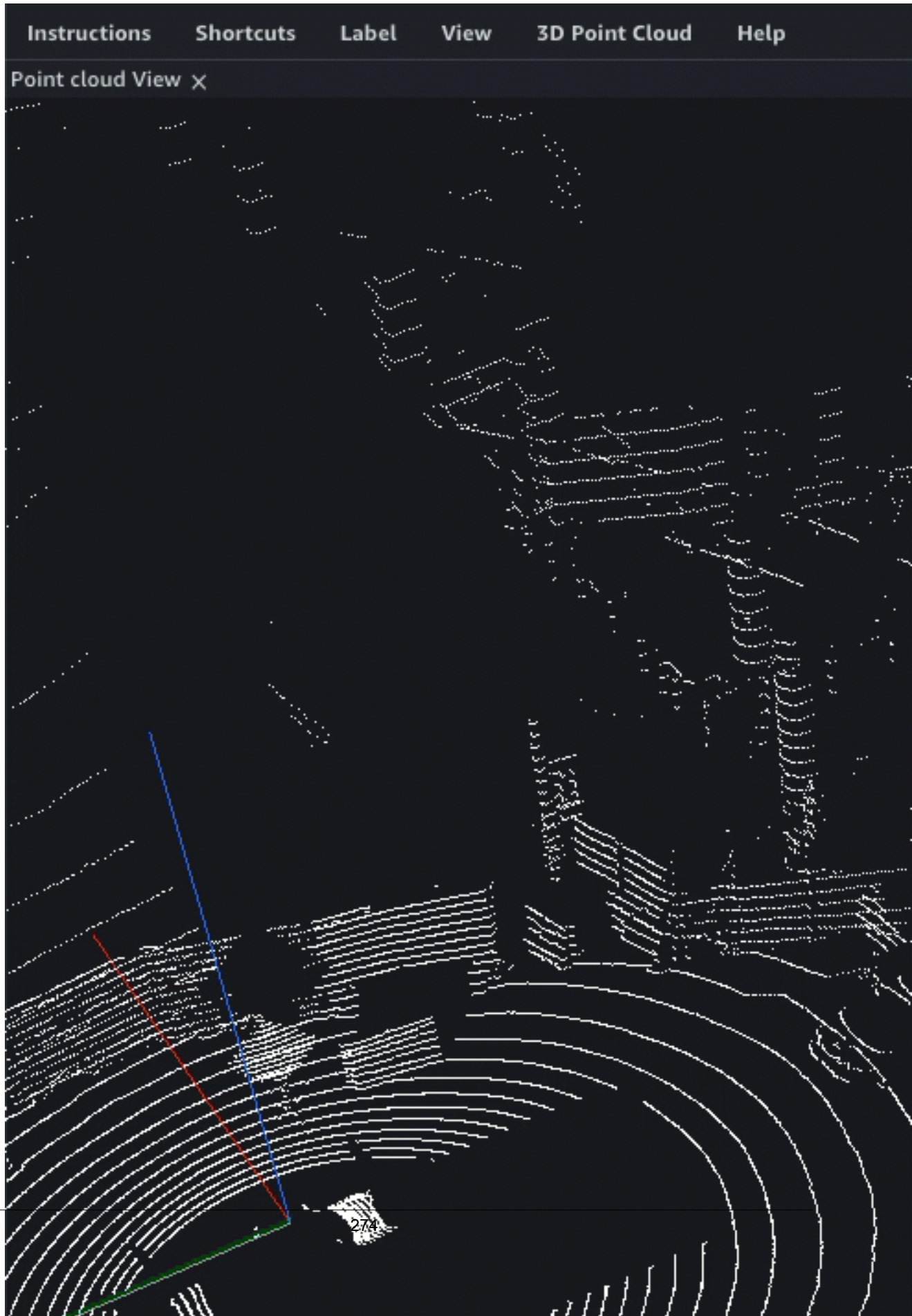
查看工作人员任务界面

Ground Truth 为工作人员提供了 Web 门户和工具以完成 3D 点云对象跟踪注释任务。在创建标记作业 ARN，您 Ground Truth 以在`HumanTaskUiArn`参数。在控制台中使用该任务类型创建标记作业时，将自动使用该 UI。在控制台中创建标记作业时，您可以预览该工作人员 UI 并与之交互。如果您是新用户，建议您使用控制台创建标记作业，以确保标签属性、点云帧和图像（如果适用）按预期方式显示。

以下是 3D 点云对象跟踪工作人员任务界面的 GIF，并说明了工作人员如何浏览序列中的点云帧。注释工具是工作人员任务界面的一部分。它们不适用于预览界面。


Instructions
Shortcuts
Label
View
3D point Cloud
Help
Point cloud View ×
Q
W
E
A
S
D

在工作人员添加一个长方体后，将在序列中具有相同 ID 的所有帧上复制该长方体。在工作人员在另一个帧中调整长方体后，Ground Truth 将插入该对象的移动信息，并在手动调整的帧之间调整所有长方体。以下 GIF 说明了该插入功能。在左下角的导航栏中，红色区域表示手动调整的帧。


Instructions
Shortcuts
Label
View
3D Point Cloud
Help
Point cloud View

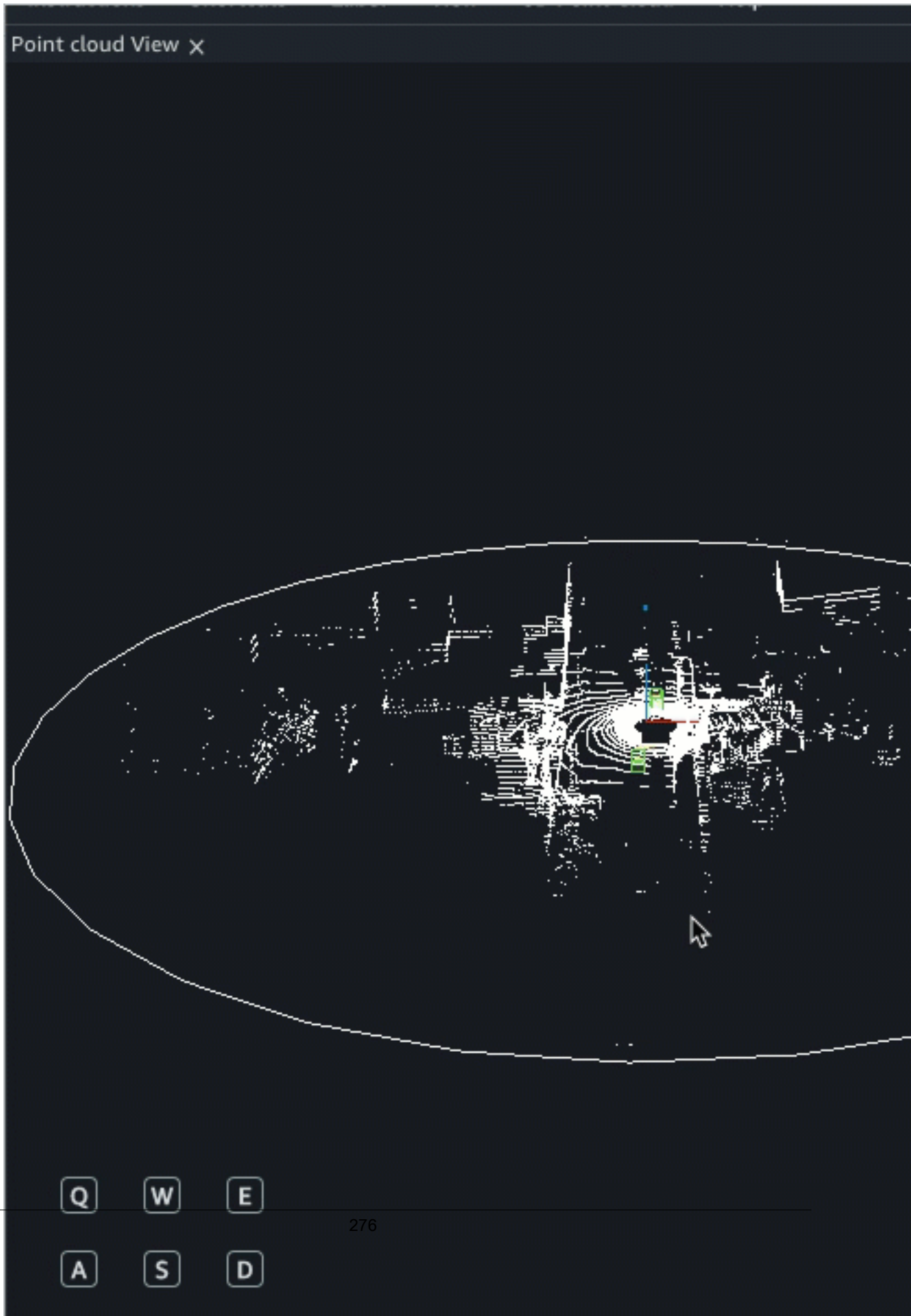
如果您提供用于传感器融合的摄像机数据，图像将与点云帧中的场景进行匹配。这些图像显示在工作人员门户中，如以下 GIF 中所示。

工作人员可以使用键盘和鼠标以在 3D 场景中导航。他们可以：

- 双击点云中的特定对象以将其放大。
- 使用鼠标滚轮或触控板以放大和缩小点云。
- 同时使用键盘箭头键和 Q、E、A 和 D 键以向上、向下、向左和向右移动。使用键盘 W 和 S 键以放大和缩小。

在工作人员将长方体放置在 3D 场景后，将显示侧视图，其中具有三个投影的侧视图：俯视图、侧视图和后视图。这些侧视图显示放置的长方体内部和周围的点，并帮助工作人员优化该区域中的长方体边界。工作人员可以使用鼠标以放大和缩小每个侧视图。

以下视频说明了在 3D 点云和在侧视图中的移动情况。


Point cloud View
Q
W
E
A
S
D

可以使用其他视图选项和功能。有关工作人员 UI 的全面概述，请参阅工作人员说明页面。

工作人员工具

工作人员可以通过以下方法浏览 3D 点云：使用鼠标和键盘快捷键放大和缩小以及在云中的所有方向上移动。如果工作人员单击点云中的一个点，UI 将自动放大到该区域。工作人员可以使用不同的工具在对象周围绘制 3D 长方体。有关更多信息，请参阅辅助标记工具。

在工作人员将 3D 长方体放置在点云后，他们可以使用多种视图调整这些长方体以紧靠汽车周围：直接在 3D 长方体中，在框周围具有三个放大点云透视图的侧视图中以及直接在 2D 图像中（如果包含用于传感器融合的图像）。

View options (视图选项)；使工作人员能够轻松隐藏或查看标签文本、地面网格以及其他点属性。工作人员还可以在透视投影和正交投影之间进行选择。

辅助标记工具

使用 UX、机器学习和计算机视觉驱动的辅助标记工具执行 3D 点云对象跟踪任务，以帮助工作人员快速准确 Ground Truth 注释 3D 点云。以下辅助标记工具可用于该任务类型：

- 标签自动填充— 在工作人员将长方体添加到帧时，具有相同尺寸和方向的长方体将自动添加到序列中的所有帧。
- 插值标签— 在工作人员在两个帧中标记单个对象后，Ground Truth 将使用这些注释插入该对象在这两个帧之间的移动信息。
- 批量标签和属性管理— 工作人员可以批量添加、删除和重命名注释、标记类别属性和帧属性。
  - 工作人员可以在帧前面或后面手动删除给定对象的注释。例如，如果某个对象不再位于第 10 帧后面的场景中，工作人员可以在该帧后面删除该对象的所有标签。
  - 如果工作人员不小心批量删除了对象的所有注释，他们可以重新添加这些注释。例如，如果工作人员在第 100 帧前面删除对象的所有注释，他们可以将其批量添加到这些帧中。
  - 工作人员可以在一个帧中重命名标签，并在所有帧中使用新名称更新分配了该标签的所有 3D 长方体。
  - 工作人员可以使用批量编辑功能在多个帧中添加或编辑标签类别属性和框架属性。
- 捕捉— 工作人员可以在对象周围添加一个长方体，并使用键盘快捷键或菜单选项让 Ground Truth 的自动适合工具将长方体紧靠对象边界周围。
- 适合地面— 在工作人员将长方体添加到 3D 场景后，工作人员可以自动将长方体贴靠到地面上。例如，工作人员可以使用该功能将长方体贴靠到场景中的道路或人行道。
- 多视图标签— 在工作人员将 3D 长方体添加到 3D 场景后，侧面板将显示前面和两个侧面透视图，以帮助工作人员调整紧靠对象周围的长方体。工作人员可以注释 3D 点云，侧面板和调整内容将实时显示在另一个视图中。
- 传感器融合— 如果您提供用于传感器融合的数据，工作人员可以在 3D 场景和 2D 图像中调整注释，注释将实时投影到另一个视图中。
- 自动合并立方体— 如果工作人员确定两个具有不同标签的长方体实际表示一个对象，他们可以在所有帧中自动合并这些长方体。
- 查看选项— 使工作人员能够轻松隐藏或查看标签文本、地面网格以及其他点属性，例如颜色或强度。工作人员还可以在透视投影和正交投影之间进行选择。

创建 3D 点云对象跟踪标记作业

您可以使用 SageMaker 控制台或 API 操作 `CreateLabelingJob` 创建 3D 点云标记作业。要为该任务类型创建标记作业，您需要具有以下内容：

- 序列输入清单文件。要了解如何创建这种类型的清单文件，请参阅创建点云序列输入清单 (p. 395)。如果您是 Ground Truth 3D 点云标记模式的新用户，我们建议您查阅。接受的原始 3D 数据格式 (p. 388).
- 来自私有或供应商人力的工作组。您无法将 Amazon Mechanical Turk 用于 3D 点云标记作业。要了解如何创建人力和工作组，请参阅创建和管理人力 (p. 463)。

此外，请确保您已查阅分配 IAM 权限以使用 Ground Truth (p. 447)并满足相应的要求。

要了解如何使用控制台或 API 创建标记作业，请参阅以下几节。

创建标记作业 (API)

本节介绍了使用 SageMaker API 操作 `CreateLabelingJob` 创建标记作业时需要了解的详细信息。该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅部分`CreateLabelingJob`.

创建标记作业 (API) (p. 356)概述了 `CreateLabelingJob` 操作。请按照这些说明进行操作，并在配置请求时执行以下操作：

- 您必须为 `HumanTaskUiArn` 输入一个 ARN。使用 `arn:aws:sagemaker:<region>:394669845002:human-task-ui/PointCloudObjectTracking`。Replace*<region>*使用Amazon在其中创建标记作业的区域。

  不应具有 `UiTemplateS3Uri` 参数的条目。
- `LabelAttributeName` 必须以 `-ref` 结尾。例如：*ot-labels*`-ref`。
- 输入清单文件必须是点云帧序列清单文件。有关更多信息，请参阅 创建点云序列输入清单 (p. 395)。
- 您可以在标签类别配置文件中指定标签、标签类别和帧属性以及工作人员说明。有关更多信息，请参阅创建具有标签类别和框架属性的标记类别配置文件 (p. 364)以了解如何创建该文件。
- 您需要为注释前和注释后 (ACS) Lambda 函数提供预定义的 ARN。这些 ARN 特定于您用于创建标记作业的 Amazon 区域。
  - 要查找注释前 Lambda ARN，请参阅`PreHumanTaskLambdaArn`. 请使用您在其中创建标记作业的区域以查找以 `PRE-3DPointCloudObjectTracking` 结尾的正确 ARN。
  - 要查找注释后 Lambda ARN，请参阅`AnnotationConsolidationLambdaArn`. 请使用您在其中创建标记作业的区域以查找以 `ACS-3DPointCloudObjectTracking` 结尾的正确 ARN。
- `NumberOfHumanWorkersPerDataObject` 中指定的工作人员数应该为 `1`。
- 3D 点云标记作业不支持自动数据标记。您不应在 `LabelingJobAlgorithmsConfig` 中指定参数值。
- 3D 点云对象跟踪标记作业可能需要几小时才能完成。您可以在 `TaskTimeLimitInSeconds` 中为这些标记作业指定更长的时间限制（最多 7 天或 604,800 秒）。

创建标记作业（控制台）

您可以按照创建标记作业（控制台） (p. 353)中的说明进行操作，以了解如何在 SageMaker 控制台中创建 3D 点云对象跟踪标记作业。在创建标记作业时，请注意以下事项：

- 输入清单文件必须是序列清单文件。有关更多信息，请参阅 创建点云序列输入清单 (p. 395)。
- （可选）您可以提供标签类别属性。工作人员可以将其中的一个或多个属性分配给注释，以提供有关该对象的更多信息。例如，您可能希望使用 occluded 属性，以使工作人员确定对象何时被部分遮挡。
- 3D 点云标记任务不支持自动数据标记和注释合并。
- 3D 点云对象跟踪标记作业可能需要几小时才能完成。在选择工作组时，您可以为这些标记作业指定更长的时间限制（最多 7 天或 604,800 秒）。

创建 3D 点云对象跟踪调整或验证标记 Job

您可以使用基本实际控制台创建调整和验证标记作业。`CreateLabelingJob`API。要了解有关调整和验证标签作业的更多信息，以及了解如何创建任务，请参阅验证和调整标签 (p. 318).

创建调整标注作业时，标签作业的输入数据可以包括以前的标签或外部来源的标签以及偏航、倾斜和滚动测量值。在调整作业中，音调和滚动将在工作人员界面中可视化，但无法修改。偏航是可调的。

Ground Truth 使用 Tait-Bryan 角度和以下内在旋转来在工作人员 UI 中可视化偏航、俯仰和滚动。首先，旋转根据 z 轴（偏航）应用于车辆。接下来，旋转的车辆根据内在的 y 轴（俯仰）旋转。最后，车辆根据内在的 x”-轴（滚动）旋转。

输出数据格式

在创建 3D 点云对象跟踪标记作业时，任务将发送给工作人员。在这些工作人员完成其任务时，其注释写入到在创建标记作业时指定的 Amazon S3 存储桶中。输出数据格式决定了在标记任务状态时在 Amazon S3 存储桶中看到的内容（ LabelingJob状态) 是`Completed`.

如果您是基本实际情况的新用户，请参阅输出数据 (p. 414)以了解有关基本实际输出数据格式的更多信息。要了解 3D 点云对象跟踪输出数据格式，请参阅 3D 点云对象跟踪输出 (p. 433)。

3D 点云语义分割

语义分割涉及将 3D 点云的各个点划分到预指定的类别。如果您希望工作人员为 3D 点云创建点级语义分割掩膜，请使用该任务类型。例如，如果您指定 `car`、`pedestrian` 和 `bike` 类，则工作人员每次选择一个类，然后在点云中为该类适用的所有点指定相同的颜色。

对于该任务类型，工作人员标记的数据对象是单点云帧。Ground Truth 使用您提供的点云数据生成 3D 点云可视化内容。您也可以提供摄像机数据，以便为工作人员提供有关帧中的场景的更多视觉信息，并帮助工作人员绘制对象。当工作人员在 2D 图像或 3D 点云中绘制对象时，绘制内容将显示在另一个视图中。

您可以使用 3D 点云语义分割调整任务类型以调整在 3D 点云对象检测标记作业中创建的注释。

如果您是 Ground Truth 3D 点云标记模式的新用户，我们建议您查阅。3D 点云标记作业概述 (p. 286). 该标记模式与其他 Ground Truth 任务类型不同，该主题概述了在创建 3D 点云标记作业时应注意的重要详细信息。

主题

- 查看工作人员任务界面 (p. 279)
- 创建 3D 点云语义分割标记作业 (p. 285)
- 创建 3D 点云语义分割调整或验证标记 Job (p. 286)
- 输出数据格式 (p. 286)

查看工作人员任务界面

Ground Truth 为工作人员提供了 Web 门户和工具以完成 3D 点云语义分割注释任务。在创建标记作业 ARN，您 Ground Truth 以在`HumanTaskUiArn`参数。在控制台中使用该任务类型创建标记作业时，将自动使用该 UI。在控制台中创建标记作业时，您可以预览该工作人员 UI 并与之交互。如果您是新用户，建议您使用控制台创建标记作业，以确保标签属性、点云帧和图像（如果适用）按预期方式显示。

以下是 3D 点云语义分割工作人员任务界面的 GIF。如果您提供用于传感器融合的摄像机数据，图像将与点云帧中的场景进行匹配。工作人员可以在 3D 点云或 2D 图像中绘制对象，绘制内容将显示在另一个介质的相应位置中。这些图像显示在工作人员门户中，如以下 GIF 中所示。

Instructions
Shortcuts
View
3D Point Cloud
Paint
Help
Point cloud View

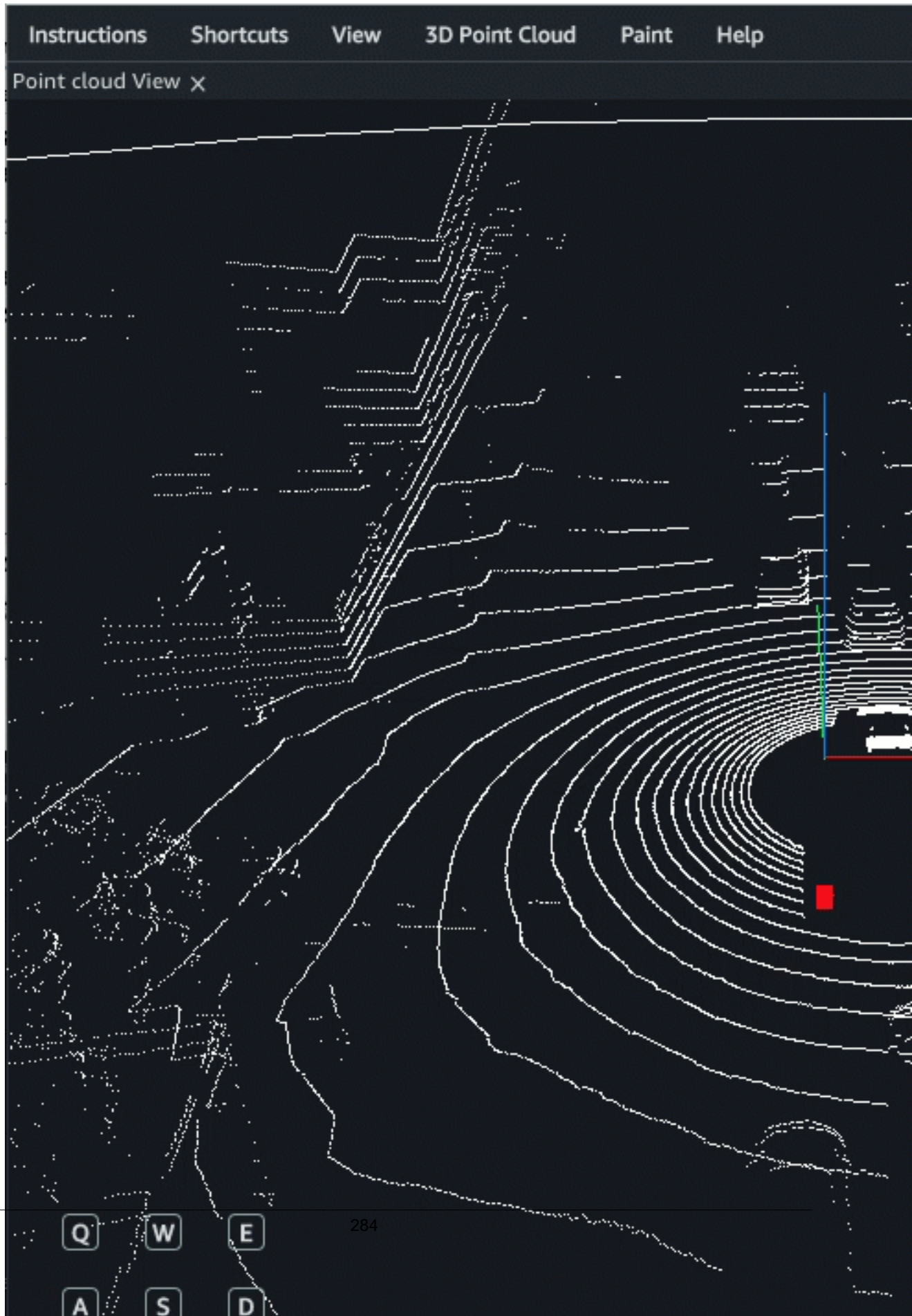
工作人员可以使用键盘和鼠标以在 3D 场景中导航。他们可以：

- 双击点云中的特定对象以将其放大。
- 使用鼠标滚轮或触控板以放大和缩小点云。
- 同时使用键盘箭头键和 Q、E、A 和 D 键以向上、向下、向左和向右移动。使用键盘 W 和 S 键以放大和缩小。

以下视频说明了在 3D 点云中的移动情况。工作人员可以隐藏和重新展开所有侧视图和菜单。在该 GIF 中，已折叠侧视图和菜单。

Instructions
Shortcuts
View
3D Point Cloud
Paint
Help
Point cloud View
Q
W
E
A
S
D

以下 GIF 说明了工作人员如何快速标记多个对象，使用 Unpaint (取消绘制) 选项优化绘制的对象，然后仅查看已绘制的点。


Instructions
Shortcuts
View
3D Point Cloud
Paint
Help
Point cloud View
Q
W
E
A
S
D

可以使用其他视图选项和功能。有关工作人员 UI 的全面概述，请参阅工作人员说明页面。

工作人员工具

工作人员可以通过以下方法浏览 3D 点云：使用鼠标和键盘快捷键放大和缩小以及在云中的所有方向上移动。在您创建语义分割作业时，工作人员可以使用以下工具：

- 画笔；用于绘制和取消绘制对象。工作人员可以通过以下方法绘制对象：选择一种标签类别，然后在 3D 点云中进行绘制。工作人员可以通过以下方法取消绘制对象：从标签类别菜单中选择 Unpaint (取消绘制) 选项并使用画笔擦除绘制内容。
- 多边形工具；工作人员可用来在点云中选择和绘制区域。
- 背景绘制工具；使工作人员能够在他们已注释的对象后面进行绘制，而不更改原始注释。例如，在绘制道路上的所有汽车后，工作人员可以使用该工具绘制道路。
- View options (视图选项)；使工作人员能够轻松隐藏或查看标签文本、地面网格以及其他点属性，例如颜色或强度。工作人员还可以在透视投影和正交投影之间进行选择。

创建 3D 点云语义分割标记作业

您可以使用 SageMaker 控制台或 API 操作创建 3D 点云标记作业。`CreateLabelingJob`. 要为该任务类型创建标记作业，您需要具有以下内容：

- 单帧输入清单文件。要了解如何创建这种类型的清单文件，请参阅创建点云帧输入清单文件 (p. 390)。如果您是 Ground Truth 3D 点云标记模式的新用户，我们建议您查阅。接受的原始 3D 数据格式 (p. 388).
- 来自私有或供应商人力的工作组。您无法将 Amazon Mechanical Turk 工作人员用于 3D 点云标记作业。要了解如何创建人力和工作组，请参阅创建和管理人力 (p. 463)。
- 标签类别配置文件。有关更多信息，请参阅 创建具有标签类别和框架属性的标记类别配置文件 (p. 364)。

此外，请确保您已查阅分配 IAM 权限以使用 Ground Truth (p. 447)并满足相应的要求。

可以使用以下几节之一以了解如何使用控制台或 API 创建标记作业。

创建标记作业（控制台）

您可以按照中的说明进行创建标记作业（控制台） (p. 353)以了解如何在 SageMaker 控制台中创建 3D 点云语义分割标记作业。在创建标记作业时，请注意以下事项：

- 输入清单文件必须是单帧清单文件。有关更多信息，请参阅 创建点云帧输入清单文件 (p. 390)。
- 3D 点云标记任务不支持自动数据标记和注释合并。
- 3D 点云语义分割标记作业可能需要几小时才能完成。在选择工作组时，您可以为这些标记作业指定更长的时间限制（最多 7 天或 604,800 秒）。

创建标记作业 (API)

本节介绍了使用 SageMaker API 操作创建标记作业时需要了解的详细信息。`CreateLabelingJob`. 该 API 为所有 Amazon 开发工具包定义了该操作。要查看此操作支持的特定于语言的 SDK 列表，请查看另请参阅部分:`CreateLabelingJob`.

创建标记作业 (API) (p. 356)页面概述了 `CreateLabelingJob` 操作。请按照这些说明进行操作，并在配置请求时执行以下操作：

- 您必须为 `HumanTaskUiArn` 输入一个 ARN。使用 `arn:aws:sagemaker:<region>:394669845002:human-task-ui/PointCloudSemanticSegmentation`。Replace`<region>`使用Amazon在其中创建标记作业的区域。

不应具有 `UiTemplateS3Uri` 参数的条目。

- `LabelAttributeName` 必须以 `-ref` 结尾。例如：*`ss-labels`*`-ref`。
- 输入清单文件必须是单帧清单文件。有关更多信息，请参阅 创建点云帧输入清单文件 (p. 390)。
- 您可以在标签类别配置文件中指定标签和工作人员说明。请参阅创建具有标签类别和框架属性的标记类别配置文件 (p. 364)以了解如何创建该文件。
- 您需要为注释前和注释后 (ACS) Lambda 函数提供预定义的 ARN。这些 ARN 特定于您用于创建标记作业的 Amazon 区域。
  - 要查找注释前 Lambda ARN，请参阅`PreHumanTaskLambdaArn`. 请使用您在其中创建标记作业的区域以查找正确的 ARN。例如，如果您在 us-east-1 中创建标记作业，则 ARN 为 `arn:aws:lambda:us-east-1:432418664414:function:PRE-3DPointCloudSemanticSegmentation`。
  - 要查找注释后 Lambda ARN，请参阅`AnnotationConsolidationLambdaArn`. 请使用您在其中创建标记作业的区域以查找正确的 ARN。例如，如果您在 us-east-1 中创建标记作业，则 ARN 为 `arn:aws:lambda:us-east-1:432418664414:function:ACS-3DPointCloudSemanticSegmentation`。
- `NumberOfHumanWorkersPerDataObject` 中指定的工作人员数应该为 `1`。
- 3D 点云标记作业不支持自动数据标记。您不应在 `LabelingJobAlgorithmsConfig` 中指定参数值。
- 3D 点云语义分割标记作业可能需要几小时才能完成。您可以在 `TaskTimeLimitInSeconds` 中为这些标记作业指定更长的时间限制（最多 7 天或 604,800 秒）。

创建 3D 点云语义分割调整或验证标记 Job

您可以使用 Ground Truth 控制台创建调整和验证标记作业。`CreateLabelingJob`API。要了解有关调整和验证标签作业的更多信息，以及了解如何创建任务，请参阅验证和调整标签 (p. 318).

输出数据格式

在创建 3D 点云语义分割标记作业时，任务将发送给工作人员。在这些工作人员完成其任务时，其注释写入到在创建标记作业时指定的 Amazon S3 存储桶中。输出数据格式决定了在标记任务状态时在 Amazon S3 存储桶中看到的内容（ LabelingJobStatus) 是`Completed`.

如果您是 Ground Truth 的新用户，请参阅输出数据 (p. 414)以了解有关 Ground Truth 输出数据格式的更多信息。要了解 3D 点云对象检测输出数据格式，请参阅 3D 点云语义分割输出 (p. 429)。

3D 点云标记作业概述

本主题概述了 GGround Truth Trture 3D 点云标记作业的独特功能。您可以使用 3D 点云标记作业，以让工作人员在 3D 点云中标记对象，这些点云是从 3D 传感器（例如 LiDAR）和深度摄像机中生成的，或者是从 3D 重建中生成的（通过拼接无人机等代理捕获的图像）。

作业预处理时间

在创建 3D 点云标记作业时，您需要提供输入清单文件 (p. 388)。输入清单文件可能是：

- 帧输入清单文件；在每一行中具有单个点云帧。
- 序列输入清单文件；在每一行中具有单个序列。序列定义为点云帧的时间序列。

对于这两种类型的清单文件，作业预处理时间（即，Ground Truth 开始向工作人员发送任务之前的时间）取决于您在输入清单文件中提供的点云帧的总数和大小。对于帧输入清单文件，这是清单文件中的行数。对于序列清单文件，这是每个序列中的帧数乘以清单文件中的总序列数或总行数。

此外，每个点云的点数和融合的传感器数据对象（例如图像）数也会影响作业预处理时间。平均而言，GGround Truth Trture 可以在大约 5 分钟内预处理 200 个点云帧。如果您创建具有大量点云帧的 3D 点

云标记作业，作业预处理时间可能会更长。例如，如果您创建一个包含 4 个点云序列的序列输入清单文件，并且每个序列包含 200 个点云，将预处 Ground Truth 800 个点云，因此，作业预处理时间可能大约为 20 分钟。在此期间，标记作业状态为 `InProgress`。

在预处理 3D 点云标记作业时，您将收到 CloudWatch 消息，通知您作业的状态。要查找这些消息，请在标记作业日志中搜索 `3D_POINT_CLOUD_PROCESSING_STATUS`。

适用于帧输入清单文件，您的 CloudWatch 日志将具有类似下面的消息：

```
{
    "labeling-job-name": "example-point-cloud-labeling-job",
    "event-name": "3D_POINT_CLOUD_PROCESSING_STATUS",
    "event-log-message": "datasetObjectId from: 0 to 10, status: IN_PROGRESS"
}
```

事件日志消息 `datasetObjectId from: 0 to 10, status: IN_PROGRESS` 指定在输入清单中已处理的帧数。每次处理一个帧后，您会收到一条新消息。例如，在处理单个帧后，您收到另一条消息以指示 `datasetObjectId from: 1 to 10, status: IN_PROGRESS`。

适用于序列输入清单文件，您的 CloudWatch 日志将具有类似下面的消息：

```
{
    "labeling-job-name": "example-point-cloud-labeling-job",
    "event-name": "3D_POINT_CLOUD_PROCESSING_STATUS",
    "event-log-message": "datasetObjectId: 0, status: IN_PROGRESS"
}
```

事件日志消息 `datasetObjectId from: 0, status: IN_PROGRESS` 指定在输入清单中已处理的序列数。每次处理一个序列后，您会收到一条新消息。例如，在处理单个序列后，在开始处理下一个序列时，您收到一条消息以指示 `datasetObjectId from: 1, status: IN_PROGRESS`。

作业完成时间

工作人员可能需要花几小时的时间才能完成 3D 点云标记作业。您可以在创建标记作业时设置工作人员可处理每个任务的总时间。您可以为工作人员处理任务设置的最长时间为 7 天。默认值为 3 天。

强烈建议您创建工作人员可在 12 小时内完成的任务。在处理任务时，工作人员必须将工作人员 UI 保持打开状态。他们可以随时保存工作，GGround Truth Trture 将每 15 分钟保存一次他们的工作。

使用 SageMaker 时`CreateLabelingJob`在 API 操作中，设置工作人员可以使用的总时间`TaskTimeLimitInSeconds`的参数`HumanTaskConfig`.

在控制台中创建标记作业时，您可以在选择人力类型和工作组时指定该时间限制。

人力

在创建 3D 点云标记作业时，您需要指定一个工作组以完成点云注释任务。您可以从自己的工作人员的私有人力中选择一个工作组，或者从在 Amazon Web Services Marketplace 上选择的供应商人力中选择一个工作组。您无法将 Amazon Mechanical Turk 人力用于 3D 点云标记作业。

要了解供应商人力的更多信息，请参阅管理供应商人力 (p. 466)。

要了解如何创建和管理私有人力，请参阅使用私有人力 (p. 467)。

工作人员用户界面 (UI)

Trture 提供了工作人员用户界面 (UI)、工具和辅助标记功能，以帮助工作人员完成 3D 点云标记任务。

在控制台中创建标记作业时，您可以预览工作人员 UI。

在使用 API 操作创建标记作业时`CreateLabelingJob`，您必须在参数中提供 Ground Truth 提供的 ARN`HumanTaskUiArn`为任务类型指定工作器 UI。您可以使用`HumanTaskUiArn`与 SageMaker 一起使用`RenderUiTemplate`预览工作人员 UI 的 API 操作。

您可以提供在工作人员 UI 中显示的工作人员说明、标签以及标签类别属性（可选）。

标签类别属性

在创建 3D 点云对象跟踪或对象检测标记作业时，可以添加一个或多个标签类别属性. 您可以添加帧属性对于所有 3D 点云任务类型：

- 标签类别属性— 选项（字符串）、自由格式文本框或与一个或多个标签关联的数值字段的列表。工作人员使用它来提供有关标签的元数据。
- Frame 属性— 发送工作人员进行注释的每个点云框架上显示的选项（字符串）、自由格式文本框或数值字段的列表。工作人员使用它来提供有关帧的元数据。

此外，您可以使用标签和帧属性让工作人员在 3D 点云标签验证作业中验证标签。

有关这些属性的更多信息，请参阅以下部分。要了解如何向标注作业添加标签类别和框架属性，请使用创建标记 Job部分上的“Task 类型” 页您选择的。

标签类别属性

将标签类别属性添加到标签，以使工作人员能够提供有关他们创建的注释的更多信息。标签类别属性将添加到单个标签或所有标签中。在将标签类别属性应用于所有标签时，它称为全局标签类别属性.

例如，如果添加标签类别汽车，您可能还希望捕获有关标记的汽车的其他数据，例如，是否遮挡了汽车或汽车的大小。您可以使用标签类别属性以捕获该元数据。在此示例中，如果添加了属性遮挡了到汽车标签类别，你可以分配局部的、完全、不到遮挡了属性，并允许工作人员选择其中一个选项。

创建标签验证作业时，您可以将标签类别属性添加到希望工作人员验证的每个标签中。

帧属性

添加帧属性以使工作人员能够提供有关单个点云帧的更多信息。最多可以指定 10 个帧属性，这些属性将出现在所有帧中。

例如，您可以添加一个允许工作人员输入数字的帧属性。您可能希望使用此属性让工作人员识别他们在特定帧中看到的对象数量。

在另一个示例中，您可能想要提供一个自由格式文本框，让员工能够为问题提供免费表单答案。

在创建标签验证作业时，可以添加一个或多个帧属性，以请求工作人员提供有关点云帧中所有标签的反馈。

工作人员说明

您可以提供工作人员说明，以帮助工作人员完成点云标记任务。您可能希望使用这些说明以执行以下操作：

- 注释对象时的最佳实践和要避免的事项。
- 提供的标签类别属性（对于对象检测和对象跟踪任务）以及如何使用它们的说明。
- 有关在使用键盘快捷键进行标记时如何节省时间的建议。

在创建标记作业时，您可以使用 SageMaker 控制台添加工作人员说明。如果使用 API 操作`CreateLabelingJob`创建标记作业，您可以在标签类别配置文件中指定工作人员说明。

除了您的说明外，Ground Truth 还提供一个链接，以帮助工作人员导航和使用工作人员门户。请在工作人员说明 (p. 290)中选择任务类型以查看这些说明。

拒绝任务

员工可以拒绝任务。

如果说明不清楚、输入数据不正确显示或遇到任务的其他问题，工作人员会拒绝任务。如果每个数据集对象的工作人员数量 (`NumberOfHumanWorkersPerDataObject`) 拒绝任务，数据对象被标记为已过期且不会发送给其他工作人员。

3D 点云标记作业权限要求

在创建 3D 点云标记作业时，除了分配 IAM 权限以使用 Ground Truth (p. 447)中提供的权限要求以外，您还必须将一个 CORS 策略添加到包含输入清单文件的 S3 存储桶中。

将 CORS 权限策略添加到 S3 存储桶

在创建 3D 点云标记作业时，您可以在 S3 中指定输入数据和清单文件所在的存储桶以及存储输出数据的存储桶。这些存储桶可能是相同的。您必须将以下跨域资源共享 (CORS) 策略附加到输入和输出存储桶。如果您使用 Amazon S3 控制台向存储桶添加策略，则必须使用 JSON 格式。

JSON

```
[
        {
            "AllowedHeaders": [
                "*"
            ],
            "AllowedMethods": [
                "GET",
                "HEAD",
                "PUT"
            ],
            "AllowedOrigins": [
                "*"
            ],
            "ExposeHeaders": [
                "Access-Control-Allow-Origin"
            ],
            "MaxAgeSeconds": 3000
        }
    ]
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
    <CORSConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <CORSRule>
        <AllowedOrigin>*</AllowedOrigin>
        <AllowedMethod>GET</AllowedMethod>
        <AllowedMethod>HEAD</AllowedMethod>
        <AllowedMethod>PUT</AllowedMethod>
        <MaxAgeSeconds>3000</MaxAgeSeconds>
        <ExposeHeader>Access-Control-Allow-Origin</ExposeHeader>
        <AllowedHeader>*</AllowedHeader>
    </CORSRule>
    </CORSConfiguration>
```

要了解如何将 CORS 策略添加到 S3 存储桶，请参阅如何通过 CORS 添加跨域资源共享？在亚马逊简单存储服务用户指南中。

工作人员说明

本主题概述了可用于完成 3D 点云标记任务的工作人员门户和工具。首先，从主题中选择您处理的任务类型。

对于调整作业，请选择生成要调整的标签的原始标记作业任务类型。根据需要，查看并调整任务中的标签。

> Important
>
> 建议您使用谷歌 Chrome 浏览器或 Firefox 网络浏览器完成任务。

主题

- 3D 点云语义分割 (p. 290)
- 3D 点云对象检测 (p. 299)
- 3D 点云对象跟踪 (p. 308)

3D 点云语义分割

可以使用该页面熟悉可用于完成 3D 点云语义分割任务的用户界面和工具。

主题

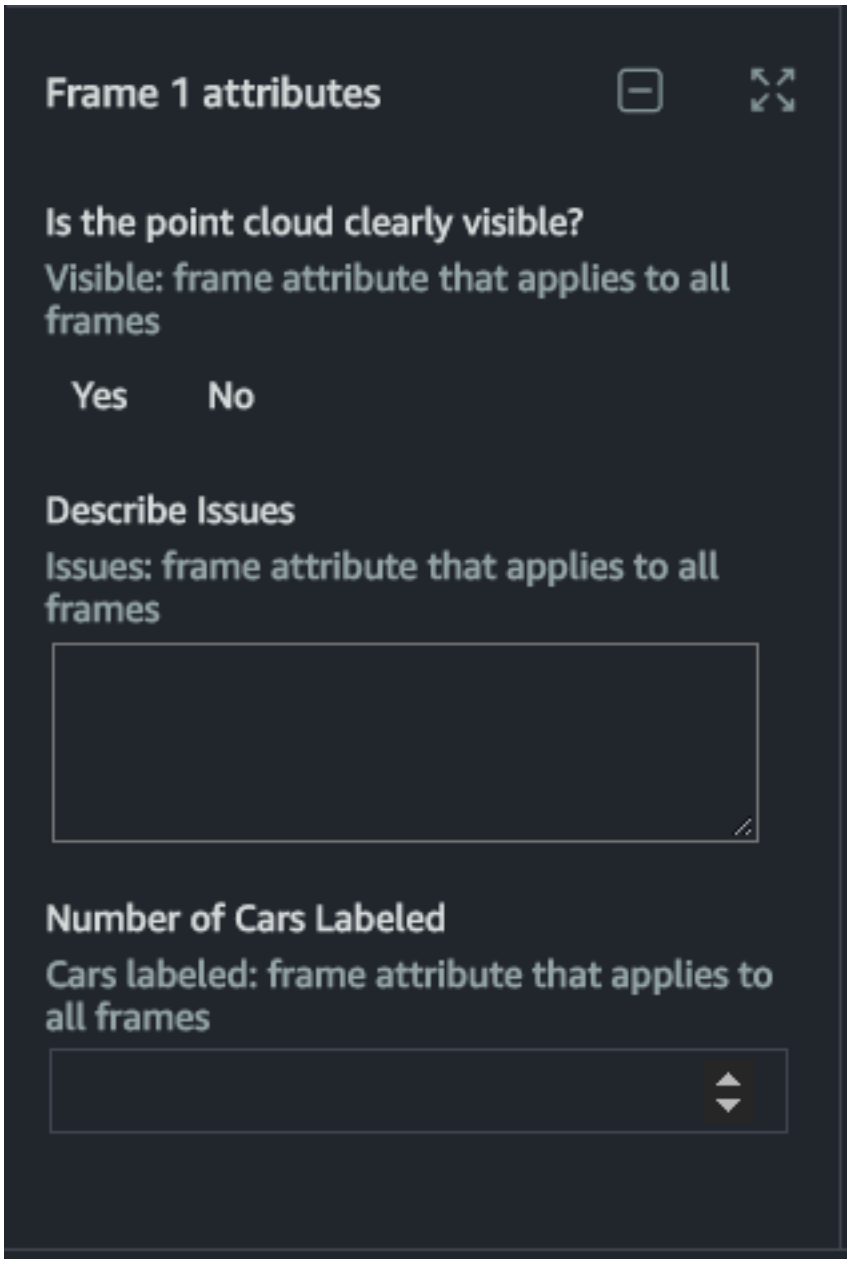
- 您的任务 (p. 290)
- 浏览 UI (p. 295)
- 图标指南 (p. 297)
- Shortcuts (p. 298)
- 释放、停止和恢复以及拒绝任务 (p. 298)
- 保存您的工作并提交 (p. 298)

您的任务

在处理 3D 点云语义分割任务时，您需要从工作人员门户右侧的 Annotations (注释) 菜单中使用 Label Categories (标签类别) 下拉菜单选择一个类别。在选择类别后，使用画笔和多边形工具在该类别适用的 3D 点云中绘制每个对象。例如，如果选择 Car (汽车) 类别，您将使用这些工具在点云中绘制所有汽车。以下视频说明了如何使用画笔工具绘制对象。

如果在工作人员门户中看到一个或多个图像，您可以在这些图像或 3D 点云中绘制，绘制内容将显示在另一个介质中。

你可能会在标签菜单。使用这些属性提示输入有关点云的其他信息。



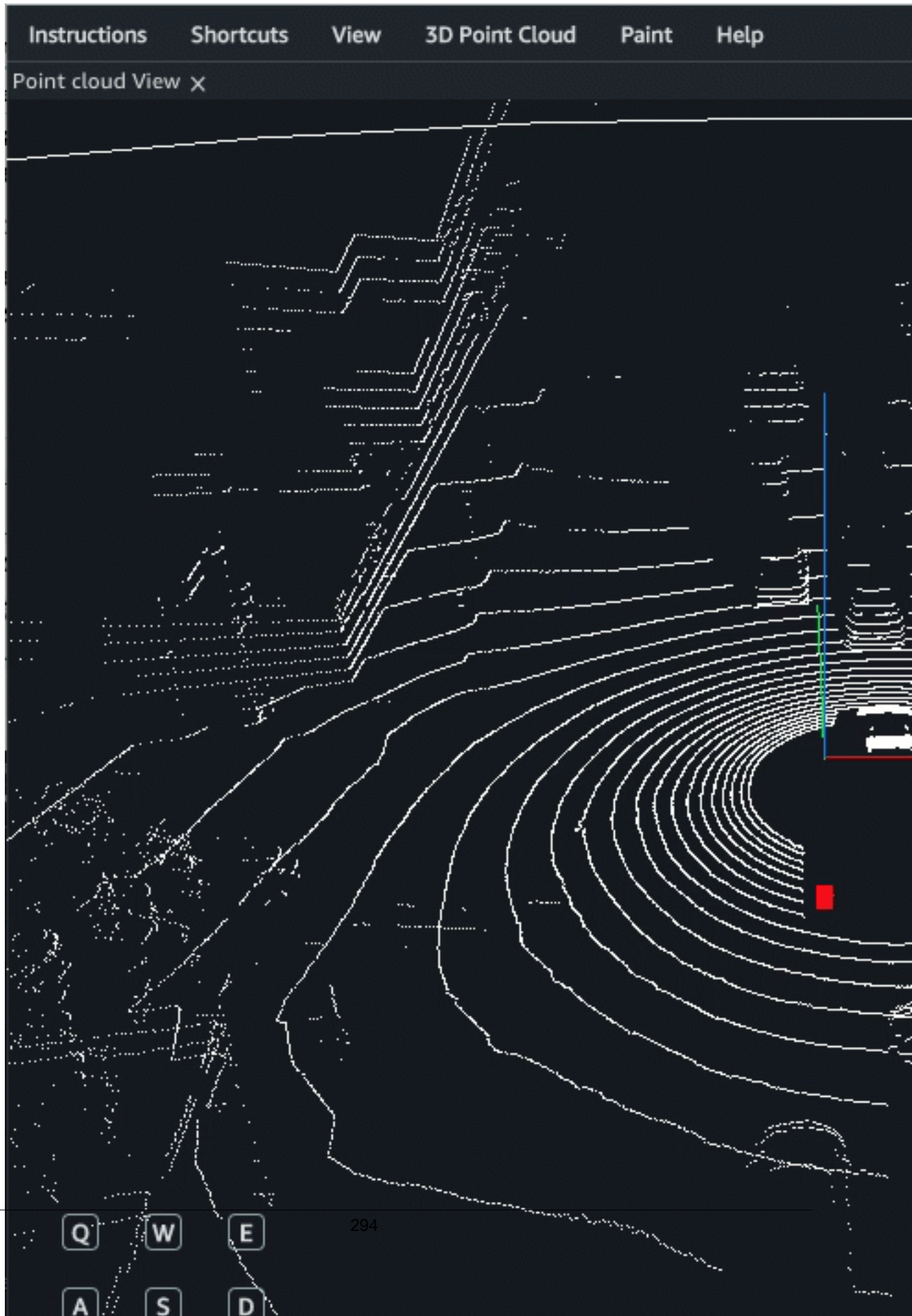
Important

如果您在打开任务时看到已绘制对象，请调整这些注释。

以下视频包含可进行注释的图像。可能在您的任务中看不到图像。

Instructions
Shortcuts
View
3D Point Cloud
Paint
Help
Point cloud View

在使用标签类别绘制一个或多个对象后，您可以从右侧的 Label Category (标签类别) 菜单中选择该类别以仅查看为该类别绘制的点。


Instructions
Shortcuts
View
3D Point Cloud
Paint
Help
Point cloud View
Q
W
E
A
S
D

浏览 UI

您可以使用键盘和鼠标以在 3D 场景中导航。您可以：

- 双击点云中的特定对象以将其放大。
- 使用鼠标滚轮或触控板以放大和缩小点云。
- 同时使用键盘箭头键和 Q、E、A 和 D 键以向上、向下、向左和向右移动。使用键盘 W 和 S 键以放大和缩小。

以下视频说明了在 3D 点云和在侧视图中的移动情况。您可以使用全屏图标隐藏和重新展开所有侧视图。在该 GIF 中，已折叠侧视图和菜单。

Instructions
Shortcuts
View
3D Point Cloud
Paint
Help
Point cloud View
Q
W
E
A
S
D

当您位于工作人员 UI 时，您会看到以下菜单：

- 说明在开始执行任务之前，请查看这些说明。
- Shortcuts— 可以使用该菜单以查看键盘快捷键，它们可用于导航点云和使用提供的注释工具。
- 查看— 可以使用该菜单以打开或关闭不同的视图选项。例如，您可以使用该菜单将地面网格添加到点云中，以及选择点云的投影。
- 3D 点云— 使用该菜单为点云中的点添加其他属性，例如颜色和像素强度。请注意，其中的某些或所有选项可能不可用。
- 绘制— 可以使用该菜单以修改画笔的功能。

在打开任务时，移动场景图标处于打开状态，您可以使用鼠标和屏幕的点云区域中的导航按钮在点云中四处移动。要返回到首次打开任务时看到的原始视图，请选择重置场景图标。

在选择绘制图标后，您可以将绘制内容添加到点云和图像（如果包含）中。您必须再次选择移动场景图标以移动到 3D 点云或图像中的另一个区域。

要折叠右侧的所有面板并使 3D 点云全屏显示，请选择全屏图标。

对于摄像机图像和侧面板，您可以使用以下视图选项：

- C— 在点云视图上查看摄像机角度。
- F— 在点云视图上查看用于捕获该图像的摄像机的截头锥体或视野。
- P— 查看覆盖在图像上的点云。

图标指南

可以使用该表以了解工作人员任务门户中提供的图标。

图标	名称	说明
	画笔	选择该图标以打开画笔工具。要使用该工具，请使用鼠标选择要绘制的对象并在该对象上移动。在选择该工具后，所有绘制的内容与您选择的类别相关联。
	多边形	选择该图标以使用多边形绘制工具。可以使用该工具在要绘制的对象周围绘制多边形。在选择该工具后，在周围绘制多边形的所有内容与您选择的类别相关联。
	重置场景	选择该图标以将点云、侧面板以及所有图像（如果适用）的视图重置为首次打开任务时的原始位置。
	移动场景	选择该图标以移动场景。默认情况下，在首次启动任务时，将选择该图标。

图标	名称	说明
	全屏	选择该图标以使 3D 点云可视化内容全屏显示，并折叠所有侧面板。
	统治者	使用此图标测量点云中的距离（以米为单位）。如果您的说明要求您在距立方体中心或用于捕获数据的对象的给定距离内注释所有对象，则可能需要使用此工具。 选择此图标时，可以用鼠标选择起点（第一个标记）将起点（第一个标记）放置在点云中的任意位置。该工具将自动使用插值在距离所选位置的阈值距离范围内的最近点上放置标记，否则标记将放置在地面上。如果错误地放置了起点，则可以使用 Esc 键恢复标记放置位置。 放置第一个标记后，你会看到一条虚线和一个动态标签，表示你离第一个标记移动的距离。单击点云上的其他位置以放置第二个标记。放置第二个标记时，虚线变为实心，距离被设置。 在设置距离后，您可以通过选择任一标记编辑该距离。您可以通过选择标尺上的任意位置并使用键盘上的 Delete 键来删除标尺。

Shortcuts

Shortcuts (快捷方式) 菜单中列出的快捷方式可以帮助您导航 3D 点云并使用绘制工具。

在开始执行任务之前，建议您查看Shortcuts菜单并熟悉这些命令。

释放、停止和恢复以及拒绝任务

当你打开标签任务时，右上角的三个按钮允许你拒绝任务（拒绝任务），释放它（发布任务），稍后再停止并恢复该操作 (稍后停和恢复)。以下列表描述了选择这些选项之一时会发生什么情况：

- 拒绝任务：只有在任务出现问题时，才应拒绝任务，例如 3D 点云、图像或用户界面的问题。如果您拒绝任务，您将无法返回该任务。
- 发布任务：如果你发布了一个任务，你就会失去在该任务上完成的所有工作。任务发布后，团队中的其他工作人员可以领取它。如果有足够的员工接受任务，您可能无法返回该任务。当你选择此按钮然后选择confirm，您将返回到工作人员门户网站。如果任务仍可用，则其状态将为Available. 如果其他员工拿起它，它将从您的门户中消失。
- 稍后停和恢复：您可以使用稍后停和恢复按钮停止工作并稍后返回任务。您应使用Save（保存）在选择按钮以保存您的工作之前，稍后停和恢复. 当你选择此按钮然后选择confirm，您将返回到工作人员门户，任务状态为Stopped. 你可以选择相同的任务来恢复对它的工作。

  请注意，创建标签任务的人员指定了一个时间限制，在此限制内所有任务都需要完成。如果您没有在该时限内返回并完成此任务，则该任务将过期并且不会提交您的工作。有关详情，请联系您的管理员。

保存您的工作并提交

您应定期保存您的工作。Ground Truth 每 15 分钟自动保存一次您的工作。

在打开任务时，您必须先在其中完成您的工作，然后再按 Submit (提交)。

3D 点云对象检测

可以使用该页面熟悉可用于完成 3D 点云对象检测任务的用户界面和工具。

主题

- 您的任务 (p. 299)
- 浏览 UI (p. 300)
- 图标指南 (p. 307)
- Shortcuts (p. 308)
- 释放、停止和恢复以及拒绝任务 (p. 308)
- 保存您的工作并提交 (p. 308)

您的任务

在处理 3D 点云对象检测任务时，您需要从工作人员门户右侧的 Annotations (注释) 菜单中使用 Label Categories (标签类别) 菜单选择一个类别。在选择类别后，可以使用添加长方体和适合长方体工具，在该类别适用的 3D 点云中使长方体紧靠对象周围。在放置长方体后，您可以直接在点云和右侧显示的三个面板中修改其尺寸、位置和方向。

如果在工作人员门户中看到一个或多个图像，您也可以在这些图像或 3D 点云中修改长方体，编辑内容将显示在另一个介质中。

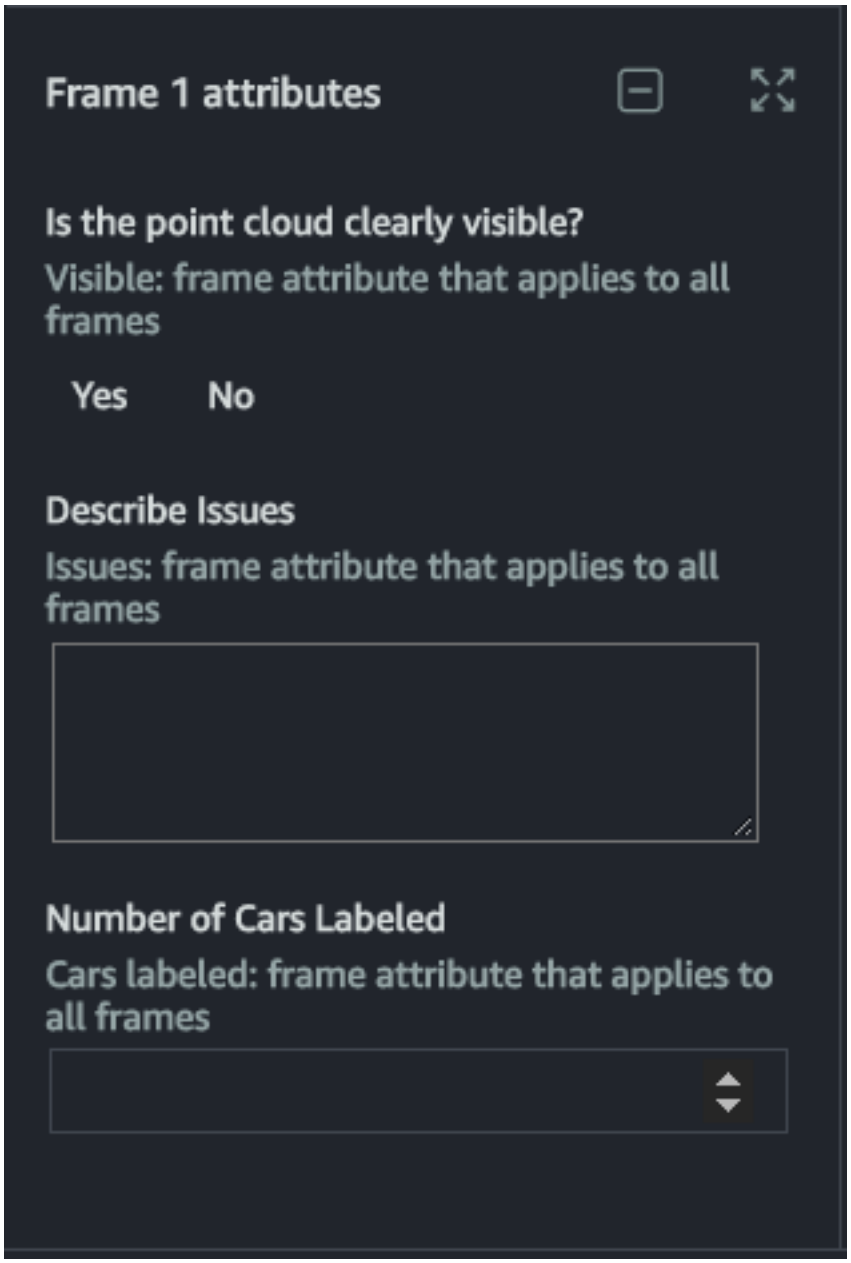
如果在打开任务时看到长方体已添加到 3D 点云中，请调整这些长方体并根据需要添加其他长方体。

要编辑长方体（包括移动长方体，改变长方体方向和更改长方体尺寸），您必须使用快捷键。您可以在 UI 的 Shortcuts (快捷方式) 菜单中看到快捷键的完整列表。以下是在开始执行标记任务之前应熟悉的重要组合键。

Mac 命令	Windows 命令	操作
Cmd + 拖动	Ctrl + 拖动	修改长方体的尺寸。
Option + 拖动	Alt + 拖动	移动长方体。
Shift + 拖动	Shift + 拖动	旋转长方体。
Option + O	Alt + O	使在点周围绘制的长方体紧靠这些点。在使用该选项之前，请确保长方体完全包围相关对象。
Option + G	Alt + G	将长方体放置在地面上。

单个标签可能有一个或多个标签属性。如果标签具有与其关联的标签属性，则当您从Label菜单。为所有标签属性填写必需的值。

你可能会在标签菜单。使用这些属性提示输入有关每个帧的其他信息。



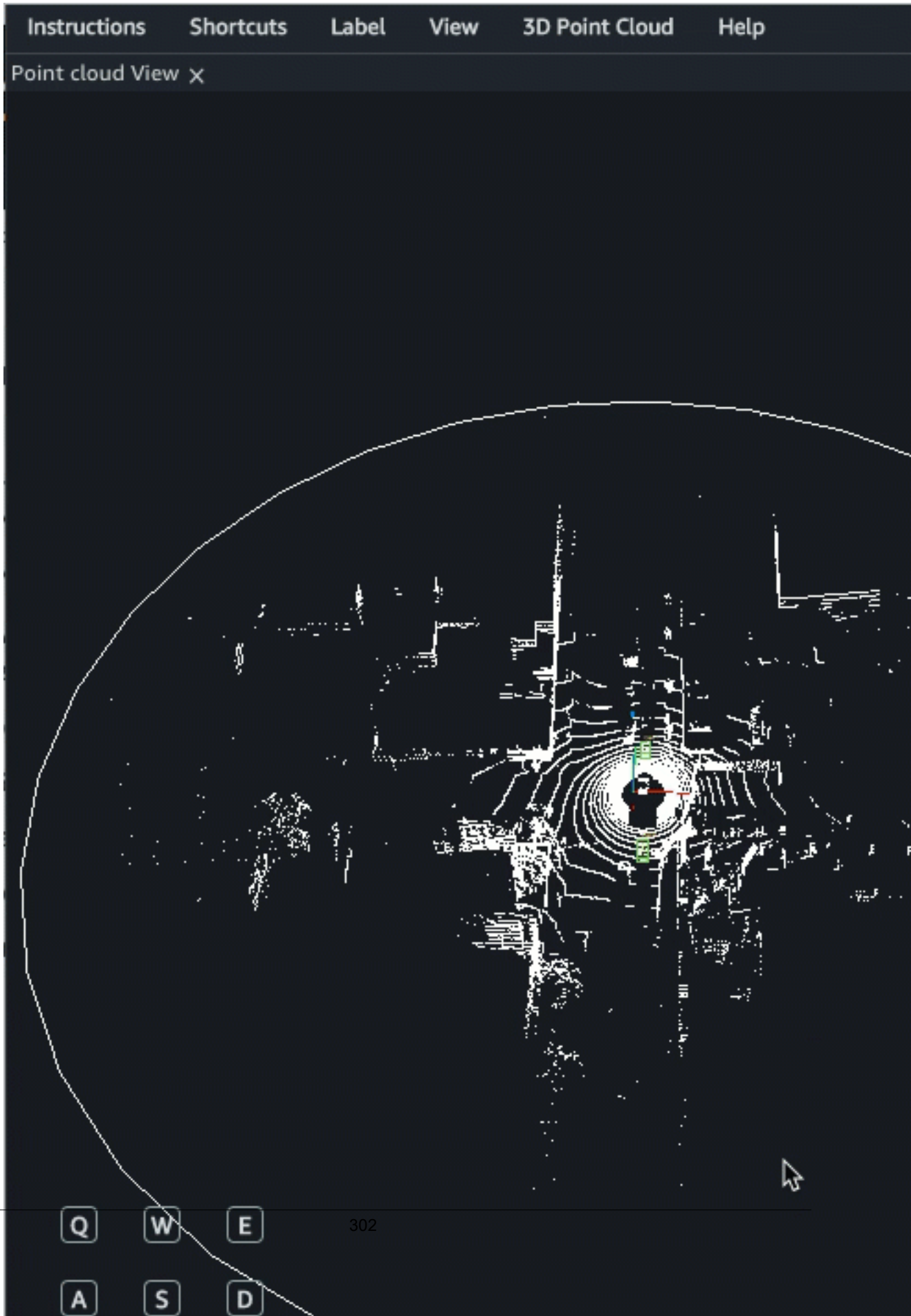
浏览 UI

您可以使用键盘和鼠标以在 3D 场景中导航。您可以：

- 双击点云中的特定对象以将其放大。
- 您可以使用键盘上的 [和] 键放大标签并从一个标签移动到下一个标签。如果没有选择标签，当你选择 [或] 时，UI 将放大到Label列表。
- 使用鼠标滚轮或触控板以放大和缩小点云。
- 同时使用键盘箭头键和 Q、E、A 和 D 键以向上、向下、向左和向右移动。使用键盘 W 和 S 键以放大和缩小。

在您将长方体放置在 3D 场景后，将显示侧视图，其中具有三个投影的侧视图：俯视图、侧视图和后视图。这些侧视图显示放置的长方体内部和周围的点，并帮助工作人员优化该区域中的长方体边界。工作人员可以使用鼠标以放大和缩小每个侧视图。

以下视频说明了在 3D 点云和在侧视图中的移动情况。

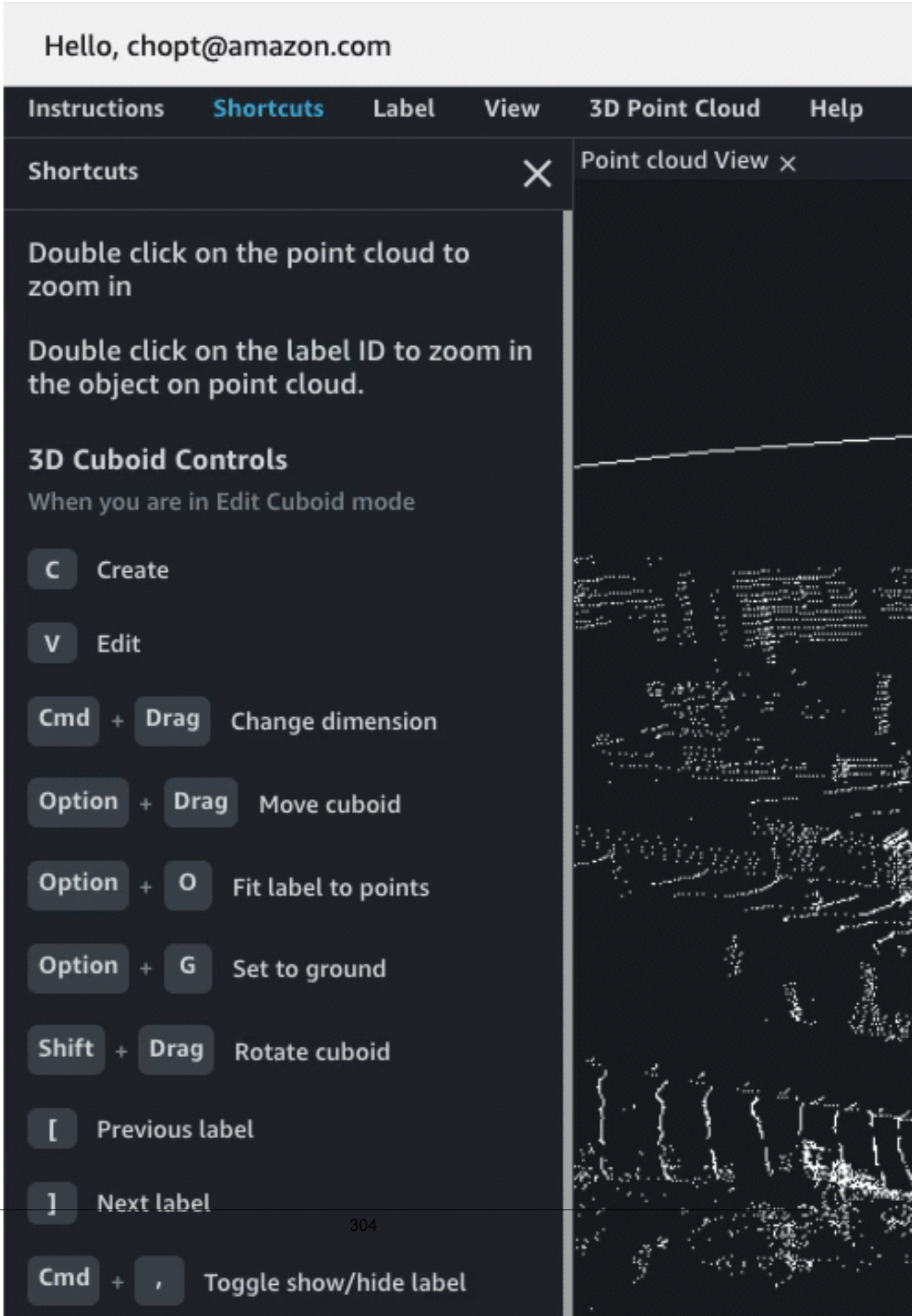

Instructions
Shortcuts
Label
View
3D Point Cloud
Help
Point cloud View
Q
W
E
A
S
D

当您位于工作人员 UI 时，您会看到以下菜单：

- 说明在开始执行任务之前，请查看这些说明。
- Shortcuts— 可以使用该菜单以查看键盘快捷键，它们可用于导航点云和使用提供的注释工具。
- 标签— 使用此菜单以修改长方体。首先，选择一个长方体，然后从该菜单中选择一个选项。该菜单包括辅助标记工具，例如将长方体放置在地面上以及自动使长方体紧靠对象的边界。
- 查看— 可以使用该菜单以打开或关闭不同的视图选项。例如，您可以使用该菜单将地面网格添加到点云中，以及选择点云的投影。
- 3D 点云— 使用该菜单为点云中的点添加其他属性，例如颜色和像素强度。请注意，这些选项可能不可用。

在打开任务时，移动场景图标处于打开状态，您可以使用鼠标和屏幕的点云区域中的导航按钮在点云中四处移动。要返回到首次打开任务时看到的原始视图，请选择重置场景图标。重置视图不会修改注释。

在选择添加长方体图标后，您可以将长方体添加到 3D 点云可视化内容中。在添加长方体后，您可以在三个视图（俯视图、侧视图和前视图）和图像（如果包含）中对其进行调整。


Hello, chopt@amazon.com
Instructions
Shortcuts
Label
View
3D Point Cloud
Help
Shortcuts
Point cloud View
Double click on the point cloud to zoom in
Double click on the label ID to zoom in the object on point cloud.
3D Cuboid Controls
When you are in Edit Cuboid mode
C Create
V Edit
Cmd + Drag Change dimension
Option + Drag Move cuboid
Option + O Fit label to points
Option + G Set to ground
Shift + Drag Rotate cuboid
[Previous label
] Next label
Cmd + , Toggle show/hide label

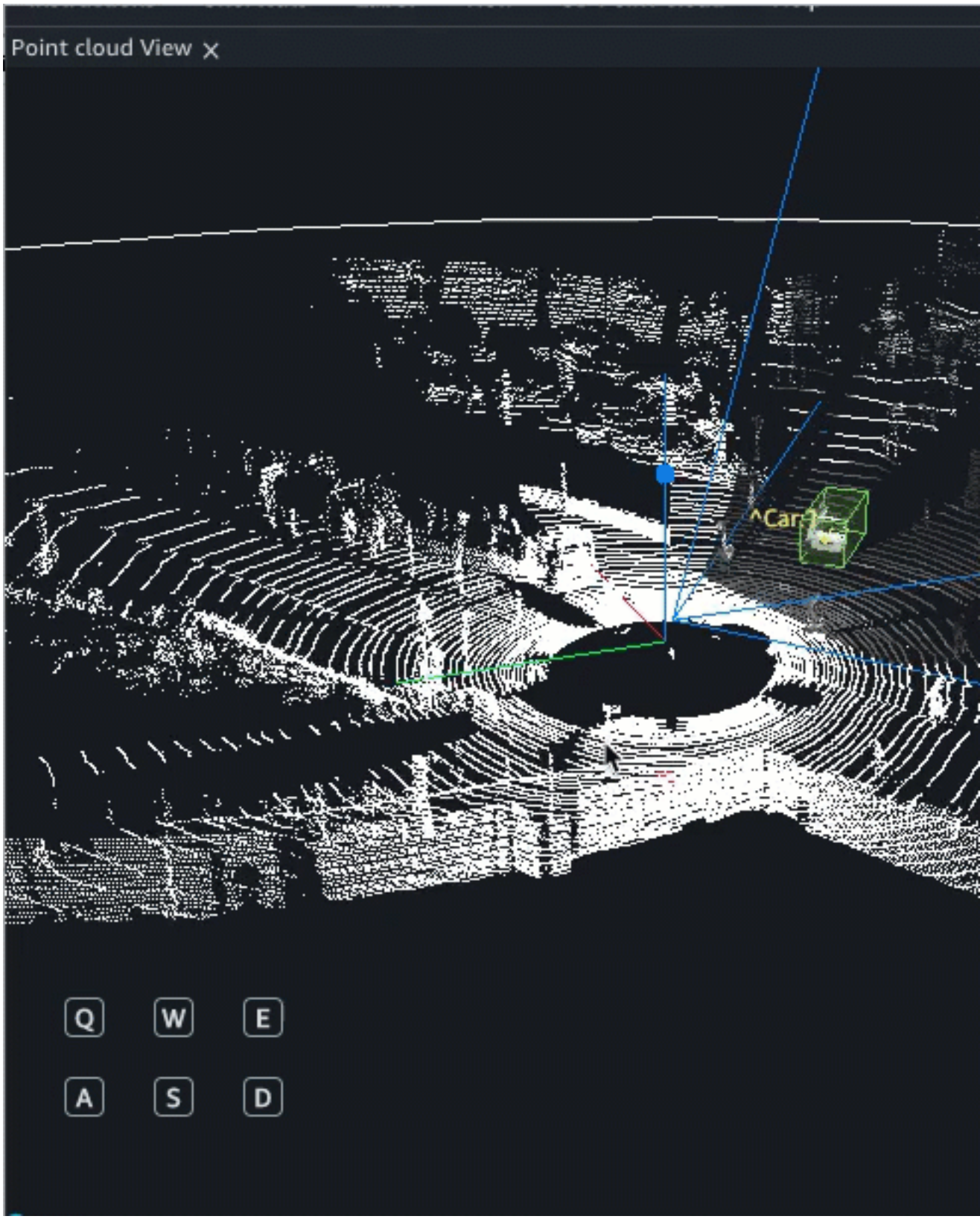
您必须再次选择移动场景图标以移动到 3D 点云或图像中的另一个区域。

要折叠右侧的所有面板并使 3D 点云全屏显示，请选择全屏图标。

如果包含摄像机图像，您可以使用以下视图选项：

- C— 在点云视图上查看摄像机角度。
- F— 在点云视图上查看用于捕获该图像的摄像机的截头锥体或视野。
- P— 查看覆盖在图像上的点云。
- B— 查看图像中的长方体。

以下视频说明了如何使用这些视图选项。F 选项用于查看摄像机的视野（灰色区域），C 选项显示摄像机朝向的方向和摄像机的角度（蓝线），而 B 选项用于查看长方体。


Point cloud View
Q
W
E
A
S
D

图标指南

可以使用该表以了解您在工作人员任务门户中看到的图标。

图标		说明
	添加长方体	选择该图标以添加长方体。您添加的每个长方体与您选择的类别相关联。
	编辑长方体	选择该图标以编辑长方体。在添加长方体后，您可以编辑其尺寸、位置和方向。在添加长方体后，它自动切换到编辑长方体模式。
	统治者	使用此图标测量点云中的距离（以米为单位）。如果您的说明要求您在距立方体中心或用于捕获数据的对象的给定距离内注释所有对象，则可能需要使用此工具。 选择此图标时，可以用鼠标选择起点（第一个标记）将起点（第一个标记）放置在点云中的任意位置。该工具将自动使用插值在距离所选位置的阈值距离范围内的最近点上放置标记，否则标记将放置在地面上。如果错误地放置了起点，则可以使用 Esc 键恢复标记放置位置。 放置第一个标记后，你会看到一条虚线和一个动态标签，表示你离第一个标记移动的距离。单击点云上的其他位置以放置第二个标记。放置第二个标记时，虚线变为实心，距离被设置。 在设置距离后，您可以通过选择任一标记编辑该距离。您可以通过选择标尺上的任意位置并使用键盘上的 Delete 键来删除标尺。
	重置场景	选择该图标以将点云、侧面板以及所有图像（如果适用）的视图重置为首次打开任务时的原始位置。
	移动场景	选择该图标以移动场景。默认情况下，在首次启动任务时，将选择该图标。
	全屏	选择该图标以使 3D 点云可视化内容全屏显示，并折叠所有侧面板。
	显示标签	在 3D 点云可视化内容以及图像（如果适用）中显示标签。

图标		说明
	隐藏标签	在 3D 点云可视化内容以及图像（如果适用）中隐藏标签。
	删除标签	删除标签。

Shortcuts

Shortcuts (快捷方式) 菜单中列出的快捷方式可以帮助您导航 3D 点云，并使用工具添加和编辑长方体。

在开始执行任务之前，建议您查看Shortcuts菜单并熟悉这些命令。您需要使用一些 3D 长方体控件以编辑长方体。

释放、停止和恢复以及拒绝任务

当你打开标签任务时，右上角的三个按钮允许你拒绝任务（拒绝任务），释放它（发布任务），稍后再停止并恢复该操作 (稍后停和恢复)。以下列表描述了选择这些选项之一时会发生什么情况：

- 拒绝任务：只有在任务出现问题时，才应拒绝任务，例如 3D 点云、图像或用户界面的问题。如果您拒绝任务，您将无法返回该任务。
- 发布任务：如果你发布了一个任务，你就会失去在该任务上完成的所有工作。任务发布后，团队中的其他工作人员可以领取它。如果有足够的员工接受任务，您可能无法返回该任务。当你选择此按钮然后选择confirm，您将返回到工作人员门户网站。如果任务仍可用，则其状态将为Available. 如果其他员工拿起它，它将从您的门户中消失。
- 稍后停和恢复：您可以使用稍后停和恢复按钮停止工作并稍后返回任务。您应使用Save（保存）选择按钮以保存您的工作稍后停和恢复. 当你选择此按钮然后选择confirm，您将返回到工作人员门户，任务状态为Stopped. 你可以选择相同的任务来恢复对它的工作。

  请注意，创建标签任务的人员指定了一个时间限制，在此限制内所有任务都需要完成。如果您没有在该时限内返回并完成此任务，则该任务将过期并且不会提交您的工作。有关详情，请联系您的管理员。

保存您的工作并提交

您应定期保存您的工作。Ground Truth 每 15 分钟自动保存一次您的工作。

在打开任务时，您必须先在其中完成您的工作，然后再按 Submit (提交)。

3D 点云对象跟踪

可以使用该页面熟悉可用于完成 3D 点云对象检测任务的用户界面和工具。

主题

- 您的任务 (p. 309)
- 浏览 UI (p. 311)
- 批量编辑标签类别和框架属性 (p. 316)
- 图标指南 (p. 316)

- Shortcuts (p. 318)
- 释放、停止和恢复以及拒绝任务 (p. 318)
- 保存您的工作并提交 (p. 318)

您的任务

在处理 3D 点云对象跟踪任务时，您需要从工作人员门户右侧的 Annotations (注释) 菜单中使用 Label Categories (标签类别) 菜单选择一个类别。在选择类别后，可以使用添加长方体和适合长方体工具，在该类别适用的 3D 点云中使长方体紧靠对象周围。在放置长方体后，您可以直接在点云和右侧显示的三个面板中修改其位置、尺寸和方向。如果在工作人员门户中看到一个或多个图像，您也可以在这些图像或 3D 点云中修改长方体，编辑内容将显示在另一个介质中。

> Important
>
> 如果在打开任务时看到长方体已添加到 3D 点云帧中，请调整这些长方体并根据需要添加其他长方体。

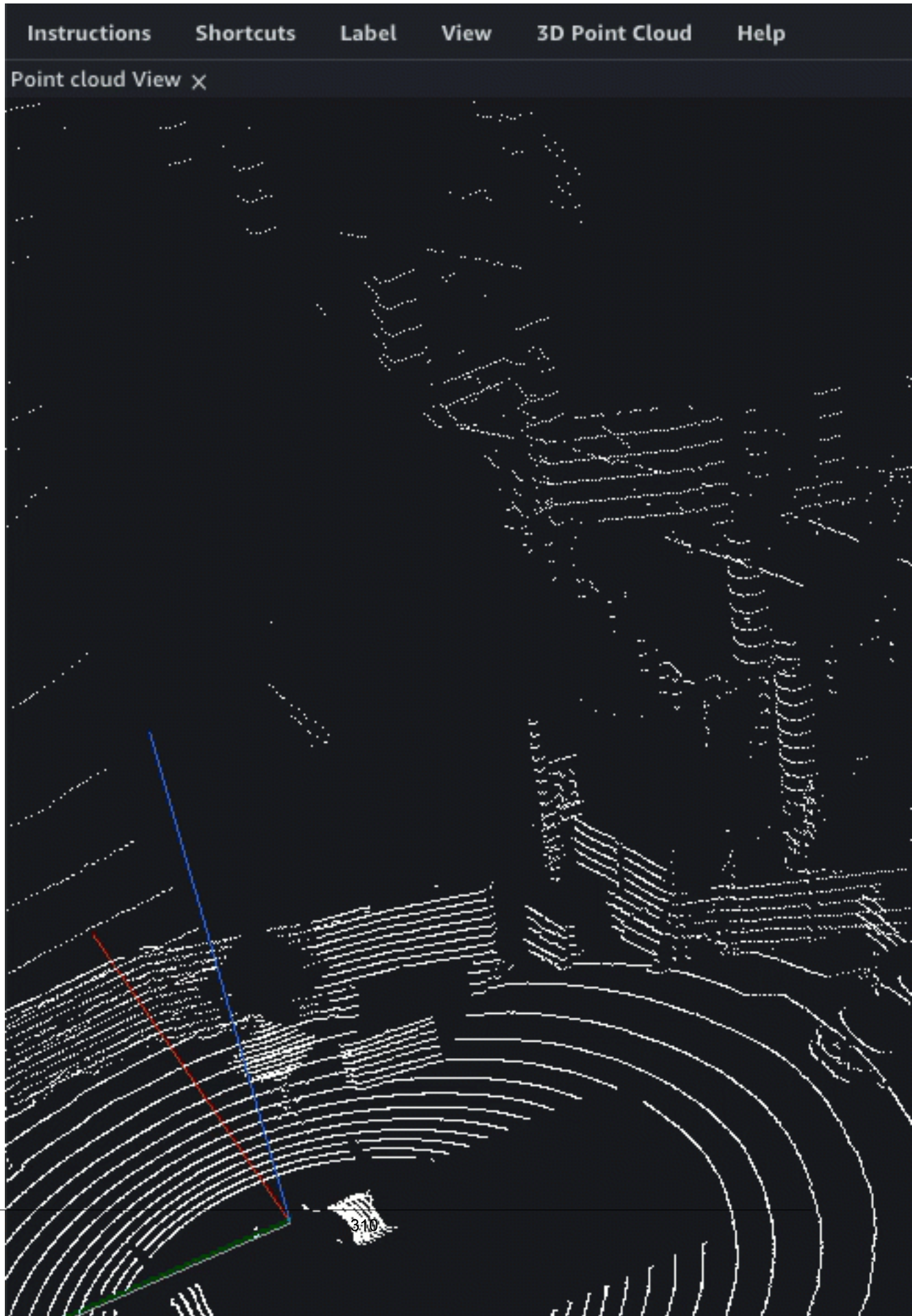
要编辑长方体（包括移动长方体，改变长方体方向和更改长方体尺寸），您必须使用快捷键。您可以在 UI 的 Shortcuts (快捷方式) 菜单中看到快捷键的完整列表。以下是在开始执行标记任务之前应熟悉的重要组合键。

Mac 命令	Windows 命令	操作
Cmd + 拖动	Ctrl + 拖动	修改长方体的尺寸。
Option + 拖动	Alt + 拖动	移动长方体。
Shift + 拖动	Shift + 拖动	旋转长方体。
Option + O	Alt + O	使在点周围绘制的长方体紧靠这些点。在使用该选项之前，请确保长方体完全包围相关对象。
Option + G	Alt + G	将长方体放置在地面上。

在您打开任务时，将加载两个帧。如果任务包含超过两个帧，您需要使用左下角的导航栏或加载帧图标以加载其他帧。在提交之前，您应在所有帧中注释和调整标签。

在将长方体紧靠对象边界周围后，使用 UI 左下角的导航栏导航到另一个帧。如果该同一对象已移到新位置，请添加另一个长方体并使其紧靠对象边界周围。每次手动添加长方体时，您会看到屏幕左下角的帧序列栏变为红色，其中，按时间顺序列出该帧在序列中的位置。

在放置长方体后，UI 自动推断该对象在所有其他帧中的位置。这被称为插值. 您可以使用箭头查看该对象的移动情况以及推断和手动创建的长方体。根据需要，调整推断的长方体。以下视频说明了如何在帧之间导航。以下视频说明了如果在一个帧中添加长方体，然后在另一帧中对其进行调整，UI 将如何自动推断长方体在所有中间帧中的位置。

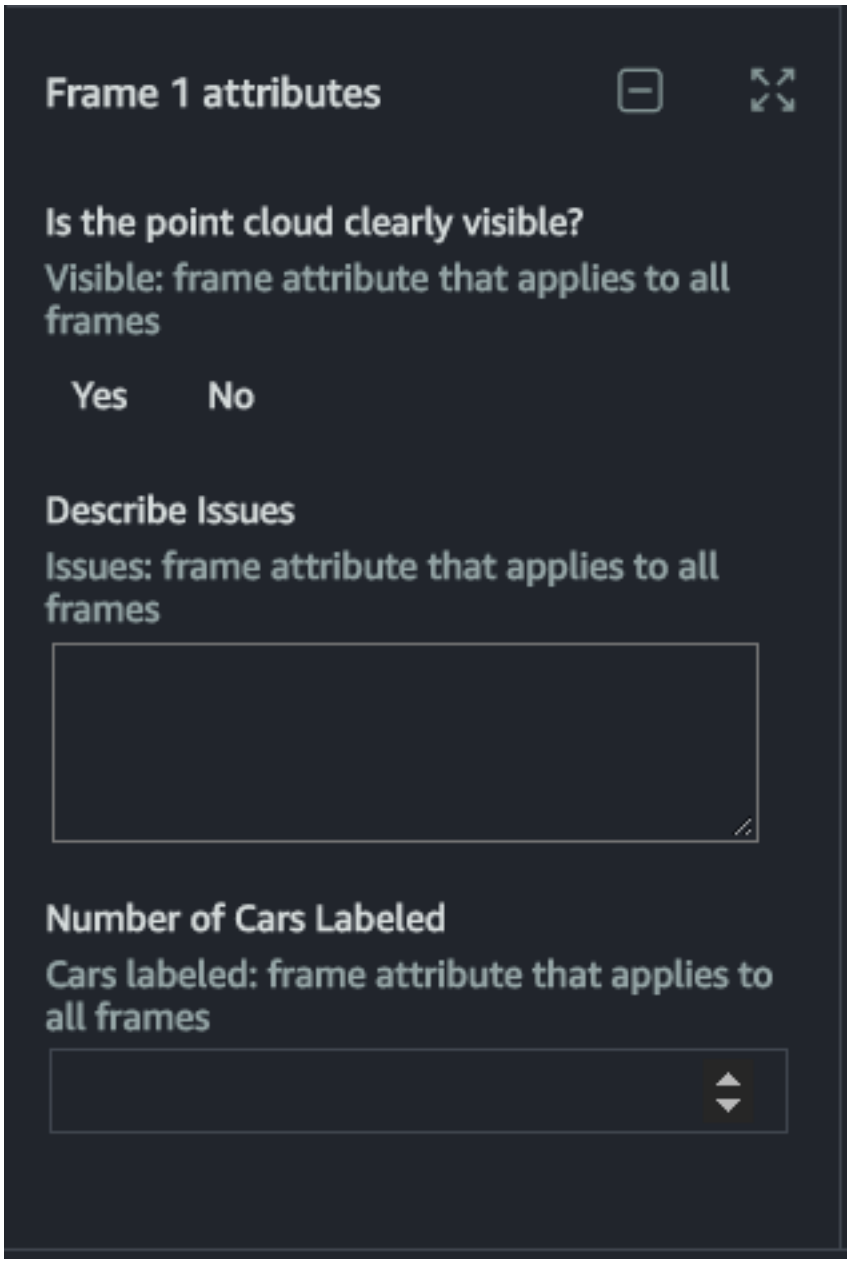

Instructions
Shortcuts
Label
View
3D Point Cloud
Help
Point cloud View

Tip

您可以使用 3D Point Cloud 菜单项关闭帧间的自动立方体插值。Select3D 点云从顶部菜单中，然后选择在帧之间插入立方体. 这将取消选中此选项并停止立方体插值。您可以重新选择此项目以重新打开立方体插值。
关闭立方体插值不会影响已经在帧之间插值的立方体。

单个标签可能有一个或多个标签属性。如果标签具有与其关联的标签属性，则当您从Label菜单。为所有标签属性填写必需的值。

你可能会在Label菜单。这些属性将出现在任务的每个帧中。使用这些属性提示输入有关每个帧的其他信息。



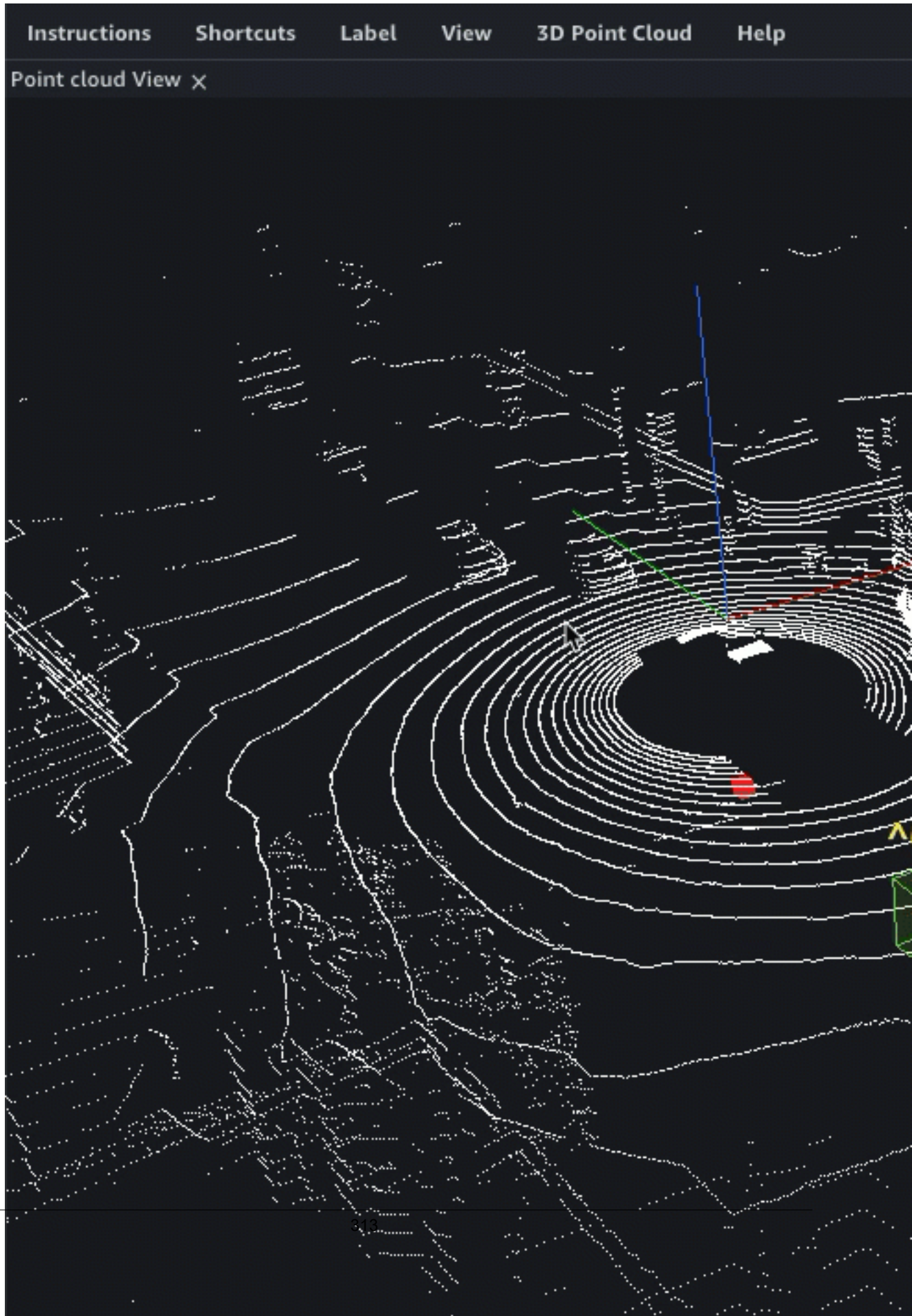
浏览 UI

您可以使用键盘和鼠标以在 3D 场景中导航。您可以：

- 双击点云中的特定对象以将其放大。
- 您可以使用键盘上的 [和] 键放大标签并从一个标签移动到下一个标签。如果没有选择标签，当你选择 [或] 时，UI 将放大到Label列表。
- 使用鼠标滚轮或触控板以放大和缩小点云。
- 同时使用键盘箭头键和 Q、E、A 和 D 键以向上、向下、向左和向右移动。使用键盘 W 和 S 键以放大和缩小。

在您将长方体放置在 3D 场景后，将显示侧视图，其中具有三个投影的侧视图：俯视图、侧视图和后视图。这些侧视图显示放置的长方体内部和周围的点，并帮助工作人员优化该区域中的长方体边界。工作人员可以使用鼠标以放大和缩小每个侧视图。

以下视频说明了在 3D 点云和在侧视图中的移动情况。


Instructions
Shortcuts
Label
View
3D Point Cloud
Help
Point cloud View

当您位于工作人员 UI 时，您会看到以下菜单：

- 说明在开始执行任务之前，请查看这些说明。
- Shortcuts— 可以使用该菜单以查看键盘快捷键，它们可用于导航点云和使用提供的注释工具。
- 标签— 使用此菜单以修改长方体。首先，选择一个长方体，然后从该菜单中选择一个选项。该菜单包括辅助标记工具，例如将长方体放置在地面上以及自动使长方体紧靠对象的边界。
- 查看— 可以使用该菜单以打开或关闭不同的视图选项。例如，您可以使用该菜单将地面网格添加到点云中，以及选择点云的投影。
- 3D 点云— 使用该菜单为点云中的点添加其他属性，例如颜色和像素强度。请注意，这些选项可能不可用。

在打开任务时，移动场景图标处于打开状态，您可以使用鼠标和屏幕的点云区域中的导航按钮在点云中四处移动。要返回到首次打开任务时看到的原始视图，请选择重置场景图标。

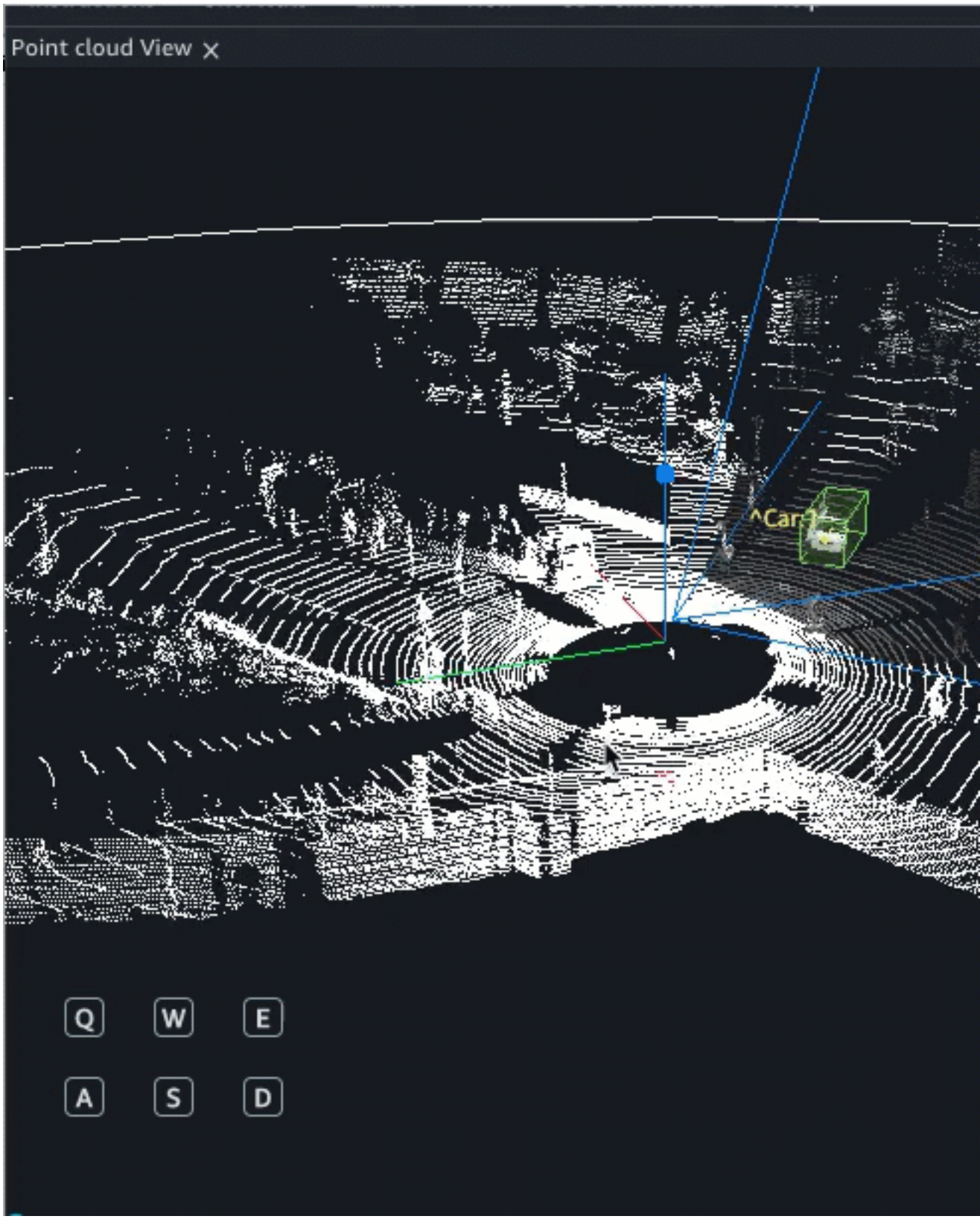
在选择添加长方体图标后，您可以将长方体添加到点云和图像（如果包含）中。您必须再次选择移动场景图标以移动到 3D 点云或图像中的另一个区域。

要折叠右侧的所有面板并使 3D 点云全屏显示，请选择全屏图标。

如果包含摄像机图像，您可以使用以下视图选项：

- C— 在点云视图上查看摄像机角度。
- F— 在点云视图上查看用于捕获该图像的摄像机的截头锥体或视野。
- P— 查看覆盖在图像上的点云。
- B— 查看图像中的长方体。

以下视频说明了如何使用这些视图选项。F 选项用于查看摄像机的视野（灰色区域），C 选项显示摄像机朝向的方向和摄像机的角度（蓝线），而 B 选项用于查看长方体。


Point cloud View
Q
W
E
A
S
D

删除长方体

你可以选择一个方体或标签 ID，然后：

- 删除您正在查看的当前帧中的单个立方体。
- 在正在查看的帧之前或之后删除带有该标签 ID 的所有立方体。
- 删除所有帧中带有该标签 ID 的所有立方体。

删除 cuboid 的一个常见用例是对象是否离开场景。

您可以使用这些选项中的一个或多个选项来删除具有相同标签 ID 的手动放置和插值的立方体。

- 要删除当前所在帧之前或之后的所有立方体，请选择立方体，然后选择标签用户界面顶部的菜单项，然后选择在之前的帧中删除要么在下一帧中删除. 使用快捷键菜单查看可用于这些选项的快捷键。
- 要删除所有帧中的标签，请选择删除所有帧来自 的标签菜单，或者使用快捷方式Shift + 删除在键盘上。
- 要从单个框架中删除单个立方体，请选择立方体，然后选择垃圾桶图标 ( ) 在该标签 ID 旁边Label右侧边栏或使用键盘上的 Delete 键删除该立方体。

如果您在不同的帧中手动放置了多个具有相同标签的立方体，那么当您删除其中一个手动放置的立方体时，所有插值的立方体都会进行调整。发生这种调整是因为在计算插值立方体的位置时，UI 使用手动放置的立方体作为锚点。移除其中一个锚点时，UI 必须重新计算插值立方体的位置。

如果你从框架中删除一个立方体，但后来决定要把它拿回来，你可以使用复制到以前的帧要么复制到下一帧中的选项标签菜单以将立方体分别复制到以下所有上一帧或所有帧中。

批量编辑标签类别和框架属性

您可以批量编辑标签属性和框架属性。

批量编辑属性时，您可以指定要将编辑应用到的一个或多个帧范围。您选择的属性将在该范围内的所有帧中进行编辑，包括指定的起始帧和结束帧。批量编辑标签属性时，您指定的范围必须包含标签属性附加到的标签。如果您指定的帧不包含该标签，您将收到错误。

要批量编辑属性必须首先指定属性的所需值。例如，如果要从更改属性是到否，你必须选择否，然后执行批量编辑。

还可以为尚未填充的属性指定新值，然后使用批量编辑功能在多个帧中填充该值。为此，请为属性选择所需值并完成以下过程。

要批量编辑标签或属性：

1. 使用鼠标右键单击要批量编辑的属性。
2. 指定要使用短划线应用批量编辑的帧范围 (-) 在文本框中。例如，如果要将编辑应用于帧 1 到 10，请输入`1-10`. 如果你想将编辑应用于第二到五帧，八到十和二十输入`2-5,8-10,20`.
3. 选择 Confirm（确认）。

如果收到错误消息，请验证您输入的范围是否有效，并且指定的所有帧中都存在与正在编辑的 label 属性关联的标签（如果适用）。

您可以使用复制到以前的帧和复制到下一帧中的选项标签菜单位于屏幕顶部。

图标指南

可以使用该表以了解您在工作人员任务门户中看到的图标。

图标		说明
	添加长方体	选择该图标以添加长方体。您添加的每个长方体与您选择的类别相关联。
	编辑长方体	选择该图标以编辑长方体。在添加长方体后，您可以编辑其尺寸、位置和方向。在添加长方体后，它自动切换到编辑长方体模式。
	统治者	使用此图标测量点云中的距离（以米为单位）。如果您的说明要求您在距立方体中心或用于捕获数据的对象的给定距离内注释所有对象，则可能需要使用此工具。 选择此图标时，可以用鼠标选择起点（第一个标记）将起点（第一个标记）放置在点云中的任意位置。该工具将自动使用插值在距离所选位置的阈值距离范围内的最近点上放置标记，否则标记将放置在地面上。如果错误地放置了起点，则可以使用 Esc 键恢复标记放置位置。 放置第一个标记后，你会看到一条虚线和一个动态标签，表示你离第一个标记移动的距离。单击点云上的其他位置以放置第二个标记。放置第二个标记时，虚线变为实心，距离被设置。 在设置距离后，您可以通过选择任一标记编辑该距离。您可以通过选择标尺上的任意位置并使用键盘上的 Delete 键来删除标尺。
	重置场景	选择该图标以将点云、侧面板以及所有图像（如果适用）的视图重置为首次打开任务时的原始位置。
	移动场景	选择该图标以移动场景。默认情况下，在首次启动任务时，将选择该图标。
	全屏	选择该图标以使 3D 点云可视化内容全屏显示，并折叠所有侧面板。
	加载帧	选择该图标以加载其他帧。

图标		说明
	隐藏标签	在 3D 点云可视化内容以及图像（如果适用）中隐藏标签。
	显示标签	在 3D 点云可视化内容以及图像（如果适用）中显示标签。
	删除标签	删除标签。只能使用该选项删除您手动创建或调整的标签。

Shortcuts

Shortcuts (快捷方式) 菜单中列出的快捷方式可以帮助您导航 3D 点云，并使用工具添加和编辑长方体。

在开始执行任务之前，建议您查看Shortcuts菜单并熟悉这些命令。您需要使用一些 3D 长方体控件以编辑长方体。

释放、停止和恢复以及拒绝任务

当你打开标签任务时，右上角的三个按钮允许你拒绝任务（拒绝任务），释放它（发布任务），稍后再停止并恢复该操作 (稍后停和恢复)。以下列表描述了选择这些选项之一时会发生什么情况：

- 拒绝任务：只有在任务出现问题时，才应拒绝任务，例如 3D 点云、图像或用户界面的问题。如果您拒绝任务，您将无法返回该任务。
- 发布任务：使用此选项可以释放任务并允许其他人处理任务。当你发布任务时，你会放松该任务完成的所有工作，你的团队中的其他工作人员可以接受它。如果有足够的员工接受任务，您可能无法返回该任务。当你选择此按钮然后选择confirm，您将返回到工作人员门户网站。如果任务仍可用，则其状态将为Available. 如果其他员工拿起它，它将从您的门户中消失。
- 稍后停和恢复：您可以使用稍后停和恢复按钮停止工作并稍后返回任务。您应使用Save（保存）选择按钮以保存您的工作稍后停和恢复. 当你选择此按钮然后选择confirm，您将返回到工作人员门户，任务状态为Stopped. 你可以选择相同的任务来恢复对它的工作。

  请注意，创建标签任务的人员指定了一个时间限制，在此限制内所有任务都需要完成。如果您没有在该时限内返回并完成此任务，则该任务将过期并且不会提交您的工作。有关详情，请联系您的管理员。

保存您的工作并提交

您应定期保存您的工作。Ground Truth 每 15 分钟自动保存一次您的工作。

在打开任务时，您必须先在其中完成您的工作，然后再按 Submit (提交)。

验证和调整标签

当需要验证数据集上的标签时，Amazon SageMaker Ground Truth 提供让工作人员验证标签是否正确或调整之前的标签的功能。

这些类型的作业分为两个不同的类别：

- 标签验证— 工作人员指出现有标签是否正确或评级其质量，并可添加注释来解释其推理。工作人员将无法修改或调整标签。

  如果您创建 3D 点云或视频帧标签调整或验证作业，则可以选择使标签类别属性（不支持 3D 点云语义分割）和帧属性由工作人员编辑。
- 标签调整— 工作人员调整之前的注释以及标签类别和框架属性（如果适用）以更正它们。

以下 Ground Truth内置任务类型支持调整和验证标签作业：

- 边界框
- 语义分割
- 3D 点云对象检测、3D 点云对象跟踪和 3D 点云语义分割
- 所有视频帧对象检测和视频帧对象跟踪任务类型 — 边界框、折线、多边形和关键点

> **Tip**
>
> 对于 3D 点云和视频帧标注验证作业，建议将新的标签类别属性或帧属性添加到标注作业中。工作人员可以使用这些属性来验证单个标签或整个框架。要了解有关标签类别和帧属性的详细信息，请参阅工作人员用户界面 (UI) (p. 287)对于 3D 点云和工作人员用户界面 (UI) (p. 246)用于视频帧。

您可以使用 SageMaker 控制台或 API 启动标签验证和调整作业。

主题

- 创建验证和调整标签作业的要求 (p. 319)
- 创建标签验证 Job（控制台） (p. 320)
- 创建标签调整 Job（控制台） (p. 321)
- 启动标签验证或调整作业 (API) (p. 322)
- 输出清单中的标签验证和调整数据 (p. 324)
- 注意事项和考虑事项 (p. 324)

创建验证和调整标签作业的要求

要创建标签验证或调整作业，您必须满足以下标准。

- 对于非串流标记作业：您使用的输入清单文件必须包含 label 属性名称（`LabelAttributeName`）要调整的标签。当您链接成功完成的标签作业时，输出清单文件将用作新链接作业的输入清单文件。要了解有关每种任务类型生成的输出清单文件格式的更多 Ground Truth 息，请参阅输出数据 (p. 414).

  对于串流标记作业：您发送到调整或验证贴标任务的 Amazon SNS 输入主题的 Amazon SNS 消息必须包含要调整或验证的标签的标签属性名称。要查看如何使用流式标签作业创建调整或验证标签作业的示例，请参阅此Jupyter 笔记本示例在 GitHub 中。
- 验证或调整标签作业的任务类型必须与原始作业的任务类型相同，除非您正在使用图像标签验证 (p. 223)任务类型来验证边界框或语义分割图像标签。有关视频帧任务类型要求的更多详细信息，请参阅下一个要点。
- 对于视频帧注释验证和调整作业，必须使用与之前的标注作业创建批注时使用的相同批注任务类型。例如，如果您创建视频帧对象检测作业以让工作人员在对象周围绘制边界框，然后创建视频对象检测调整作业，则必须指定边界框作为注释任务类型。要了解更多视频帧标记任务类型，请参阅任务类型 (p. 245).
- 为调整或验证标签作业选择的任务类型必须支持审计工作流程。以下 Ground Truth内置任务类型支持调整和验证标签作业：边界框、语义分割、3D 点云对象检测、3D 点云对象跟踪和 3D 点云语义分割，以及所有视频帧对象检测和视频帧对象跟踪任务类型 — 边界框、折线、多边形和关键点。

创建标签验证 Job（控制台）

边界框和语义分段标记作业是通过选择标签验证控制台中的任务类型。要为 3D 点云和视频帧任务类型创建验证作业，必须选择与原始标注作业相同的任务类型，然后选择显示现有标签。可以使用以下几节之一为任务类型创建标签验证作业。

主题

- 创建图像标签验证 Job（控制台）(p. 320)
- 创建点云或视频帧标签验证 Job（控制台）(p. 320)

创建图像标签验证 Job（控制台）

使用以下过程可以使用控制台创建边界框或语义分割验证作业。此过程假定您已经创建了边界框或语义分割标记作业并且其状态为 “完成” (完成)。这是生成要验证的标签的标签作业。

要创建图像标签验证作业：

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/然后选择标记作业.
2. 通过链接 (p. 444)先前的作业或从头开始，同时指定包含已标记的数据对象的输入清单，以启动新的标记作业。
3. 在任务类型窗格中，选择标签验证.
4. 选择 Next（下一步）。
5. 在 Workers (工作人员) 部分中，选择您要使用的人力类型。有关人力选项的更多详细信息，请参阅创建和管理人力 (p. 463)。
6. （可选）在选择人力后，指定任务超时和任务过期时间.
7. 在现有标签显示选项窗格中，系统显示清单中的可用标签属性名称。选择标签属性名称，以标识要工作人员验证的标签。Ground Truth 尝试通过分析清单来检测和填充这些值，但您可能需要设置正确的值。
8. 使用工具设计人员的说明区域，以提供有关要求以前的标记者执行哪些操作以及当前的验证者需要检查哪些内容的上下文。

   您可以添加工作人员选择的新标签来验证标签。例如，您可以要求工作人员验证图像质量，然后提供标签Clear和模糊. 工作人员还可以选择添加评论来解释他们的选择。
9. 选择 See preview (查看预览) 可检查工具是否正确显示之前的标签，并清楚显示标签验证任务。
10. 选择 Create (创建)。这将创建并启动标记作业。

创建点云或视频帧标签验证 Job（控制台）

使用以下过程可以使用控制台创建 3D 点云或视频帧验证作业。此过程假定您已使用生成要验证的标签类型的任务类型创建了标签作业，其状态为 “完成”。

要创建图像标签验证作业：

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/然后选择标记作业.
2. 通过链接 (p. 444)先前的作业或从头开始，同时指定包含已标记的数据对象的输入清单，以启动新的标记作业。
3. 在任务类型窗格中，选择与链接的标签作业相同的任务类型。例如，如果原始标注作业是视频帧对象检测关键点标注作业，请选择该任务类型。
4. 选择 Next（下一步）。
5. 在 Workers (工作人员) 部分中，选择您要使用的人力类型。有关人力选项的更多详细信息，请参阅创建和管理人力 (p. 463)。
6. （可选）在选择人力后，指定任务超时和任务过期时间.
7. 切换旁边的开关显示现有标签.

8. Select验证.
9. 适用于标记属性名称，从清单中选择与要显示以进行验证的标签对应的名称。您只能看到与上一屏幕上选择的任务类型匹配的标签属性名称。Ground Truth 尝试通过分析清单来检测和填充这些值，但您可能需要设置正确的值。
10. 使用工具设计人员的说明区域，以提供有关要求以前的标记者执行哪些操作以及当前的验证者需要检查哪些内容的上下文。

    您无法修改或添加新标签。您可以删除、修改和添加新的标签类别属性或框架属性。建议将新的标签类别属性或框架属性添加到标注作业中。工作人员可以使用这些属性来验证单个标签或整个框架。

    默认情况下，工作人员将无法编辑预先存在的标签类别属性和框架属性。如果要使标签类别或框架属性可编辑，请选择允许工作人员编辑此属性该属性的复选框。

    要了解有关标签类别和帧属性的详细信息，请参阅工作人员用户界面 (UI) (p. 287)对于 3D 点云和工作人员用户界面 (UI) (p. 246)用于视频帧。
11. 选择 See preview (查看预览) 可检查工具是否正确显示之前的标签，并清楚显示标签验证任务。
12. 选择 Create (创建)。这将创建并启动标记作业。

创建标签调整 Job（控制台）

可以使用以下几节之一为任务类型创建标签验证作业。

主题

- 创建图像标签调整 Job（控制台）(p. 321)
- 创建点云或视频帧标签调整 Job（控制台）(p. 322)

创建图像标签调整 Job（控制台）

使用以下过程可以使用控制台创建边界框或语义分割调整标记作业。此过程假定您已经创建了边界框或语义分割标记作业并且其状态为 “完成” (完成)。这是生成要调整的标签的标签作业。

创建图像标签调整作业（控制台）

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/然后选择标记作业.
2. 通过链接 (p. 444)先前的作业或从头开始，同时指定包含已标记的数据对象的输入清单，以启动新的标记作业。
3. 选择与原始标签作业相同的任务类型。
4. 选择 Next（下一步）。
5. 在 Workers (工作人员) 部分中，选择您要使用的人力类型。有关人力选项的更多详细信息，请参阅创建和管理人力 (p. 463)。
6. （可选）在选择人力后，指定任务超时和任务过期时间.
7. Expand现有标签显示选项通过选择标题旁边的箭头。
8. 选中 I want to display existing labels from the dataset for this job (我想显示此作业的数据集中的现有标签) 旁边的框。
9. 适用于标记属性名称，从清单中选择与要显示以进行调整的标签对应的名称。您只能看到与上一屏幕上选择的任务类型匹配的标签属性名称。Ground Truth 尝试通过分析清单来检测和填充这些值，但您可能需要设置正确的值。
10. 使用工具设计人员的说明区域，以提供有关要求以前的标记者执行哪些任务以及当前的验证者需要检查和调整哪些内容的上下文。
11. 选择 See preview (查看预览) 以检查工具是否正确显示之前的标签并清楚地显示任务。
12. 选择 Create (创建)。这将创建并启动标记作业。

创建点云或视频帧标签调整 Job（控制台）

使用以下过程可以使用控制台创建 3D 点云或视频帧调整作业。此过程假定您已使用生成要验证的标签类型的任务类型创建了标签作业，其状态为 “完成”。

要创建 3D 点云或视频帧标签调整作业（控制台）

1. 打开 SageMaker 控制台：https://console.amazonaws.cn/sagemaker/然后选择标记作业.
2. 通过链接 (p. 444)先前的作业或从头开始，同时指定包含已标记的数据对象的输入清单，以启动新的标记作业。
3. 选择与原始标签作业相同的任务类型。
4. 切换旁边的开关显示现有标签.
5. Select调整.
6. 适用于标记属性名称，从清单中选择与要显示以进行调整的标签对应的名称。您只能看到与上一屏幕上选择的任务类型匹配的标签属性名称。Ground Truth 尝试通过分析清单来检测和填充这些值，但您可能需要设置正确的值。
7. 使用工具设计人员的说明区域，以提供有关要求以前的标记者执行哪些操作以及当前的调整者需要检查哪些内容的上下文。

   您不能删除或修改现有标签，但可以添加新标签。您可以删除、修改和添加新的标签类别属性或框架属性。

   默认情况下，工作人员可以编辑预先存在的标签类别属性和框架属性。如果要使标签类别或框架属性不可编辑，请取消选择允许工作人员编辑此属性该属性的复选框。

   要了解有关标签类别和帧属性的详细信息，请参阅工作人员用户界面 (UI) (p. 287)对于 3D 点云和工作人员用户界面 (UI) (p. 246)用于视频帧。
8. 选择 See preview (查看预览) 以检查工具是否正确显示之前的标签并清楚地显示任务。
9. 选择 Create (创建)。这将创建并启动标记作业。

启动标签验证或调整作业 (API)

通过链接已成功完成的作业启动标签验证或调整作业或使用 `CreateLabelingJob` 操作从头开始启动新作业。该过程几乎与使用设置新的标记作业相同。`CreateLabelingJob`，经过一些修改。使用以下部分了解需要进行哪些修改才能将标签作业链接起来创建调整或验证标签作业。

当您使用 Ground Truth API 创建调整或验证标签作业时，必须使用不同的`LabelAttributeName`比原来的标记作业。原始标签作业是用于创建要调整或验证的标签的作业。

> **Important**
>
> 您在中为调整或验证作业识别的标签类别配置文件`LabelCategoryConfigS3Uri`的`CreateLabelingJob`必须包含原始标签作业中使用的相同标签。您可以添加新标签。对于 3D 点云和视频帧作业，您可以将新的标签类别和帧属性添加到标签类别配置文件中。

边界框和语义分割

要创建边界框或语义分割标签验证或调整作业，请使用以下准则指定`CreateLabelingJob`operation.

- 使用`LabelAttributeName`参数以指定要用于已验证或已调整的标签的输出标签名称。你必须使用不同的`LabelAttributeName`比原来的标签作业使用的那个。
- 如果要链接作业，则在自定义 UI 模板中指定要调整或验证的先前标记作业中的标签。要了解如何创建自定义模板，请参阅 创建自定义 Worker 任务模板 (p. 2648)。

确定 UI 模板在`UiTemplateS3Uri`参数。SageMaker 提供可以在自定义模板中使用的小部件来显示旧标签。使用以下 crowd 元素之一中的 `initial-value` 属性来提取需要验证或调整的标签，并将其包含在您的任务模板中：

- crowd-semantic-segmentation (p. 529)在自定义 UI 任务模板中使用此 crowd 元素来指定需要验证或调整的语义分割标签。
- crowd-bounding-box (p. 489)在自定义 UI 任务模板中使用此 crowd 元素来指定需要验证或调整的边界框标签。

- `LabelCategoryConfigS3Uri` 参数必须包含与以前的标记作业相同的标签类别。
- 使用边界框或语义分段调整或验证 lambda ARN`PreHumanTaskLambdaArn`和`AnnotationConsolidationLambdaArn`：
  - 对于边界框，调整标签作业 lambda 函数 ARN 以`AdjustmentBoundingBox`验证 lambda 函数 ARN 以`VerificationBoundingBox`.
  - 对于语义分段，调整标记作业 lambda 函数 ARN 以`AdjustmentSemanticSegmentation`验证 lambda 函数 ARN 以`VerificationSemanticSegmentation`.

3D 点云和视频帧

- 使用`LabelAttributeName`参数以指定要用于已验证或已调整的标签的输出标签名称。你必须使用不同的`LabelAttributeName`比原来的标签作业使用的那个。
- 您必须使用人工任务 UI Amazon 资源名称 (ARN)`HumanTaskUiArn`) 用于原始标签作业。要查看支持的 ARN，请参阅`HumanTaskUiArn`.
- 在标签类别配置文件中，您必须指定标签属性名称 (`LabelAttributeName`) 用于创建调整或验证标签作业的上一个标签作业`auditLabelAttributeName`参数。
- 您可以指定标记作业是否为验证要么调整使用标记作业`editsAllowed`标签类别配置文件中的参数由`LabelCategoryConfigS3Uri`参数。
  - 适用于验证为作业添加标签，您必须使用`editsAllowed`参数以指定不能修改所有标签。`editsAllowed`必须将其设为`"none"`在每个条目中`labels`. 或者，您可以指定工作人员是否可以调整标签类别属性和框架属性。
  - 或者，对于调整为作业添加标签，你可以使用`editsAllowed`参数来指定工作人员可以或不能修改的标签、标签类别属性和框架属性。如果不使用此参数，则所有标签、标签类别属性和框架属性都将可调整。

  要了解有关的更多信息`editsAllowed`参数和配置标签类别配置文件，请参阅标签类别配置文件架构 (p. 365).
- 使用 3D 点云或视频帧调整 lambda ARN`PreHumanTaskLambdaArn`和`AnnotationConsolidationLambdaArn`对于调整和验证标签作业：
  - 对于 3D 点云，调整和验证标签作业 lambda 函数 ARN 以`Adjustment3DPointCloudSemanticSegmentation`、`Adjustment3DPointCloudObjectTracking`, 和`Adjustment3DPointCloudObjectDetection`分别用于 3D 点云语义分割、对象检测和对象跟踪。
  - 对于视频帧，调整和验证标签作业 lambda 函数 ARN 以`AdjustmentVideoObjectDetection`和`AdjustmentVideoObjectTracking`分别用于视频帧物体检测和物体跟踪。

Ground Truth 将标签验证或调整作业中的输出数据存储在 S3 存储桶中指定的 S3 存储桶中。`S3OutputPath`的参数`CreateLabelingJob`operation. 有关标签验证或调整标记作业中的输出数据的更多信息，请参阅 输出清单中的标签验证和调整数据 (p. 324)。

输出清单中的标签验证和调整数据

Amazon SageMaker Ground Truth 将标签验证数据写入标签元数据中的输出清单。它将两个属性添加到元数据中：

- type 属性，值为 groundtruth/label-verification。
- worker-feedback 属性，具有 comment 值数组。工作人员输入注释时添加此属性。如果没有注释，则不会显示该字段。

以下示例输出清单显示标签验证数据的显示方式：

```
{
  "source-ref":"S3 bucket location",
  "verify-bounding-box":"1",
  "verify-bounding-box-metadata":
  {
    "class-name": "bad",
    "confidence": 0.93,
    "type": "groundtruth/label-verification",
    "job-name": "verify-bounding-boxes",
    "human-annotated": "yes",
    "creation-date": "2018-10-18T22:18:13.527256",
    "worker-feedback": [
      {"comment": "The bounding box on the bird is too wide on the right side."},
      {"comment": "The bird on the upper right is not labeled."}
    ]
  }
}
```

调整任务的工作人员输出类似于原始任务的工作人员输出，但它包含调整后的值以及 adjustment-status 属性（值为 adjusted 或 unadjusted）以指示是否进行了调整。

有关不同任务的输出的更多示例，请参阅 输出数据 (p. 414)。

注意事项和考虑事项

要在创建标签验证或调整作业时获得预期行为，请仔细验证输入数据。

- 如果您使用的是图像数据，请验证清单文件是否包含十六进制 RGB 颜色信息。
- 为节省处理成本，请筛选数据以确保标记作业输入清单中不包含不需要的对象。
- 添加所需的 Amazon S3 权限以确保您的输入数据得到正确处理。

当您使用 Ground Truth API 创建调整或验证标签作业时，必须使用不同的LabelAttributeName比原来的标记作业。

语义分割作业的颜色信息要求

为了在验证或调整任务中正确重现颜色信息，该工具需要清单中的十六进制 RGB 颜色信息（例如 #FFFFFF 表示白色）。当您设置语义分割验证或调整作业时，工具检查清单以确定是否存在此信息。如果找不到它，Amazon SageMaker Ground Truth 将显示错误消息和结束作业设置。

在语义分割工具的先前迭代中，类别颜色信息不会以十六进制 RGB 格式输出到输出清单。在采用验证和调整工作流程的同时，将这一功能引入了输出清单。因此，较旧的输出清单与此新工作流程不兼容。

在开始作业之前筛选您的数据

Amazon SageMaker Ground Truth 将处理输入清单中的所有对象。如果您有一个部分标记的数据集，则可能需要使用创建自定义清单。Amazon S3 Select 查询在输入清单上。未标记的对象单独失败，但它们不会导致作业失败，并且可能会产生处理成本。筛选掉您不希望验证的对象会降低您的成本。

如果使用控制台创建验证作业，则可以使用此处提供的筛选工具。如果您使用 API 创建作业，请根据需要使数据筛选成为工作流程的一部分。

创建自定义标记工作流

本文档指导您完成使用自定义标记模板设置工作流的过程。要了解有关启动标记作业的更多信息，请参阅开始使用 (p. 206). 在该节中，在选择 Task type (任务类型) 时，请选择 Custom labeling task (自定义标记任务)，然后按照该节中的说明对其进行配置。

主题

- 第 1 步：设置您的员工队伍 (p. 325)
- 第 2 步：创建自定义工作人员任务模板 (p. 326)
- 第 3 步：使用处理Amazon Lambda (p. 330)
- 演示模板：使用注释图片crowd-bounding-box (p. 343)
- 演示模板：使用的标记意图crowd-classifier (p. 346)
- 通过 API 实现的自定义工作流 (p. 350)

有关创建自定义标记工作流的更多信息，请参使用亚马逊构建自定义数据标签工作 SageMaker Ground Truth.

第 1 步：设置您的员工队伍

在此步骤中，您使用控制台确定要使用的工作人员类型，并为工作人员类型进行所需的后续选择。此步骤假定您目前已经完成了开始使用 (p. 206)部分中的步骤，并已选择 Custom labeling task (自定义标记任务) 作为任务类型。

配置人力。

1. 首先从 Worker types (工作人员类型) 中选择一个选项。目前提供三种类型：
   - Public (公有) 使用由 Amazon Mechanical Turk 提供支持的按需独立承包商人力。按任务对这些人力付费。
   - Private (私有) 使用您的员工或承包商处理需要保留在组织内的数据。
   - 供应商使用专注于提供数据标记服务的第三方供应商，可通过AmazonMarketplace。
2. 如果选择 Public (公有) 选项，将要求您设置 number of workers per dataset object (每个数据集对象的工作人员数量)。如果让多个工作人员对同一个对象执行同一个任务，则可能有助于提高结果的准确性。默认为三个。您可以提高或降低此数目，具体取决于所需的准确性。

   还要求您使用下拉菜单设置 Price per task (每个任务的价格)。该菜单根据完成任务所需的时间建议价格点。

   确定这一点的推荐方法是首先通过 private (私有) 人力对任务运行简短测试。测试提供了完成任务所需的时间的现实估计值。然后，您可以在 Price per task (每个任务的价格) 菜单上选择您估计所需的范围。如果平均时间超过 5 分钟，请考虑将任务拆分成更小的单位。

下一页

第 2 步：创建自定义工作人员任务模板 (p. 326)

第 2 步：创建自定义工作人员任务模板

一个工作者任务模板是 Ground Truth 用来自定义工作人员用户界面 (UI) 或人工任务 UI 的文件。您可以使用 HTML、CSS、JavaScript 创建工作线程任务模板，液体模板语言, 和Crowd HTML 元素. Liquid 用于自动化模板，Crowd HTML 元素可用于包含常见的注释工具并提供提交给 Ground Truth 的逻辑。

- 开始基本模板 (p. 326)
- 在本地开发模板 (p. 326)
- 使用外部资产 (p. 326)
- 跟踪您的变量 (p. 327)
- 简单示例 (p. 327)
- 使用 Liquid 添加自动化 (p. 328)
- 端到端演示 (p. 330)
- 下一页 (p. 330)

使用以下主题了解如何创建工作线程任务模板。你可以在上看到 Ground Truth 工作人员任务模板示例存储库GitHub.

开始基本模板

您可以在 Ground Truth 控制台中使用模板编辑器开始创建模板。该编辑器包括许多预先设计的基本模板以及 HTML 和 Crowd HTML 元素自动填充功能。

要访问 Ground Truth 自定义模板编辑器：

1. 按照中的说明操作创建标记作业（控制台）(p. 353)然后选择Custom (自定义)用于标签作业任务类型.
2. 当你选择时下一步，你将能够访问模板编辑器和基本模板自定义标签任务设置部分。
3. （可选）从下拉菜单中选择基本模板模板. 如果你更喜欢从头开始创建模板，请选择Custom (自定义)从下拉菜单中获取最小的模板框架。

在本地开发模板

尽管您需要在控制台中测试模板将如何处理传入的数据，但您可以通过在 HTML 文件的顶部添加此代码，在浏览器中测试模板的 HTML 和自定义元素的外观。

Example

这会加载必要的代码来呈现自定义 HTML 元素。如果您希望在首选编辑器而非在控制台中开发您模板的外观，则使用此方法。

但请记住，这不会解析您的变量。您可能希望在本地开发时将它们替换示例内容。

使用外部资产

亚马逊 SageMaker Ground Truth 自定义模板允许嵌入外部脚本和样式表。例如，以下代码块演示了如何添加位于`https://www.example.com/my-enhancement-styles.css`转到模板。

Example

```
<script src="https://www.example.com/my-enhancment-script.js"></script>
```

```
<link rel="stylesheet" type="text/css" href="https://www.example.com/my-enhancement-
styles.css">
```

如果您遇到错误，请确保您的原始服务器随资产一起发送正确的 MIME 类型和编码标头。

例如，远程脚本的 MIME 和编码类型为：`application/javascript;CHARSET=UTF-8`.

对于远程样式表，MIME 和编码类型为：`text/css;CHARSET=UTF-8`.

跟踪您的变量

在下面构建示例的过程中，将有一个步骤用于向示例添加变量，以表示可能会随任务或工作人员不同而变化的数据片段。如果您以其中一个示例模板开始，您将需要确保您知道它已经使用的变量。当你创建预注释时 AmazonLambda 脚本，其输出将需要包含您选择保留的任何这些变量的值。

用于变量的值可能来自于清单文件。将为注释前 Lambda 提供数据对象中的所有键值对。如果这是一个简单的传递脚本，将数据对象中的值键与模板中的变量名称相匹配是将这些值传递到工作人员看到的任务表单的最简单方法。

简单示例

所有任务都以 `<crowd-form> </crowd-form>` 元素开始和结束。与标准 HTML `<form>` 元素类似，所有格式代码都应位于它们之间。

对于简单的推文分析任务，请使用 `<crowd-classifier>` 元素。它需要以下属性：

- 名称 - 用于格式输出结果的变量名称。
- 类别 - JSON 格式的一系列可能的答案。
- 标题 - 用于注释工具的标题

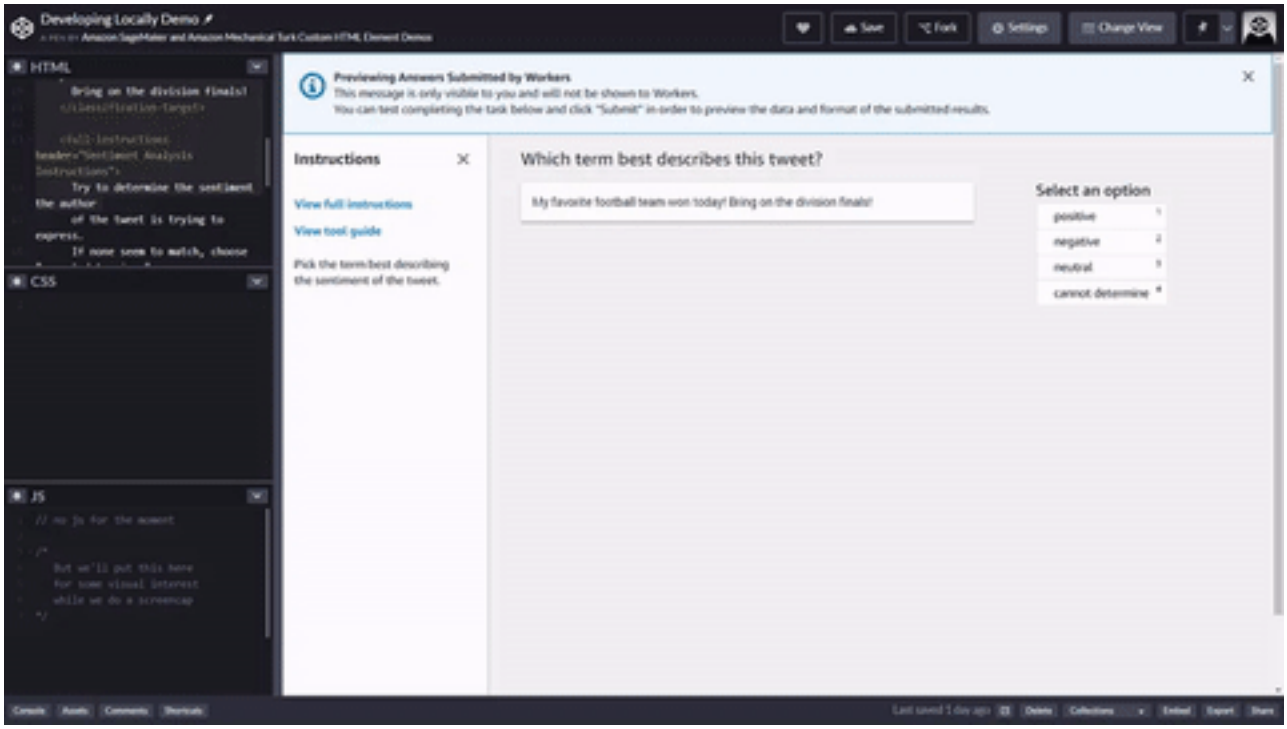
作为 `<crowd-classifier>` 元素的子元素，您必须有三个区域。

- <classification-target> - 工作人员将根据在上面 `categories` 属性中指定的选项进行分类的文本。
- <full-instructions> - 工具中的“查看完整说明”链接中提供的说明。这可以留空，但建议您提供适当的说明来获得更好的结果。
- <short-instructions> - 对于工具栏边栏中显示的任务的更简要说明。这可以留空，但建议您提供适当的说明来获得更好的结果。

此工具的简单版本如下所示。

Example 使用 **`crowd-classifier`**

您可以复制代码并将其粘贴到 Ground Truth 标记作业创建工作流程中的编辑器以预览工具，或尝试使用在 CodePen 上演示此代码。



使用 Liquid 添加自动化

我们的自定义模板系统使用 Liquid 进行自动化。它是开源内联标记语言。在 Liquid 中，单大括号和百分比符号之间的文本是指令或标签执行诸如控制流或迭代之类的操作。双大括号之间的文本是一个变量或用于输出变量值的对象。

Liquid 的最常见用途是分析来自注释前 Lambda 的数据，并提取相关变量以创建任务。注释前 Lambda (p. 331) 返回的 `taskInput` 对象将作为模板中的 `task.input` 对象。

清单的数据对象中的属性将作为 `event.dataObject` 传递到 注释前 Lambda (p. 331)。简单的传递脚本仅将该对象作为 `taskInput` 对象返回。您可以将清单中的值表示为变量，如下所示。

Example 清单数据对象

```
{
  "source": "This is a sample text for classification",
  "labels": [ "angry" , "sad" , "happy" , "inconclusive" ],
  "header": "What emotion is the speaker feeling?"
}
```

Example 使用变量的示例 HTML

```
<crowd-classifier
  name='tweetFeeling'
  categories='{{ task.input.labels | to_json }}'
  header='{{ task.input.header }}' >
<classification-target>
  {{ task.input.source }}
</classification-target>
```

请注意，在上面的 `labels` 属性中添加了“ `| to_json`”。这是一个筛选条件，用于将数组转换为数组的 JSON 表示形式。变量筛选条件将在下一节进行说明。

以下列表包括两种类型的 Liquid 标签，您可能会发现这些标签对自动化模板输入数据处理有用 如果您选择以下标签类型之一，您将被重定向到 Liquid 文档。

- 控制流：包括编程逻辑运算符`if/else`、`unless`, 和`case/when`.
- 迭代: 使您能够使用 for 循环之类的语句重复运行代码块。

  有关使用 Liquid 元素创建 for 循环的 HTML 模板的示例，请参阅翻译-审查和更正 .iqu.htmlGitHub 中。

有关更多信息和文档，请访问 Liquid 主页。

变量筛选条件

除了标准之外液体过滤器和操作，Ground Truth 提供了几个额外的筛选条件。通过在变量名称之后放置竖线 (|) 字符，然后指定筛选条件名称，以应用筛选条件。筛选条件可以采用以下形式链接起来：

Example

```
{{ <content> | <filter> | <filter> }}
```

自动转义和显式转义

默认情况下，输入将进行 HTML 转义，以防止在变量文本和 HTML 之间产生混淆。您可以显式添加 `escape` 筛选条件，以使其对于正读取正在进行转义的模板源的用户更显而易见。

escape_once

`escape_once`确保如果您已经转义代码，则不会重新转义。例如，使 & 不会变为 &;。

skip_autoescape

当您的内容要用作 HTML 时，`skip_autoescape` 很有用。例如，您可能在边界框的完整说明中有一些文本段落和一些图像。

使用**`skip_autoescape`**节俭地

模板中的最佳实践是避免使用 `skip_autoescape` 传递功能代码或标记，除非您绝对确信您对正传递的内容具有严格的控制。如果您传递用户输入，则可能会让您的工作人员面对跨站点脚本攻击。

to_json

`to_json`将您提供的内容编码为 JSON（JavaScript 对象表示法）。如果您向其提供对象，它会对其序列化。

grant_read_access

`grant_read_access`获取 S3 URI 并将其编码为 HTTPS URL（具有针对该资源的短期访问令牌）。这样，就可以向工作人员显示存储在 S3 存储桶中但原本无法公开访问的照片、音频或视频对象。

Example 的筛选条件

输入

```
auto-escape: {{ "Have you read 'James & the Giant Peach'?" }}
explicit escape: {{ "Have you read 'James & the Giant Peach'?" | escape }}
explicit escape_once: {{ "Have you read 'James &amp; the Giant Peach'?" | escape_once }}
skip_autoescape: {{ "Have you read 'James & the Giant Peach'?" | skip_autoescape }}
to_json: {{ jsObject | to_json }}
```

```
grant_read_access: {{ "s3://mybucket/myphoto.png" | grant_read_access }}
```

Example

输出

```
auto-escape: Have you read &#39;James &amp; the Giant Peach&#39;?
explicit escape: Have you read &#39;James &amp; the Giant Peach&#39;?
explicit escape_once: Have you read &#39;James &amp; the Giant Peach&#39;?
skip_autoescape: Have you read 'James & the Giant Peach'?
to_json: { "point_number": 8, "coords": [ 59, 76 ] }
grant_read_access: https://s3.amazonaws.com/mybucket/myphoto.png?<access token and other
 params>
```

Example 自动化的分类模板。

要自动执行简单文本分类示例，请将推文文本替换为变量。

文本分类模板如下，添加了自动化功能。更改/添加以粗体突出显示。

前面示例中的推文文本现已替换为一个对象。这些区域有：`entry.taskInput`使用对象`source`（或者您在注释前 Lambda 中指定的其他名称）作为文本的属性名称，它将由于介于双大括号之间直接插入在 HTML 中。

端到端演示

您可查看以下内容 end-to-end 演示包括示例 Lambda 函数：

- 演示模板：使用注释图片`crowd-bounding-box` (p. 343)
- 演示模板：使用的标记意图`crowd-classifier` (p. 346)

下一页

第 3 步：使用处理Amazon Lambda (p. 330)

第 3 步：使用处理Amazon Lambda

在这一步中，您将了解如何创建和指定两种类型的AmazonLambda创建自定义标记工作流程所需的函数：

- 注释前 Lambda：此函数在将发送到标签作业的每个数据对象发送给工作人员之前启动和预处理发送到您的标签作业。
- 注释后 Lambda：一旦工作人员提交任务，此函数将处理结果。如果您为每个数据对象指定多个 worker，则此函数可能包括合并注释的逻辑。

如果您是 Lambda 和 Ground Truth 的新用户，我们建议您使用本节中的页面，如下所示：

1. 首先，查看注释前和注释后 Lambda 函数要求 (p. 331).
2. 然后，使用页面使用所需的权限Amazon Lambda随着 Ground Truth (p. 337)了解在 Ground Truth 自定义标记作业中使用注释前和注释后 Lambda 函数的安全性和权限要求。
3. 接下来，您需要访问 Lambda 控制台或使用 Lambda 的 API 来创建函数。使用该部分为自定义标签工作流程创建 Lambda 函数 (p. 340)以了解如何创建 Lambda 函数。
4. 要了解如何测试 Lambda 函数，请参阅。测试预注释和注释后 Lambda 函数 (p. 341).
5. 在您创建预处理和后处理 Lambda 函数后，请从Lambda 函数部分出现在 Ground Truth 控制台中的自定义 HTML 代码编辑器之后。要了解如何在`CreateLabelingJob`请参阅 API 请求创建标记作业 (API) (p. 356).

有关包含注释前和注释后 Lambda 函数的示例的自定义标记工作流程，请参阅演示模板：使用注释图片crowd-bounding-box (p. 343)“文档中）。

主题

- 注释前和注释后 Lambda 函数要求 (p. 331)
- 使用所需的权限Amazon Lambda随着 Ground Truth (p. 337)
- 为自定义标签工作流程创建 Lambda 函数 (p. 340)
- 测试预注释和注释后 Lambda 函数 (p. 341)

注释前和注释后 Lambda 函数要求

使用本节可以了解发送到注释前和注释后 Lambda 函数的请求的语法，以及 Ground Truth 运行自定义标签工作流程所需的响应语法。

主题

- 注释前 Lambda (p. 331)
- 注释后 Lambda (p. 333)

注释前 Lambda

在将标记任务发送到工作线程之前，将调用注释前 Lambda 函数。

Ground Truth 向您的 Lambda 函数发送 JSON 格式的请求，以提供有关标签作业和数据对象的详细信息。下表包含预注释请求模式。下面介绍了每个参数。

Data object identified with "source-ref"

```
{
    "version": "2018-10-16",
    "labelingJobArn": <labelingJobArn>
    "dataObject" : {
        "source-ref": <s3Uri>
    }
}
```

Data object identified with "source"

```
{
    "version": "2018-10-16",
    "labelingJobArn": <labelingJobArn>
    "dataObject" : {
        "source": <string>
    }
}
```

- `version`（字符串）：这是 Ground Truth 内部使用的版本号。
- `labelingJobArn`（字符串）：这是您的贴标作业的 Amazon 资源名称或 ARN。在使用 Ground Truth API 操作时，此 ARN 可用于引用标签作业，例如`DescribeLabelingJob`.
- 这些区域有：`dataObject`（JSON 对象）：密钥包含单个 JSON 行，可以是来自您的输入清单文件或从 Amazon SNS 发送的。清单中的 JSON 行对象的大小最多可达 100 千字节，并且包含各种数据。对于非常基本的图像注释作业，`dataObject`JSON 可能只包含`source-ref`注释后，标识要注释的图像。如果数据对象（例如，一行文本）直接包含在输入清单文件中，则使用`source`. 如果您创建验证或调整作业，则此行可能包含上一个标签作业中的标签数据和元数据。

下表包括预注释请求的代码块示例。这些示例请求中的每个参数都在选项卡式表下面进行了说明。

Data object identified with "source-ref"

```
{
    "version": "2018-10-16",
    "labelingJobArn": "arn:aws:sagemaker:<aws_region>:<aws_account_number>:labeling-
job/<labeling_job_name>"
    "dataObject" : {
        "source-ref": "s3://<input-data-bucket>/<data-object-file-name>"
    }
}
```

Data object identified with "source"

```
{
    "version": "2018-10-16",
    "labelingJobArn": "arn:aws:sagemaker:<aws_region>:<aws_account_number>:labeling-
job/<labeling_job_name>"
    "dataObject" : {
        "source": "Sue purchased 10 shares of the stock on April 10th, 2020"
    }
}
```

作为回报，Ground Truth 需要格式如下的响应：

Example 预期的返回数据

```
{
    "taskInput": <json object>,
    "isHumanAnnotationRequired": <boolean> # Optional
}
```

在上面的示例中，<json object>需要包含所有自定义工作人员任务模板所需的数据。如果您正在执行一个边界框任务（其中指令始终保持相同），则它可能只是您的映像文件的 HTTP (S) 或 Amazon S3 资源。如果它是一个情感分析任务且不同的对象可能具有不同的选择，则它是对象引用（作为字符串）和选项（作为字符串数组）。

的影响**isHumanAnnotationRequired**

此值是可选的，因为它默认为true. 显式设置该值的主要使用案例是当您想要将此数据对象从人工标记范围中排除时。

如果你的清单中有混合对象，其中一些对象需要人工注释，有些不需要它，则可以包含isHumanAnnotationRequired每个数据对象中的值。您可以向预注解 Lambda 添加逻辑以动态确定对象是否需要注释，并相应地设置此布尔值。

预注解 Lambda 函数的示例

以下是基本的预注解 Lambda 函数访问中的 JSON 对象dataObject从最初的请求中获得，然后将其返回taskInput参数。

```
import json

def lambda_handler(event, context):
    return {
        "taskInput":  event['dataObject']
    }
```

假定输入清单文件使用"source-ref"为了识别数据对象，与此预注解 Lambda 相同的标签作业中使用的工作人员任务模板必须包含类似以下内容的 Liquid 元素才能提取dataObject：

```
{{ task.input.source-ref | grant_read_access }}
```

如果使用输入清单文件source为了识别数据对象，工作任务模板可以提取dataObject包含以下内容：

```
{{ task.input.source }}
```

以下预注解 Lambda 示例包括用于标识中使用的密钥的逻辑dataObject，然后使用指向那个数据对象taskObject在 Lambda 的退货声明中。

```
import json

def lambda_handler(event, context):

    # Event received
    print("Received event: " + json.dumps(event, indent=2))

    # Get source if specified
    source = event['dataObject']['source'] if "source" in event['dataObject'] else None

    # Get source-ref if specified
    source_ref = event['dataObject']['source-ref'] if "source-ref" in event['dataObject']
 else None

    # if source field present, take that otherwise take source-ref
    task_object = source if source is not None else source_ref

    # Build response object
    output = {
        "taskInput": {
            "taskObject": task_object
        },
        "humanAnnotationRequired": "true"
    }

    print(output)
    # If neither source nor source-ref specified, mark the annotation failed
    if task_object is None:
        print(" Failed to pre-process {} !".format(event["labelingJobArn"]))
        output["humanAnnotationRequired"] = "false"

    return output
```

注释后 Lambda

当所有工作人员都对数据对象进行注释或何时TaskAvailabilityLifetimeInSeconds已达到（以先到者为准），Ground Truth 将这些注释发送到注释后 Lambda。此 Lambda 通常用于合并注释 (p. 437)。

> **Tip**
>
> 要查看合并后 Lambda 函数的示例，请参阅annotation_consolidation_lambda.py中的aws-sagemaker-地面真相食谱 GitHub 存储库。

以下代码块包含注释后请求模式。下面的项目符号列表中介绍了每个参数。

```
{
    "version": "2018-10-16",
    "labelingJobArn": <string>,
    "labelCategories": [<string>],
```

```
    "labelAttributeName": <string>,
    "roleArn" : <string>,
    "payload": {
        "s3Uri": <string>
    }
 }
```

- `version`（字符串）：Ground Truth 内部使用的版本号。
- `labelingJobArn`（字符串）：贴标作业的亚马逊资源名称或 ARN。在使用 Ground Truth API 操作时，此 ARN 可用于引用标签作业，例如`DescribeLabelingJob`.
- `labelCategories`（字符串列表）：包括您在控制台中指定或包含在标签类别配置文件中的标签类别和其他属性。
- `labelAttributeName`（字符串）：标签作业的名称或创建标签作业时指定的标签属性名称。
- `roleArn`（字符串）：您在创建标记作业时指定的 IAM 执行角色的 Amazon 资源名称 (ARN)。
- `payload`（JSON 对象）：包含一个 JSON`s3Uri`key，它标识 Amazon S3 中该数据对象的注释数据的位置。下面的第二个代码块显示此注释文件的示例。

下面的代码块包含注释后请求示例。此示例请求中的每个参数都在代码块下面进行了说明。

Example 注释后的 Lambda 请求

```
{
    "version": "2018-10-16",
    "labelingJobArn": "arn:aws:sagemaker:us-west-2:111122223333:labeling-job/labeling-job-
name",
    "labelCategories": ["Ex Category1","Ex Category2", "Ex Category3"],
    "labelAttributeName": "labeling-job-attribute-name",
    "roleArn" : "arn:aws:iam::111122223333:role/role-name",
    "payload": {
        "s3Uri": "s3://DOC-EXAMPLE-BUCKET/annotations.json"
    }
 }
```

> **Note**
>
> 如果没有工作人员在数据对象上工作并且`TaskAvailabilityLifetimeInSeconds`已到达，则数据对象被标记为失败且不作为注释后 Lambda 调用的一部分包括在内。

以下代码块包含有效负载架构。这是由`s3Uri`注释后 Lambda 请求中的参数`payload`JSON 对象。例如，如果之前的代码块是注释后的 Lambda 请求，则以下注释文件位于`s3://DOC-EXAMPLE-BUCKET/annotations.json`.

下面的项目符号列表中介绍了每个参数。

Example 的注释文件

```
[
    {
        "datasetObjectId": <string>,
        "dataObject": {
            "s3Uri": <string>,
            "content": <string>
        },
        "annotations": [{
            "workerId": <string>,
            "annotationData": {
                "content": <string>,
                "s3Uri": <string>
            }
```

```
        }]
    }
]
```

- `datasetObjectId`（字符串）：标识 Ground Truth 分配给您发送到标签作业的每个数据对象的唯一 ID。
- `dataObject`（JSON 对象）：被标记的数据对象。如果数据对象包含在输入清单文件中并使用`source`key（例如，字符串），`dataObject`包括`content`key，它标识数据对象。否则，数据对象的位置（例如，链接或 S3 URI）标识`s3Uri`.
- `annotations`（JSON 对象列表）：此列表包含工作人员为此提交的每个注释的单个 JSON 对象`dataObject`. 单个 JSON 对象包含一个唯一的`workerId`它可以用来识别提交该注释的工作人员。这些区域有：`annotationData`key 包含以下内容之一：
  - `content`（字符串）：包含注释数据。
  - `s3Uri`（字符串）：包含一个用于标识注释数据位置的 S3 URI。

下表包含了您可能在负载中找到不同类型注释的内容的示例。

Named Entity Recognition Payload

```
[
    {
      "datasetObjectId": "1",
      "dataObject": {
        "content": "Sift 3 cups of flour into the bowl."
      },
      "annotations": [
        {
          "workerId": "private.us-west-2.ef7294f850a3d9d1",
          "annotationData": {
            "content": "{\"crowd-entity-annotation\":{\"entities\":[{\"endOffset
\":4,\"label\":\"verb\",\"startOffset\":0},{\"endOffset\":6,\"label\":\"number
\",\"startOffset\":5},{\"endOffset\":20,\"label\":\"object\",\"startOffset\":15},
{\"endOffset\":34,\"label\":\"object\",\"startOffset\":30}]}}"
          }
        }
      ]
    }
]
```

Semantic Segmentation Payload

```
[
    {
      "datasetObjectId": "2",
      "dataObject": {
        "s3Uri": "s3://DOC-EXAMPLE-BUCKET/gt-input-data/images/bird3.jpg"
      },
      "annotations": [
        {
          "workerId": "private.us-west-2.ab1234c5678a919d0",
          "annotationData": {
            "content": "{\"crowd-semantic-segmentation\":{\"inputImageProperties\":
{\"height\":2000,\"width\":3020},\"labelMappings\":{\"Bird\":{\"color\":\"#2ca02c\"}},
\"labeledImage\":{\"pngImageData\":\"iVBOR...\"}}}"
          }
        }
      ]
    }
  ]
```

Bounding Box Payload

```
[
    {
      "datasetObjectId": "0",
      "dataObject": {
        "s3Uri": "s3://DOC-EXAMPLE-BUCKET/gt-input-data/images/bird1.jpg"
      },
      "annotations": [
        {
          "workerId": "private.us-west-2.ab1234c5678a919d0",
          "annotationData": {
            "content": "{\"boundingBox\":{\"boundingBoxes\":[{\"height\":2052,\"label
\":\"Bird\",\"left\":583,\"top\":302,\"width\":1375}],\"inputImageProperties\":
{\"height\":2497,\"width\":3745}}}"
          }
        }
      ]
    }
 ]
```

注释后 Lambda 函数可能包含类似于以下内容的逻辑，以循环浏览并访问请求中包含的所有注释。有关完整示例，请参阅annotation_consolidation_lambda.py中的aws-sagemaker-地面真相食谱 GitHub 存储库。在此 GitHub 示例中，您必须添加自己的注释合并逻辑。

```
for i in range(len(annotations)):
    worker_id = annotations[i]["workerId"]
    annotation_content = annotations[i]['annotationData'].get('content')
    annotation_s3_uri = annotations[i]['annotationData'].get('s3uri')
    annotation = annotation_content if annotation_s3_uri is None else
 s3_client.get_object_from_s3(
        annotation_s3_uri)
    annotation_from_single_worker = json.loads(annotation)

    print("{} Received Annotations from worker [{}] is [{}]"
            .format(log_prefix, worker_id, annotation_from_single_worker))
```

Tip

当您对数据运行合并算法时，可以使用Amazon用于存储结果的数据库服务，也可以将处理的结果传回 Ground Truth。您返回到 Ground Truth 的数据存储在整合注释清单中，它们存储在为标记作业进行配置期间为输出指定的 S3 存储桶中。

作为回报，Ground Truth 需要格式如下的响应：

Example 预期的返回数据

```
[
   {
        "datasetObjectId": <string>,
        "consolidatedAnnotation": {
            "content": {
                "<labelattributename>": {
                    # ... label content
                }
            }
        }
    },
   {
        "datasetObjectId": <string>,
```

```
        "consolidatedAnnotation": {
            "content": {
                "<labelattributename>": {
                    # ... label content
                }
            }
        }
    }
    .
    .
    .
]
```

此时，您要发送到 S3 存储桶的所有数据，但datasetObjectId，在content对象。

当你在中返回注释时content，这将导致作业输出清单中的条目，如下所示：

Example 输出清单中的标签格式

```
{  "source-ref"/"source" : "<s3uri or content>",
   "<labelAttributeName>": {
        # ... label content from you
    },
   "<labelAttributeName>-metadata": { # This will be added by Ground Truth
        "job_name": <labelingJobName>,
        "type": "groundTruth/custom",
        "human-annotated": "yes",
        "creation_date": <date> # Timestamp of when received from Post-labeling Lambda
    }
}
```

由于自定义模板及其收集的数据的潜在复杂的性质，Ground Truth 不提供对数据的进一步处理。

使用所需的权限Amazon Lambda随着 Ground Truth

您可能需要配置以下部分或全部内容才能创建和使用Amazon Lambda使用 Ground Truth。

- 您需要授予 IAM 角色或用户（统称为 IAM 实体）使用创建注释前和注释后 Lambda 函数的权限Amazon Lambda，并在创建标签作业时选择它们。
- 在配置标记作业时指定的 IAM 执行角色需要调用注释前和注释后 Lambda 函数的权限。
- 注释后 Lambda 函数可能需要获得访问 Amazon S3 的权限。

使用以下部分可了解如何创建 IAM 实体并授予上述权限。

主题

- 授予创建和选择Amazon Lambda函数 (p. 337)
- 授予 IAM 执行角色调用权限Amazon Lambda函数 (p. 338)
- 授予注释后 Lambda 访问注释的权限 (p. 339)

授予创建和选择Amazon Lambda函数

如果您不需要精确权限即可开发注释前和注释后 Lambda 函数，则可以附加Amazon管理的策略AWSLambda_FullAccess为 IAM 用户或角色提供。此策略授予使用所有 Lambda 功能的广泛权限，以及在其他功能中执行操作的权限AmazonLambda 与之交互的服务。

要为安全敏感型用例创建更精细的策略，请参阅文档Lambda 的基于身份的 IAM 策略在到Amazon Lambda有关如何创建适合您的使用案例的 IAM 策略的开发人员指南。

使用 Lambda 控制台的策略

如果您想授予 IAM 实体使用 Lambda 控制台的权限，请参阅使用 Lambda 控制台中的Amazon Lambda开发人员指南 的第一个版本。

此外，如果您希望用户能够使用Amazon Serverless Application Repository在 Lambda 控制台中，您必须指定*<aws-region>*要在哪里部署函数（这应该是相同的）Amazon用于创建标记作业的区域），并将以下策略添加到 IAM 角色中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "serverlessrepo:ListApplicationVersions",
                "serverlessrepo:GetApplication",
                "serverlessrepo:CreateCloudFormationTemplate"
            ],
            "Resource": "arn:aws:serverlessrepo:<aws-region>:838997950401:applications/aws-
sagemaker-ground-truth-recipe"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "serverlessrepo:SearchApplications",
            "Resource": "*"
        }
    ]
}
```

在 Ground Truth 控制台中查看 Lambda 函数的策略

要授予 IAM 实体在用户创建自定义标签作业时在 Ground Truth 控制台中查看 Lambda 函数的权限，该实体必须具有中所述的权限授予 IAM 使用权限 SageMaker Ground Truth 控制 (p. 448)，包括部分中描述的权限自定义标记流程权限 (p. 451).

授予 IAM 执行角色调用权限Amazon Lambda函数

如果添加 IAM 托管策略AmazonSageMakerGroundTruthExecution对于用于创建标签作业的 IAM 执行角色，此角色有权使用函数名称中的以下字符串之一列出和调用 Lambda 函数：`GtRecipe`、`SageMaker`、`Sagemaker`、`sagemaker`，或者`LabelingFunction`.

如果预注释或注释后 Lambda 函数名称不包含上一段中的一个术语，或者您需要比`AmazonSageMakerGroundTruthExecution`托管策略，您可以添加类似于以下内容的策略，以授予执行角色调用注释前和注释后功能的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action":
                "lambda:InvokeFunction",
            "Resource": [
                "arn:aws:lambda:<region>:<account-id>:function:<pre-annotation-lambda-
name>",
                "arn:aws:lambda:<region>:<account-id>:function:<post-annotation-lambda-
name>"
            ]
        }
```

```
    ]
}
```

授予注释后 Lambda 访问注释的权限

如中所述注释后 Lambda (p. 333)，注释后 Lambda 请求包括 Amazon S3 中注释数据的位置。此位置由`s3Uri`中的字符串`payload`对象。要在注释进来时处理它们，即使对于简单的传递函数，您也需要为注释后分配必要的权限。Lambda 执行角色从 Amazon S3 读取文件。

有很多方法可以配置 Lambda 以访问 Amazon S3 中的注释数据。两种常见的方法是：

- 允许 Lambda 执行角色代入 SageMaker 中确定的执行角色`roleArn`在注释后的 Lambda 请求中。该 SageMaker 执行角色是用于创建标签任务的角色，可以访问存储注释数据的 Amazon S3 输出存储桶。
- 授予 Lambda 执行角色的权限以直接访问 Amazon S3 输出存储桶。

使用以下部分可了解如何配置这些选项。

授予 Lambda 假设权限 SageMaker 执行角色

允许 Lambda 函数假设 SageMaker 执行角色，您必须将策略附加到 Lambda 函数的执行角色，然后修改 SageMaker 允许 Lambda 代入它的执行角色。

1. 附加以下 IAM 策略转到 Lambda 函数的执行角色以代入 SageMaker 中确定的执行角色`Resource`. Replace*222222222222*用一个Amazon账户 ID. Replace*sm-execution-role*使用的角色的名称为。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::222222222222:role/sm-execution-role"
    }
}
```

2. 修改信任策略的 SageMaker 执行角色包括以下内容`Statement`. Replace*222222222222*用一个Amazon账户 ID. Replace*my-lambda-execution-role*注释后角色的名称。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::222222222222:role/my-lambda-execution-role"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

授予 Lambda 执行角色访问 S3 的权限

您可以将类似于以下内容的策略添加到注释后 Lambda 函数执行角色，以授予 S3 读取权限。Replace*DOC-###*使用您在创建标记作业时指定的输出存储桶的名称。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

要向 Lambda 控制台中的 Lambda 执行角色添加 S3 读取权限，请使用以下过程。

将 S3 读取权限添加到注释后 Lambda：

1. 打开函数页在 Lambda 控制台中。
2. 选择注释后函数的名称。
3. 选择 Configuration (配置)，然后选择 Permissions (权限)。
4. SELECTRole name (角色名称)并且该角色的摘要页面将在 IAM 控制台中打开，并在新选项卡中打开。
5. Select附加策略.
6. 请执行下列操作之一：
   - 搜索并选择**AmazonS3ReadOnlyAccess**授予函数读取账户中所有存储桶和对象的权限。
   - 如果您需要更精细的权限，请选择创建策略然后使用上一节中的策略示例来创建策略。请注意，创建策略后，必须返回执行角色摘要页面。
7. 如果您使用AmazonS3ReadOnlyAccess选择托管策略，选择附加策略.

   如果您创建了新策略，请导航回 Lambda 执行角色摘要页面并附加刚创建的策略。

为自定义标签工作流程创建 Lambda 函数

您可以使用 Lambda 控制台创建 Lambda 函数。Amazon CLI，或者AmazonSDK 采用您所选择的受支持编程语言。使用Amazon Lambda开发者指南，了解有关以下每个选项的更多信息：

- 要了解如何使用控制台创建 Lambda 函数，请参阅。使用控制台创建 Lambda 函数.
- 要了解如何使用创建 Lambda 函数Amazon CLI，请参阅使用AmazonLambdaAmazonCommand Line Interface.
- 选择目录中的相关部分，以了解有关使用所选语言使用 Lambda 的更多信息。例如，选择使用 Python要了解有关将 Lambda 与Amazon SDK for Python (Boto3).

Ground Truth 通过Amazon Serverless Application Repository(SAR)食谱. 使用以下过程在 Lambda 控制台中选择 Ground Truth 配方。

使用 Ground Truth SAR 配方创建注释前和注释后 Lambda 函数：

1. 打开函数页在 Lambda 控制台上。
2. Select创建函数.
3. Select浏览无服务器应用存储库.
4. 在搜索文本框中，输入aws-sagemaker-地面真相食谱然后选择该应用程序。
5. Select部署. 部署该应用程序可能需要几分钟时间。

   应用程序部署后，两个函数会显示在函数Lambda 控制台的部分：serverlessrepo-aws-sagema-GtRecipePreHumanTaskFunc-*<id>*和serverlessrepo-aws-sagema-GtRecipeAnnotationConsol-*<id>*.
6. 选择这些函数之一，然后将自定义逻辑添加到代码部分。
7. 完成更改后，选择部署来部署它们。

测试预注释和注释后 Lambda 函数

您可以在 Lambda 控制台中测试预注释并发布注释 Lambda 函数。如果您是 Lambda 的新用户，您可以了解如何测试，或者调用，您的 Lambda 函数在控制台中使用创建 Lambda 函数教程中的控制台Amazon Lambda开发人员指南 的第一个版本。

您可以使用本页面上的部分了解如何测试通过Amazon Serverless Application Repository(SAR)。

主题

- 先决条件 (p. 341)
- 测试预注解 Lambda 函数 (p. 342)
- 测试注释后 Lambda 函数 (p. 342)

先决条件

您必须执行以下操作才能使用本页上描述的测试。

- 您需要访问 Lambda 控制台，并且需要创建和调用 Lambda 函数的权限。要了解如何设置这些权限，请参阅。授予创建和选择Amazon Lambda函数 (p. 337).
- 如果您尚未部署 Ground Truth SAR 配方，请使用中的步骤为自定义标签工作流程创建 Lambda 函数 (p. 340)要做到这一点。
- 要测试注释后 Lambda 函数，您必须在 Amazon S3 中有一个带有示例注释数据的数据文件。对于简单的测试，您可以将以下代码复制并粘贴到文件中，并将其另存为`sample-annotations.json`和将此文件上载到 Amazon S3. 请注意此文件的 S3 URI-您需要此信息来配置注释后的 Lambda 测试。

```
[{"datasetObjectId":"0","dataObject":{"content":"To train a machine learning model,
 you need a large, high-quality, labeled dataset. Ground Truth helps you build
 high-quality training datasets for your machine learning models."},"annotations":
[{"workerId":"private.us-west-2.0123456789","annotationData":{"content":"{\"crowd-
entity-annotation\":{\"entities\":[{\"endOffset\":8,\"label\":\"verb\",\"startOffset
\":3},{\"endOffset\":27,\"label\":\"adjective\",\"startOffset\":11},{\"endOffset\":33,
\"label\":\"object\",\"startOffset\":28},{\"endOffset\":51,\"label\":\"adjective\",
\"startOffset\":46},{\"endOffset\":65,\"label\":\"adjective\",\"startOffset\":53},
{\"endOffset\":74,\"label\":\"adjective\",\"startOffset\":67},{\"endOffset\":82,
\"label\":\"adjective\",\"startOffset\":75},{\"endOffset\":102,\"label\":\"verb
\",\"startOffset\":97},{\"endOffset\":112,\"label\":\"verb\",\"startOffset\":107},
{\"endOffset\":125,\"label\":\"adjective\",\"startOffset\":113},{\"endOffset\":134,
\"label\":\"adjective\",\"startOffset\":126},{\"endOffset\":143,\"label\":\"object
\",\"startOffset\":135},{\"endOffset\":169,\"label\":\"adjective\",\"startOffset
\":153},{\"endOffset\":176,\"label\":\"object\",\"startOffset\":170}]}}"}}]},
{"datasetObjectId":"1","dataObject":{"content":"Sift 3 cups of flour into the
 bowl."},"annotations":[{"workerId":"private.us-west-2.0123456789","annotationData":
{"content":"{\"crowd-entity-annotation\":{\"entities\":[{\"endOffset\":4,\"label
\":\"verb\",\"startOffset\":0},{\"endOffset\":6,\"label\":\"number\",\"startOffset
\":5},{\"endOffset\":20,\"label\":\"object\",\"startOffset\":15},{\"endOffset\":34,
\"label\":\"object\",\"startOffset\":30}]}}"}}]},{"datasetObjectId":"2","dataObject":
{"content":"Jen purchased 10 shares of the stock on Janurary 1st, 2020."},"annotations":
[{"workerId":"private.us-west-2.0123456789","annotationData":{"content":"{\"crowd-
entity-annotation\":{\"entities\":[{\"endOffset\":3,\"label\":\"person\",\"startOffset
\":0},{\"endOffset\":13,\"label\":\"verb\",\"startOffset\":4},{\"endOffset\":16,
\"label\":\"number\",\"startOffset\":14},{\"endOffset\":58,\"label\":\"date\",
\"startOffset\":40}]}}"}}]},{"datasetObjectId":"3","dataObject":{"content":"The
 narrative was interesting, however the character development was weak."},"annotations":
[{"workerId":"private.us-west-2.0123456789","annotationData":{"content":"{\"crowd-entity-
annotation\":{\"entities\":[{\"endOffset\":29,\"label\":\"adjective\",\"startOffset
\":18},{\"endOffset\":73,\"label\":\"adjective\",\"startOffset\":69}]}}"}}]}]
```

- 您必须使用中的指示授予注释后 Lambda 访问注释的权限 (p. 339)授予注释后 Lambda 函数的执行角色以承担 SageMaker 用于创建标签作业的执行角色。注释后 Lambda 函数使用 SageMaker 执行角色来访问注释数据文件，`sample-annotations.json`，在 S3 中。

测试预注解 Lambda 函数

使用以下过程测试在部署 Ground Truth 时创建的预注解 Lambda 函数Amazon Serverless Application Repository(SAR) 食谱。

测试 Ground Truth SAR 配方预注解 Lambda 函数

1. 打开函数页在 Lambda 控制台中。
2. 选择从 Ground Truth SAR 配方中部署的预注功能。此函数的名称类似于`serverlessrepo-aws-sagema-GtRecipePreHumanTaskFunc-<id>`.
3. 在源代码部分中，选择旁边的箭头测试.
4. Select配置测试事件.
5. 保留创建新测试事件已选择选项。
6. UNDER活动模板，请选择SageMaker Ground Truth.
7. 为您的测试提供Event name (事件名称).
8. 选择 Create (创建)。
9. 选择旁边的箭头测试再次，你应该看到你创建的测试已被选中，该测试由事件名称用点表示。如果未选中，请选择它。
10. Select测试运行测试。

运行测试后，你可以看到执行结果. 在函数日志，您会看到类似以下内容的响应：

```
START RequestId: cd117d38-8365-4e1a-bffb-0dcd631a878f Version: $LATEST
Received event: {
  "version": "2018-10-16",
  "labelingJobArn": "arn:aws:sagemaker:us-east-2:123456789012:labeling-job/example-job",
  "dataObject": {
    "source-ref": "s3://sagemakerexample/object_to_annotate.jpg"
  }
}
{'taskInput': {'taskObject': 's3://sagemakerexample/object_to_annotate.jpg'},
 'isHumanAnnotationRequired': 'true'}
END RequestId: cd117d38-8365-4e1a-bffb-0dcd631a878f
REPORT RequestId: cd117d38-8365-4e1a-bffb-0dcd631a878f Duration: 0.42 ms Billed Duration: 1
 ms Memory Size: 128 MB Max Memory Used: 43 MB
```

在此响应中，我们可以看到 Lambda 函数的输出与所需的预注释响应语法匹配：

```
{'taskInput': {'taskObject': 's3://sagemakerexample/object_to_annotate.jpg'},
 'isHumanAnnotationRequired': 'true'}
```

测试注释后 Lambda 函数

使用以下过程测试在部署 Ground Truth 时创建的注释后 Lambda 函数Amazon Serverless Application Repository(SAR) 食谱。

测试 Ground Truth SAR 配方注释后 Lambda

1. 打开函数页在 Lambda 控制台中。
2. 选择从 Ground Truth SAR 配方中部署的注释后功能。此函数的名称类似于`serverlessrepo-aws-sagema-GtRecipeAnnotationConsol-<id>`.
3. 在源代码部分中，选择旁边的箭头测试.
4. Select配置测试事件.
5. 保留创建新测试事件已选择选项。

6. UNDER活动模板，请选择SageMaker Ground Truth 注释后.
7. 为您的测试提供Event name (事件名称).
8. 按如下所示修改所提供的模板代码：

   - 在中替换 Amazon 资源名称 (ARN)`roleArn`使用的 ARN SageMaker 用于创建标签作业的执行角色。
   - 在中替换 S3 URIs`s3Uri`使用的 URI`sample-annotations.json`您添加到 Amazon S3 的文件。

   在进行这些修改后，测试应类似于以下内容：

```
{
  "version": "2018-10-16",
  "labelingJobArn": "arn:aws:sagemaker:us-east-2:123456789012:labeling-job/example-
job",
  "labelAttributeName": "example-attribute",
  "roleArn": "arn:aws:iam::222222222222:role/sm-execution-role",
  "payload": {
    "s3Uri": "s3://your-bucket/sample-annotations.json"
  }
}
```

9. 选择 Create (创建)。
10. 选择旁边的箭头测试再次，你应该看到你创建的测试已被选中，该测试由事件名称用点表示。如果未选中，请选择它。
11. SELECT测试运行测试。

在您运行测试后，您会看到`-- Consolidated Output --`中的部分函数日志，其中包含包含的所有注释的列表。`sample-annotations.json`.

演示模板：使用注释图片`crowd-bounding-box`

当您在 Amazon 中选择使用自定义模板作为您的任务类型时 SageMaker Ground Truth 控制台，你到达自定义标签任务面板. 在该面板中，您可以从多个基本模板中进行选择。模板表示一些最常见的任务，并在您创建自定义标记任务的模板时提供样本。如果你没有使用控制台，或者作为额外的追索手段，请参阅Amazon SageMaker Ground Truth 示例任务 UI有关各种标记作业任务类型的演示模板存储库。

该演示适用于 BoundingBox 模板。该演示还适用于在执行任务前后处理数据所需的 Amazon Lambda 函数。在上面的 Github 存储库中，查找可以使用的模板Amazon Lambda函数，寻找`{{ task.input.<property name> }}`在模板中。

主题

- 入门边界框自定义模板 (p. 343)
- 您自己的边界框自定义模板 (p. 344)
- 清单文件 (p. 344)
- 注释前 Lambda 函数 (p. 344)
- 注释后 Lambda 函数 (p. 345)
- 标记作业的输出 (p. 346)

入门边界框自定义模板

这是所提供的入门边界框模板。

此自定义模板使用 Liquid 模板语言，双大括号之间的每个项都是一个变量。注释前Amazon Lambda函数应该提供一个名为`taskInput`并且该对象的属性可以通过以下方式访问`{{ task.input.<property name> }}`在模板中。

您自己的边界框自定义模板

例如，假设你有大量的动物照片，在这些照片中你知道以前的图像分类工作中的图像中的动物种类。现在你想在它周围画一个边框。

在入门示例中，有三个变量：`taskObject`、`header`, 和`labels`.

其中每个变量都表示在边界框的不同部分中。

- `taskObject`是要注释的照片的 HTTP (S) URL 或 S3 URI。增加了| `grant_read_access`是一种过滤器，它将 S3 URI 转换为对该资源的短暂访问权限的 HTTPS URL。如果你使用的是 HTTP (S) URL，则不需要它。
- `header`是要标记照片上方的文字，比如 “在照片中的鸟周围画一个盒子”。
- `labels`是一个数组，表示为`['item1', 'item2', ...]`. 这些标签可以由工作人员分配给他们绘制的不同盒子。您可以有一个或多个标签。

每个变量名称均来自针对预注释 Lambda 的响应中的 JSON 对象。上面的名称仅仅是建议使用的名称，它们使用任何对您有意义的变量名称，并可以提升团队间的代码可读性。

> 仅在必要时才使用变量
>
> 如果某个字段将不会更改，您可以从模板中删除该变量并将其替换为该文本，否则您必须重复将该文本作为清单内每个对象中的值或将其编码到注释前 Lambda 函数。

Example ：最终自定义边界框模板

为简单起见，此模板将具有一个变量、一个标签和非常基本的说明。假设你的清单在每个数据对象中都有一个 “动物” 属性，那么该值可以在模板的两个部分中重复使用。

注意重复使用`{{ task.input.animal }}`在整个模板中。如果你的清单上的所有动物名称都以大写字母开头，你可以使用`{{ task.input.animal | downcase }}`，在需要显示为小写字母的句子中加入 Liquid 的内置过滤器之一。

清单文件

您的清单文件应提供您在模板中使用的变量值。您可以对注释前 Lambda 中的清单数据进行某种转换，但如果您不需要，则可以保持较低的错误风险，您的 Lambda 将运行得更快。下面是模板的一个示例清单文件。

```
{"source-ref": "<S3 image URI>", "animal": "horse"}
{"source-ref": "<S3 image URI>", "animal" : "bird"}
{"source-ref": "<S3 image URI>", "animal" : "dog"}
{"source-ref": "<S3 image URI>", "animal" : "cat"}
```

注释前 Lambda 函数

作为职位设置的一部分，提供一个Amazon Lambda函数可以调用此函数处理您的清单条目并将它们传递到模板引擎。

> 为 Lambda 函数命名
>
> 在命名函数时，最佳做法是使用以下四个字符串之一作为函数名的一部分：`SageMaker`、`Sagemaker`、`sagemaker`，或者`LabelingFunction`. 这同时适用于注释前和注释后功能。

如果您使用控制台，如果有Amazon您的账户拥有的 Lambda 函数将提供满足命名要求的函数的下拉列表供您选择一个函数。

在这个非常基本的例子中，你只需传递清单中的信息，而不对其进行任何额外的处理。此示例预注释后函数是为 Python 3.7 编写的。

```
import json

def lambda_handler(event, context):
    return {
        "taskInput": event['dataObject']
    }
```

来自清单的 JSON 对象将作为 `event` 对象的子对象提供。内部的属性`taskInput`对象将可用作模板的变量，因此只需设置`taskInput`到`event['dataObject']`将把清单对象中的所有值传递到模板，而无需单独复制它们。如果您想向模板发送更多值，则可以将它们添加到`taskInput`对象。

注释后 Lambda 函数

作为职位设置的一部分，提供一个Amazon Lambda当工作线程完成任务时，可以调用此函数处理表单数据。这可以很简单，也可以很复杂。如果您想在进来时进行应答整合和评分，则可以用您选择的评分和/或整合算法。如果您想要存储原始数据以进行脱机处理，则这是一个选项。

提供对注释后 Lambda 的权限

注释数据将位于一个文件中，该文件由 `payload` 对象中的 `s3Uri` 字符串指定。要在出现注释时处理这些注释，即使对于简单的传递函数，您也需要分配`S3ReadOnly`访问您的 Lambda 以便它可以读取注释文件。
在用于创建 Lambda 的控制台中，滚动至 Execution role (执行角色) 面板。Select从一个或多个模板创建新角色. 为角色提供一个名称。从Policy templates下拉菜单，选择Amazon S3 对象只读权限.
保存 Lambda，将保存并选择该角色。

下面是 Python 2.7 中的示例。

```
import json
import boto3
from urlparse import urlparse

def lambda_handler(event, context):
    consolidated_labels = []

    parsed_url = urlparse(event['payload']['s3Uri']);
    s3 = boto3.client('s3')
    textFile = s3.get_object(Bucket = parsed_url.netloc, Key = parsed_url.path[1:])
    filecont = textFile['Body'].read()
    annotations = json.loads(filecont);

    for dataset in annotations:
        for annotation in dataset['annotations']:
            new_annotation = json.loads(annotation['annotationData']['content'])
            label = {
                'datasetObjectId': dataset['datasetObjectId'],
                'consolidatedAnnotation' : {
                'content': {
                    event['labelAttributeName']: {
                        'workerId': annotation['workerId'],
                        'boxesInfo': new_annotation,
                        'imageSource': dataset['dataObject']
                        }
                    }
                }
            }
            consolidated_labels.append(label)

    return consolidated_labels
```

注释后 Lambda 通常将在事件对象中收到成批的任务结果。这种批处理将是 Lambda 应迭代的 `payload` 对象。您发回的内容将是满足API 合同 (p. 330).

标记作业的输出

您将在您指定的目标 S3 存储桶中以您的标记作业命名的文件夹中找到作业的输出。它将位于名为 `manifests` 的子文件夹中。

对于边界框任务，您在输出清单找到的输出将有点类似于下面的演示。该示例已清理以供打印。实际输出将是每条记录一行。

Example ：输出清单中的 JSON

```
{
  "source-ref":"<URL>",
  "<label attribute name>":
    {
       "workerId":"<URL>",
       "imageSource":"<image URL>",
       "boxesInfo":"{\"boundingBox\":{\"boundingBoxes\":[{\"height\":878, \"label\":\"bird
\", \"left\":208, \"top\":6, \"width\":809}], \"inputImageProperties\":{\"height\":924,
 \"width\":1280}}}"},
  "<label attribute name>-metadata":
    {
      "type":"groundTruth/custom",
      "job_name":"<Labeling job name>",
      "human-annotated":"yes"
    },
  "animal" : "bird"
}
```

请注意额外的`animal`将原始清单中的属性传递给输出清单，与`source-ref`以及标记数据。输入清单中的任何属性，无论它们是否在模板中使用，都将传递给输出清单。

演示模板：使用的标记意图`crowd-classifier`

如果您选择自定义模板，您将到达自定义标签任务面板. 在那里，您可以从多个入门模板中进行选择，这些模板代表了一些较常见的 这些模板提供了构建您的自定义标记任务模板的起点。

在此演示中，您可使用目的检测模板，它使用`crowd-classifier (p. 495)`元素，以及Amazon Lambda在执行任务前后处理数据所需的函数。

主题

- 入门目的检测自定义模板 (p. 346)
- 你的意图检测自定义模板 (p. 347)
- 注释前 Lambda 函数 (p. 348)
- 注释后 Lambda 函数 (p. 348)
- 你的标签作业输出 (p. 349)

入门目的检测自定义模板

这是作为起点提供的目的检测模板。

此自定义模板使用 Liquid 模板语言，双大括号之间的每个项都是一个变量。注释前AmazonLambda 函数应该提供一个名为`taskInput`并且该对象的属性可以通过以下方式访问`{{ task.input.<property name> }}`在模板中。

你的意图检测自定义模板

在入门模板中，有两个变量：crowd-classifier 元素中的 task.input.labels 属性，用于打开标记，以及 classification-target 区域的内容中的 task.input.utterance。

除非你需要提供不同语法的不同标签集合，否则避免使用变量并仅使用文本将节省处理时间并减少出错的可能性。本演示中使用的模板将删除该变量，但变量和过滤器类似to_json有更详细的解释，请参阅crowd-bounding-box示范文章。

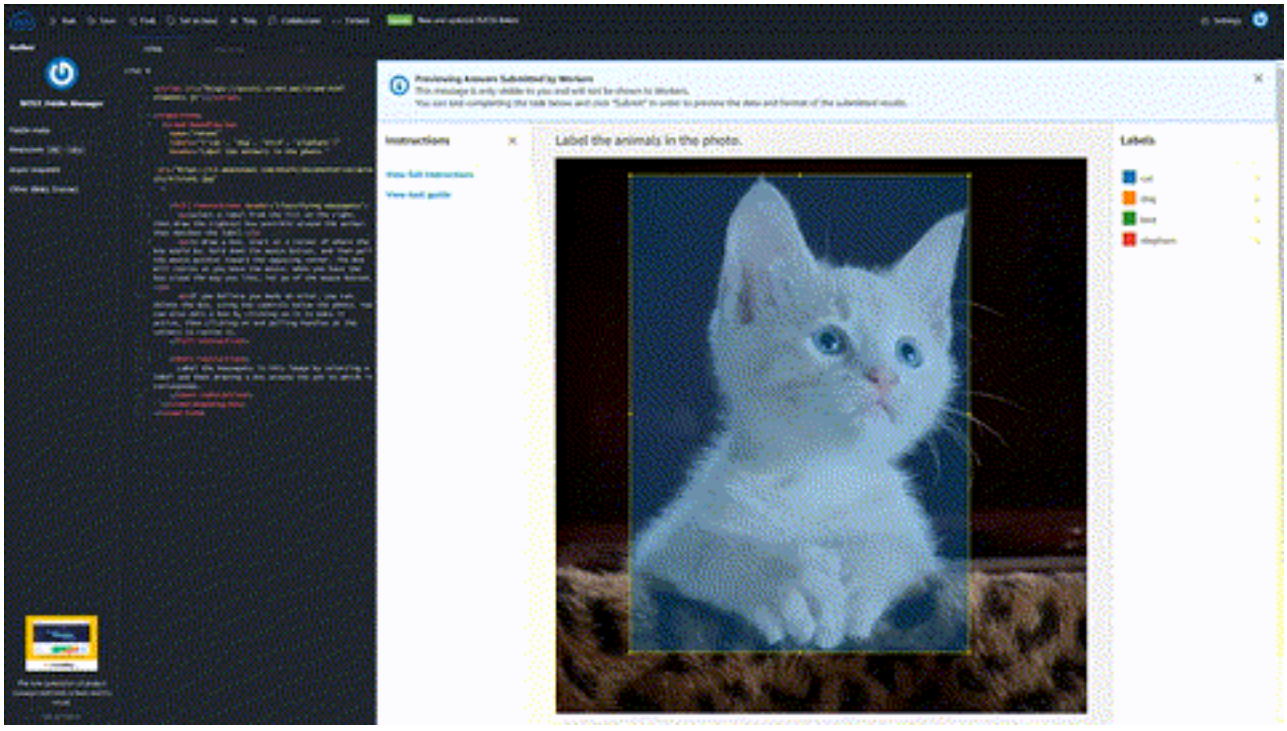
为元素设置样式

这些自定义元素中有两个有时被忽略的部分是<full-instructions>和<short-instructions>地区。好的说明会产生良好的结果。

在包含这些区域的元素中，<short-instructions>自动显示在工作人员屏幕左侧的 “说明” 窗格中。这些区域有：<full-instructions>将从该窗格顶部附近的 “查看完整说明” 链接链接进行链接。单击链接打开一个模式窗格，其中包含更详细的说明。

你不仅可以使用 HTML、CSS 和 JavaScript 在这些章节中，如果您相信可以提供一套强有力的说明和示例，帮助员工以更快的速度和准确地完成任务，我们鼓励您了解。

Example 用 jsfiddle 尝试样本



尝试示例<crowd-classifier>任务. 该示例由 jsfiddle 呈现，因此所有模板变量都被硬编码值替换。单击 “查看完整说明” 链接以查看一组具有扩展 CSS 样式的示例。你可以分叉项目来试验你自己对 CSS 的更改、添加示例图像或添加扩展 JavaScript 功能性。

Example ：最终自定义目的检测模板

这使用示例<crowd-classifier>任务，但是有一个变量用于<classification-target>. 如果您试图在一系列不同的标签作业中保持一致的 CSS 设计，则可以使用<link rel...>元素与在任何其他 HTML 文档中所做的方式一样。

Example ：您的清单文件

如果您正在为类似这样的文本分类任务手动准备清单文件，则可以下列方式格式化数据。

```
{"source": "Roses are red"}
{"source": "Violets are Blue"}
{"source": "Ground Truth is the best"}
{"source": "And so are you"}
```

这有别于用于“演示模板：使用注释图片crowd-bounding-box (p. 343)”演示的清单文件，在后者中，source-ref 用作属性名而非 source。使用source-ref将 S3 URI 指定为必须转换为 HTTP 的映像或其他文件。否则，应将 source 视为含上面的文本字符串来使用。

注释前 Lambda 函数

作为职位设置的一部分，提供一个Amazon Lambda这可以调用它来处理您的清单条目并将它们传递到模板引擎。

此 Lambda 函数必须包含以下四个字符串之一作为函数名的一部分：
SageMaker、Sagemaker、sagemaker 或 LabelingFunction。

这同时适用于注释前和注释后 Lambda。

当您使用控制台时，如果您有一个归您的账户拥有的 Lambda，则将提供满足命名要求的函数的下拉列表供您选择一个函数。

在这个非常基本的示例中，您只有一个变量，它主要是一个直通函数。下面是使用 Python 3.7 的标记前 Lambda 的一个示例。

```
import json

def lambda_handler(event, context):
    return {
        "taskInput":  event['dataObject']
    }
```

这些区域有：dataObject的财产event包含清单中数据对象的属性。

在这个演示中，这是一个简单的传递，您只是将它作为taskInput值。如果将具有这些值的属性添加到event['dataObject']对象，则它们可作为格式为 {{ task.input.*<property name>* }} 的 Liquid 变量用于您的 HTML 模板。

注释后 Lambda 函数

作为作业设置的一部分，请提供 Lambda 函数的 ARN，当工作线程完成任务时，可以调用该函数处理表单数据。这可以很简单，也可以很复杂。如果您想在数据进来时进行应答整合和评分，则可以用您选择的评分或整合算法。如果您想要存储原始数据以进行脱机处理，则这是一个选项。

为注释后 Lambda 函数设置权限

注释数据将位于一个文件中，该文件由 payload 对象中的 s3Uri 字符串指定。要在出现注释时处理这些注释，即使对于简单的传递函数，您也需要分配S3ReadOnly访问您的 Lambda 以便它可以读取注释文件。
在用于创建 Lambda 的控制台中，滚动至 Execution role (执行角色) 面板。Select从一个或多个模板创建新角色. 为角色提供一个名称。从Policy templates下拉菜单，选择Amazon S3 对象只读权限.
保存 Lambda，将保存并选择该角色。

以下示例适用于 Python 3.7。

```
import json
import boto3
from urllib.parse import urlparse

def lambda_handler(event, context):
    consolidated_labels = []

    parsed_url = urlparse(event['payload']['s3Uri']);
    s3 = boto3.client('s3')
    textFile = s3.get_object(Bucket = parsed_url.netloc, Key = parsed_url.path[1:])
    filecont = textFile['Body'].read()
    annotations = json.loads(filecont);

    for dataset in annotations:
        for annotation in dataset['annotations']:
            new_annotation = json.loads(annotation['annotationData']['content'])
            label = {
                'datasetObjectId': dataset['datasetObjectId'],
                'consolidatedAnnotation' : {
                'content': {
                    event['labelAttributeName']: {
                        'workerId': annotation['workerId'],
                        'result': new_annotation,
                        'labeledContent': dataset['dataObject']
                        }
                    }
                }
            }
            consolidated_labels.append(label)

    return consolidated_labels
```

你的标签作业输出

注释后 Lambda 通常将在事件对象中收到成批的任务结果。这种批处理将是 Lambda 应迭代的 `payload` 对象。

您将在您指定的目标 S3 存储桶中以您的标记作业命名的文件夹中找到作业的输出。它将位于名为 `manifests` 的子文件夹中。

对于目的检测任务，输出清单中的输出将有点类似于下面的演示。这个示例已进行清理并加宽间距，以便于阅读。对于机器阅读，实际输出将更加压缩。

Example ：输出清单中的 JSON

```
[
  {
    "datasetObjectId":"<Number representing item's place in the manifest>",
     "consolidatedAnnotation":
     {
       "content":
       {
         "<name of labeling job>":
         {
           "workerId":"private.us-east-1.XXXXXXXXXXXXXXXXXXXXXX",
           "result":
           {
             "intent":
             {
                 "label":"<label chosen by worker>"
             }
           },
           "labeledContent":
```

```
            {
              "content":"<text content that was labeled>"
            }
          }
        }
      }
    },
  "datasetObjectId":"<Number representing item's place in the manifest>",
     "consolidatedAnnotation":
     {
       "content":
       {
         "<name of labeling job>":
         {
           "workerId":"private.us-east-1.6UDLPKQZHYWJQSCA4MBJBB7FWE",
           "result":
           {
             "intent":
             {
                 "label": "<label chosen by worker>"
             }
           },
           "labeledContent":
           {
             "content": "<text content that was labeled>"
           }
         }
       }
     }
   },
     ...
     ...
     ...
]
```

这将有助于您创建和使用自己的自定义模板。

通过 API 实现的自定义工作流

当您创建自定义 UI 模板（步骤 2）并处理 Lambda 函数（步骤 3）后，应该将模板放在 Amazon S3 存储桶中，其文件名格式为：`<FileName>.liquid.html`.

使用 `CreateLabelingJob` 操作配置您的任务。您将使用存储在 S3 上 *`<filename>`*`.liquid.html` 文件中的自定义模板 (第 2 步：创建自定义工作人员任务模板 (p. 326)) 的位置作为 `HumanTaskConfig` 对象内 `UiConfig` 对象中 `UiTemplateS3Uri` 字段的值。

对于Amazon中所述的 Lambda 任务第 3 步：使用处理Amazon Lambda (p. 330)，注释后任务的 ARN 将用作`AnnotationConsolidationLambdaArn`字段，并且预注释任务将用作`PreHumanTaskLambdaArn`.

创建标记作业

您可以在 Amazon SageMaker 控制台中创建标记作业，以及使用Amazon使用你的首选语言运行的 SDK`CreateLabelingJob`. 在创建标记作业后，您可以使用跟踪工作人员指标（对于私有人力）和标记作业状态。CloudWatch.

在创建标签作业之前，建议您查看以下页面（如果适用）：

- 您可以使用控制台中的自动数据设置，或者在控制台或使用时使用输入清单文件来指定我们的输入数据`CreateLabelingJob`API。有关自动化数据设置，请参阅自动数据设置 (p. 379). 要了解如何创建输入清单文件，请参阅。使用输入清单文件 (p. 378).
- 查看标签作业输入数据配额：输入数据配额 (p. 385).

在选择任务类型后，请使用该页面上的主题以了解如何创建标记作业。

如果您是 Ground Truth 的新用户，我们建议您先查看中的演示。开始使用 (p. 206).

> Important
>
> Ground Truth 要求所有包含标签作业输入映像数据的 S3 存储桶都必须附加 CORS 策略。要了解更多信息，请参阅“CORS 许可要求 (p. 446)”。

主题

- 内置任务类型 (p. 351)
- 创建说明页 (p. 351)
- 创建标记作业（控制台） (p. 353)
- 创建标记作业 (API) (p. 356)
- 创建直播标记 Job (p. 360)
- 创建具有标签类别和框架属性的标记类别配置文件 (p. 364)

内置任务类型

Amazon SageMaker Ground Truth 有几种内置任务类型。Ground Truth 为内置任务类型提供了一个工作人员任务模板。此外，有些内置任务类型支持自动数据标记 (p. 438). 以下主题介绍了每种内置任务类型，并在控制台中演示由 Ground Truth 提供的工作人员任务模板。要了解如何使用这些任务类型之一在控制台中创建标签作业，请选择任务类型页面。

标签图片	Text Label	为视频和视频帧添加标签	标记 3D 点云
• Bounding Box (p. 211) • 图像分类（单个标签） (p. 218) • 图像分类（多标签） (p. 221) • 图像语义分割 (p. 215) • 验证和调整标签 (p. 318)	• 命名实体识别 (p. 224) • 文本分类（单标签） (p. 228) • 文本分类（多标签） (p. 230)	• 视频分类 (p. 233) • 视频帧物体检测 (p. 236) • 视频帧对象跟踪 (p. 240)	• 3D 点云对象检测 (p. 264) • 3D 点云对象跟踪 (p. 270) • 3D 点云语义分割 (p. 279)

> Note
>
> 每种视频帧和 3D 点云任务类型都有调整用于验证和调整之前标签作业中的标签的任务类型。选择上面的视频帧或 3D 点云任务类型页面，了解如何调整使用该任务类型创建的标签。

创建说明页

创建有关标记作业的自定义说明，以提高工作人员完成其任务的准确性。您可以修改控制台中提供的默认说明，也可以创建自己的说明。这些说明将显示在工作人员在其中完成标记任务的页面上。

说明分为两种：

- 简短说明— 工作人员完成其任务所在网页上显示的说明。这些说明应提供简单的参考内容，向工作人员显示标记对象的正确方法。

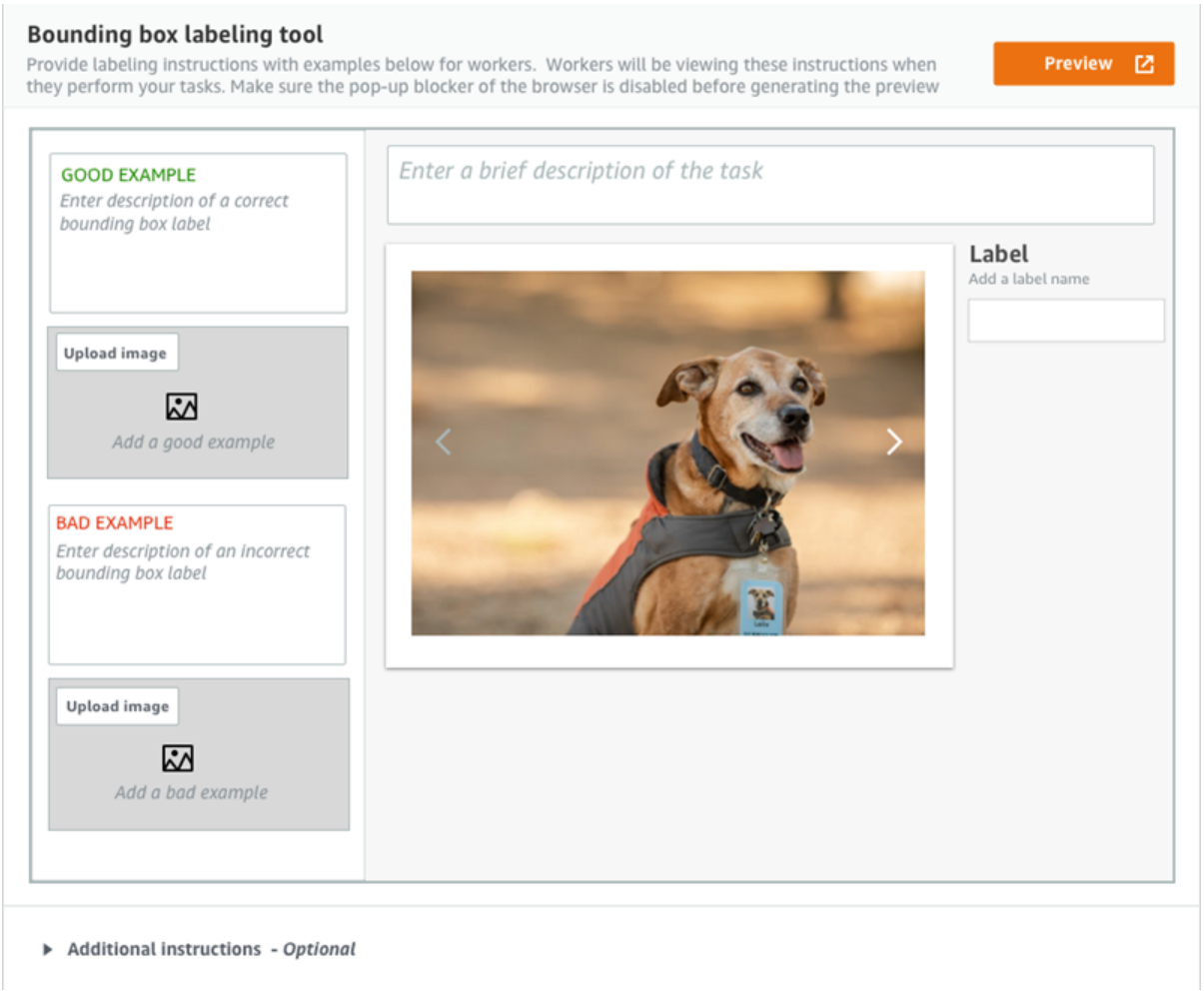
- 完整说明— 显示在对话框中的说明，该说明覆盖工作人员完成其任务所在页面的说明。如果要完成的任务使用多个示例来显示边缘情况和其他难以标记对象的情况，我们建议您提供详细说明。

当您创建标记作业时在控制台中创建说明。从任务的现有说明开始，然后使用编辑器对其进行修改以适合您的标记作业。

> Note
>
> 创建标记作业后，它将自动启动，并且您无法修改工作人员说明。如果您需要更改工作人员说明，请在创建新作业之前停止创建的标记作业、克隆它并修改工作人员说明。
> 您可以在控制台中克隆标记作业，方法是选择标记作业，然后选择Clone中的操作菜单。
> 要使用 Amazon SageMaker API 或首选 Amazon SageMaker 开发工具包克隆标记作业，请向`CreateLabelingJob`修改工作人员说明后，使用与原始作业相同的规范进行操作。

简短说明

简短说明出现在工作人员用于标记数据对象的同一个网页上。例如，下面是边界框任务的编辑页面。简短说明面板位于左侧。



请记住，工作人员只会花数秒时间查看简短说明。工作人员必须能够快速扫描并理解您的信息。在所有情况下，了解说明所花的时间都应少于完成任务所需的时间。请记住以下几点：

- 您的说明应简单明确。
- 图片比文字效果更好。为任务创建工作人员可以即刻理解的简单插图。

- 如果您必须使用文字，请使用简明的示例。
- 您的简短说明比完整说明更重要。

Amazon SageMaker Ground Truth 控制台提供了一个编辑器，以便您可以创建简短说明。用任务说明替换占位符文本和图像。通过选择预览以预览工作人员的任务页面。预览将在新窗口中打开，请确保关闭弹出窗口拦截功能，以便窗口能够显示。

完整说明

您可以在一个对话框（覆盖工作人员标记数据对象所在的页面）中为您的工作人员提供额外的说明。使用完整说明解释更复杂的任务，并向工作人员显示标记边缘案例或其他困难对象的正确方法。

您可以在 Ground Truth 控制台中使用编辑器创建完整说明。对于快速说明，请记住以下事项：

- 工作人员在前几次完成任务时需要详细说明。他们必须具有的任何信息都应出现在快速说明中。
- 图片比文字更重要。
- 文本应简明。
- 完整说明应为简短说明提供补充。请勿重复简短说明中出现的信息。

Ground Truth 控制台提供了一个编辑器，以便您可以创建完整说明。用任务说明替换占位符文本和图像。通过选择预览以预览完整说明页面。预览将在新窗口中打开，请确保关闭弹出窗口拦截功能，以便窗口能够显示。

将示例图像添加到您的说明

图像为您的工作人员提供了有用的示例。要将可公开访问的图像添加到您的说明，请执行以下操作：

- 将光标放在说明编辑器中的所需图像位置。
- 单击编辑器工具栏中的图像图标。
- 输入图像的 URL。

如果无法公开访问 Amazon S3 中的说明图像：

- 作为图像 URL，请输入：`{{ 'https://s3.amazonaws.com/`*`your-bucket-name`*`/`*`image-file-name`*`' | grant_read_access }}`。
- 这会提供一个附加了短期一次性访问代码的图像 URL，以便工作人员的浏览器可以显示该图像。将在说明编辑器中显示损坏的图像图标，但在预览该工具时，将在提供的预览中显示该图像。

创建标记作业（控制台）

您可以使用 Amazon SageMaker 控制台为所有 Ground Truth 内置任务类型和自定义标记工作流创建标记作业。对于内置任务类型，我们建议您将此页与任务类型的页面. 每个任务类型页面都包含使用该任务类型创建标签作业的具体详细信息。

您需要提供以下内容以在 SageMaker 控制台中创建标记作业：

- Amazon S3 中的输入清单文件。您可以将输入数据集放在 Amazon S3 中，并使用 Ground Truth 控制台自动生成清单文件（不支持 3D 点云标签作业）。

  或者，您可以手动创建输入清单文件。要了解如何操作，请参阅输入数据 (p. 378)。
- 用于存储输出数据的 Amazon S3 存储桶。
- 具有在 Amazon S3 中访问资源的权限并附加了 SageMaker 执行策略的 IAM 角色。对于常规解决方案，您可以将托管策略 AmazonSageMakerFullAccess 附加到 IAM 角色，并包括：`sagemaker`和句点 (.)。

有关更精细的策略，请参阅the section called “IAM 权限” (p. 447).

3D 点云任务类型还有其他安全注意事项。了解更多。

- 工作团队。您可以从 Amazon Mechanical Turk 员工、供应商或自己的私人员组成的员工中创建一个工作团队。要了解更多信息，请参阅创建和管理人力 (p. 463).

  您无法将 Mechanical Turk 人力用于 3D 点云或视频帧标记作业。
- 如果使用自定义标记工作流，您必须在 Amazon S3 中保存工作人员任务模板，并为该模板提供 Amazon S3 URI。有关更多信息，请参阅 第 2 步：创建自定义工作人员任务模板 (p. 326)。
- （可选）Amazon KMS如果希望 SageMaker 使用您自己的标记作业输出进行加密，则需要提供密钥 ARN。Amazon KMS加密密钥而不是默认的 Amazon S3 服务密钥。
- （可选）用于标记作业的数据集的现有标签。如果您希望工作人员调整、批准或拒绝标签，请使用该选项。
- 如果要创建调整或验证标签任务，则必须在 Amazon S3 中有一个输出清单文件，其中包含要调整或验证的标签。此选项仅适用于边界框和语义分割图像标注作业以及 3D 点云和视频帧标注作业。建议您使用上的说明。验证和调整标签 (p. 318)以创建验证或调整标签作业。

  Important

  您的工作组、输入清单文件、输出存储桶以及 Amazon S3 中的其他资源必须位于同一位置Amazon 您用来创建标记作业的区域。

在使用 SageMaker 控制台创建标记作业时，您可以将工作人员说明和标签添加到 Ground Truth 提供的工作人员 UI 中。在控制台中创建标记作业时，您可以预览工作人员 UI 并与之交互。你也可以在你的上面看到工作人员 UI 的预览内置任务类型页.

创建标记作业（控制台）

1. 登录位于的 SageMaker 控制台。https://console.aws.amazon.com/sagemaker/.
2. 在左侧导航窗格中，选择 Labeling jobs (标记作业)。
3. 在 Labeling jobs (标记作业) 页面上，选择 Create labeling job (创建标记作业)。
4. 对于 Job name (作业名称)，请输入标记作业的名称。
5. （可选）如果要使用一个键以标识标签，请选择 I want to specify a label attribute name different from the labeling job name (我希望指定与标记作业名称不同的标签属性名称)。如果未选择该选项，将使用在上一步中指定的标记作业名称在输出清单文件中标识标签。
6. 选择要设置的数据设置，以便在输入数据集和 Ground Truth 之间建立连接。
   - 适用于自动化数据设置：
     - 按照中的说明进行操作自动数据设置 (p. 379)用于图像、文本和视频剪辑标签作业。
     - 按照中的说明进行操作自动视频帧输入数据设置 (p. 411)对于视频帧标记作业。
   - 适用于手动数据设置：
     - 适用于输入数据集位置，请在 Amazon S3 中提供输入清单文件所在的位置。例如，如果输入清单文件 manifest.json 位于示例存储桶输入s3: //示例存储桶 /manifest.json.
     - 适用于Output 数据集位置在 Amazon S3 中提供希望 Ground Truth 存储标记作业输出数据的位置。
7. 适用于IAM 角色，选择一个现有的 IAM 角色或创建一个 IAM 角色，该角色有权访问 Amazon S3 中的资源以写入到上面指定的输出 Amazon S3 存储桶并附加了 SageMaker 执行策略。
8. （可选）对于其他配置，您可以指定希望工作人员标记数据集的哪些部分，以及是否希望 SageMaker 使用Amazon KMS加密钥。要对输出数据进行加密，您必须具有必需的Amazon KMS附加到您在上一步中提供的 IAM 角色的权限。有关更多信息，请参阅 the section called “IAM 权限” (p. 447)。
9. 在任务类型位置在下部分下任务类别中，使用下拉列表选择任务类别。
10. 在 Task selection (任务选择) 中，选择任务类型。

11. （可选）为标记作业提供标签，以便以后在控制台中轻松找到该作业。
12. 选择 Next（下一步）。
13. 在 Workers (工作人员) 部分中，选择您要使用的人力类型。有关人力选项的更多详细信息，请参阅创建和管理人力 (p. 463)。
14. （可选）在选择人力后，指定任务超时。这是为工作人员处理任务提供的最长时间。

    对于 3D 点云注释任务，默认任务超时为 3 天。文本和图像分类以及标签验证标记作业的默认超时为 5 分钟。所有其他标记作业的默认超时为 60 分钟。
15. （可选）对于边界框、语义分割、视频帧和 3D 点云任务类型，您可以选择显示现有标签如果要显示输入数据集的标签以供工作人员验证或调整。

    对于边界框和语义分割标记作业，这将创建调整标记作业。

    对于 3D 点云和视频帧标记作业：

    - Select调整以创建调整标签作业。选择此选项时，您可以添加新标签，但无法从上一个作业中删除或编辑现有标签。或者，您可以选择要员工编辑的标签类别属性和框架属性。要将某个属性设置为可编辑，请选中复选框。允许工作人员编辑此属性对于该属性。

      或者，您可以添加新的标签类别和框架属性。
    - Select验证以创建调整标签作业。如果选择此选项，您无法在上一作业中添加、修改或删除现有的标签。或者，您可以选择要员工编辑的标签类别属性和框架属性。要将某个属性设置为可编辑，请选中复选框。允许工作人员编辑此属性对于该属性。

      我们建议您可以向希望工作人员验证的标签添加新的标签类别属性，或者添加一个或多个帧属性以让工作人员提供有关整个帧的信息。

    有关更多信息，请参阅 验证和调整标签 (p. 318)。
16. 配置你的员工用户界面：

    - 如果您使用内置任务类型，指定工作人员的说明和标签。
      - 对于图像分类和文本分类（单个和多标签），您必须至少指定两个标签类别。对于所有其他内置任务类型，您必须至少指定一个标签类别。
      - （可选）如果要创建 3D 点云或视频帧标记作业，则可以指定标记类别属性（不支持 3D 点云语义分割）和帧属性。您可以将标签类别属性分配给一个或多个标签。帧属性将显示在每个点云或视频帧工作人员标签上。要了解更多信息，请参阅工作人员用户界面 (UI) (p. 287)对于 3D 点云和工作人员用户界面 (UI) (p. 246)对于视频帧。
      - （可选）添加其他说明以帮助你的员工完成你的任务。
    - 如果要创建自定义标签工作流程，则必须：
      - 输入自定义模板在代码框中。可以使用 HTML、Liquid 模板语言和我们预先构建的 Web 组件的组合来创建自定义模板。或者，您可以从下拉菜单中选择基本模板以开始使用。
      - 指定注释前和注释后 lambda 函数。要了解如何创建这些函数，请参阅第 3 步：使用处理Amazon Lambda (p. 330).
17. （可选）你可以选择请参阅预览以预览工作人员说明、标签以及与工作人员 UI 交互。在生成预览之前，请确保浏览器的弹出窗口阻止程序处于禁用状态。
18. 选择创建。

在成功创建标记作业后，您将重定向到 Labeling jobs (标记作业) 页面。您刚创建的标记作业的状态为正在进行中. 在工作人员完成任务时，将逐渐更新该状态。在成功完成所有任务后，该状态将变为 Completed (已完成)。

如果在创建标记作业时出现问题，其状态变为已失败.

要查看有关作业的更多详细信息，请选择标记作业名称。

后续步骤

在您的贴标作业状态更改为之后已完成，您可以在创建该标记作业时指定的 Amazon S3 存储桶中查看输出数据。有关输出数据格式的更多信息，请参阅输出数据 (p. 414)。

创建标记作业 (API)

要使用 Amazon SageMaker API 创建标记作业，您可以使用CreateLabelingJoboperation. 有关为内置任务类型创建标记作业的具体说明，请参阅该任务类型页面。要了解如何创建流式标记作业（一个永久运行的标记作业），请参阅。创建直播标记 Job (p. 360).

使用CreateLabelingJob操作，您需要满足以下条件：

- 工作人员任务模板 (UiTemplateS3Uri) 或人工任务 UI ARN (HumanTaskUiArn) 在 Amazon S3 中。
  - 对于 3D 点云作业、视频对象检测和跟踪作业以及 NER 作业，请使用中列出的 ARN。HumanTaskUiArn对于你的任务类型。
  - 如果您使用的是 3D 点云任务以外的内置任务类型，则可以将工作人员说明添加到其中一个预构建的模板中，并将此模板（使用 .html 或 .liquid 扩展名）保存在 S3 存储桶中。在任务类型页面上查找预构建模板。
  - 如果使用自定义标记工作流，您可以创建一个自定义模板，并将该模板保存在 S3 存储桶中。要了解如何构建自定义工作人员模板，请参阅第 2 步：创建自定义工作人员任务模板 (p. 326)。有关可用于自定义模板的自定义 HTML 元素，请参阅Crowd HTML 元素参考 (p. 485)。有关各种标记任务的演示模板的存储库，请参阅 Amazon SageMaker Ground Truth 示例任务 UI。
- 一个输入清单文件，用于指定 Amazon S3 中的输入数据。在中指定输入清单文件的位置ManifestS3Uri. 有关创建输入清单的信息，请参阅输入数据 (p. 378)。如果您创建流式标签作业，则这是可选的。要了解如何创建流式标记作业，请参阅。创建直播标记 Job (p. 360).
- 用于存储输出数据的 Amazon S3 存储桶。您可以在中指定此存储桶，还可以选择在中添加前S3OutputPath.
- 标签类别配置文件。每个标签类别名称必须唯一。要在 Amazon S3 中使用LabelCategoryConfigS3Uri参数。此文件的格式和标签类别取决于您使用的任务类型：
  - 对于图像分类和文本分类（单个和多标签），您必须至少指定两个标签类别。对于所有其他任务类型，所需的标签类别的最小数量为 1。
  - 对于指定实体识别任务，必须在此文件中提供工作人员说明。请参阅在标签类别配置文件中提供工作人员说明 (p. 227)有关详细信息和示例。
  - 对于 3D 点云和视频帧任务类型，请在中使用格式。创建具有标签类别和框架属性的标记类别配置文件 (p. 364).
  - 对于所有其他内置任务类型和自定义任务，标签类别配置文件必须是以下格式的 JSON 文件。通过将 label_1、label_2、...、label_n 替换为标签类别，确定要使用的标签。

```
{
    "document-version": "2018-11-28"
    "labels": [
        {"label": "label_1"},
        {"label": "label_2"},
        ...
        {"label": "label_n"}
    ]
}
```

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon Identity and Access Management一个 (IAM) 角色AmazonSageMakerGroundTruthExecution附加托管 IAM 策略，并具有访问 S3 存储桶的权限。在中指定此角色RoleArn. 要了解此策略的更多信息，请参阅。使用 IAM 托管策略与 Ground Truth (p. 448). 如果您需要更精细的权限，请参阅the section called “IAM 权限” (p. 447)。

  如果您的输入或输出存储桶名称不包含 sagemaker，则可以将类似于以下内容的策略附加到传递给 CreateLabelingJob 操作的角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::my_input_bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::my_output_bucket/*"
            ]
        }
    ]
}
```

- 用于处理输入和输出数据的注释前和注释后（或注释-合并）Amazon Lambda 函数 Amazon 资源名称 (ARN)。
  - Lambda 函数在每个函数中都预定义Amazon区域用于内置任务类型。要查找您所在区域的注释前 Lambda ARN，请参阅。PreHumanTaskLambdaArn. 要查找您所在区域的注释整合 Lambda ARN，请参阅。AnnotationConsolidationLambdaArn.
  - 对于自定义标记工作流，您必须提供自定义注释前和注释后 Lambda ARN。要了解如何创建这些 Lambda 函数，请参阅。第 3 步：使用处理Amazon Lambda (p. 330).
- 您在中指定的工作团队 ARN`WorkteamArn`. 当您订阅供应商员工或创建私有工作团队时，您会收到工作团队 ARN。如果您要为视频帧或点云任务类型创建标记作业，您不能使用Amazon Mechanical Turk人力。对于所有其他任务类型，要使用 Mechanical Turk 员工，请使用以下 ARN。Replace*region*使用Amazon您用来创建标签作业的区域。

  `arn:aws:sagemaker:region:394669845002:workteam/public-crowd/default`

  如果您将Amazon Mechanical Turk劳动力，使用`ContentClassifiers`中的参数`DataAttributes`的`InputConfig`要声明您的内容不含个人身份信息和成人内容。

  Ground Truth要求如果您使用 Mechanical Turk 工作人员，则输入数据不含个人身份信息 (PII)。如果您使用 Mechanical Turk 并且没有指定输入数据不含 PII，请使用`FreeOfPersonallyIdentifiableInformation`标记，你的标签作业将失败。使用`FreeOfAdultContent`标记，以声明您的输入数据不含成人内容。 如果您的任务包含成人内容，SageMaker 可能会限制可以查看您的任务的 Amazon Mechanical Turk 人力。

  要了解有关工作团队和人力的更多信息，请参阅创建和管理人力 (p. 463)。
- 如果您使用 Mechanical Turk 劳动力队伍，则必须指定工作人员在中执行单项任务时需要支付的费用**`PublicWorkforceTaskPrice`**.
- 要配置任务，必须使用以下方法提供任务描述和标题`TaskDescription`和**`TaskTitle`**分别。或者，您可以提供时间限制，以控制工作人员必须在单个任务上工作的时间（**`TaskTimeLimitInSeconds`**）以及任务在工作人员门户中保留多长时间，可供工作人员使用 (`TaskAvailabilityLifetimeInSeconds`)。
- （可选）对于某些任务类型，您可以让多个工作人员标记单个数据对象（为 `NumberOfHumanWorkersPerDataObject` 参数输入大于 1 的数字）。有关注释合并的更多信息，请参阅合并注释 (p. 437)。
- （可选）要创建自动数据标记作业，请在中指定一个 ARN。LabelingJobAlgorithmSpecificationArn在`LabelingJobAlgorithmsConfig`.

此 ARN 标识自动化数据标记作业中使用的算法。与此 ARN 关联的任务类型必须与`PreHumanTaskLambdaArn`和`AnnotationConsolidationLambdaArn`你指定。以下任务类型支持自动数据标记：图像分类、边界框、语义分割和文本分类。自动数据标记允许的最小对象数量为 1,250，我们强烈建议至少提供 5,000 个对象。要了解自动数据标记作业的更多信息，请参阅自动数据标记 (p. 438).

- （可选）你可以提供`StoppingConditions`如果符合条件，那就导致标签作业停止。您可以使用停止条件来控制贴标作业的成本。

示例

以下代码示例演示如何使用创建标记作业：`CreateLabelingJob`. 有关其他示例，我们建议您使用一个Ground Truth 标记作业SageMaker 笔记本实例的 SageMaker 示例部分中的 Jupyter 笔记本。要了解如何使用 SageMaker 示例中的笔记本示例，请参阅示例笔记本 (p. 158). 你也可以在 GitHub 上看到这些示例笔记本SageMaker 示例存储库.

Amazon SDK for Python (Boto3)

以下是一个示例：AmazonPython 开发工具包 (Boto3) 请求为美国东部（弗吉尼亚北部）区域使用私有人力创建内置任务类型的标记作业。RESPECT All`######`使用你的标签作业资源和规格。

```
response = client.create_labeling_job(
    LabelingJobName="example-labeling-job",
    LabelAttributeName="label",
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': "s3://bucket/path/manifest-with-input-data.json"
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                "FreeOfPersonallyIdentifiableInformation"|"FreeOfAdultContent",
            ]
        }
    },
    OutputConfig={
        'S3OutputPath': "s3://bucket/path/file-to-store-output-data",
        'KmsKeyId': "string"
    },
    RoleArn="arn:aws:iam::*:role/*",
    LabelCategoryConfigS3Uri="s3://bucket/path/label-categories.json",
    StoppingConditions={
        'MaxHumanLabeledObjectCount': 123,
        'MaxPercentageOfInputDatasetLabeled': 123
    },
    HumanTaskConfig={
        'WorkteamArn': "arn:aws:sagemaker:region:*:workteam/private-crowd/*",
        'UiConfig': {
            'UiTemplateS3Uri': "s3://bucket/path/custom-worker-task-template.html"
        },
        'PreHumanTaskLambdaArn': "arn:aws:lambda:us-
east-1:432418664414:function:PRE-tasktype",
        'TaskKeywords': [
            "Images",
            "Classification",
            "Multi-label"
        ],
        'TaskTitle': "Multi-label image classification task",
        'TaskDescription': "Select all labels that apply to the images shown",
        'NumberOfHumanWorkersPerDataObject': 1,
        'TaskTimeLimitInSeconds': 3600,
        'TaskAvailabilityLifetimeInSeconds': 21600,
```

```
        'MaxConcurrentTaskCount': 1000,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': "arn:aws:lambda:us-
east-1:432418664414:function:ACS-"
        },
    Tags=[
        {
            'Key': "string",
            'Value': "string"
        },
    ]
)
```

Amazon CLI

以下是一个示例：Amazon要求 CLI 请求在美国东部（弗吉尼亚北部）区域为内置任务类型创建标记作业。Amazon Mechanical Turk 人力. 有关更多信息，请参阅 。开始人类循环中的Amazon CLI命令参考. RESPECT All######使用你的标签作业资源和规格。

```
$ aws --region us-east-1 sagemaker create-labeling-job \
--labeling-job-name "example-labeling-job" \
--label-attribute-name "label" \
--role-arn "arn:aws:iam::account-id:role/role-name" \
--input-config '{
        "DataAttributes": {
            "ContentClassifiers": [
                "FreeOfPersonallyIdentifiableInformation",
                "FreeOfAdultContent"
            ]
        },
        "DataSource": {
            "S3DataSource": {
                "ManifestS3Uri": "s3://bucket/path/manifest-with-input-data.json"
            }
        }
    }' \
--output-config '{
        "KmsKeyId": "",
        "S3OutputPath": "s3://bucket/path/file-to-store-output-data"
    }' \
--human-task-config '{
        "AnnotationConsolidationConfig": {
            "AnnotationConsolidationLambdaArn": "arn:aws:lambda:us-
east-1:432418664414:function:ACS-"
        },
        "TaskAvailabilityLifetimeInSeconds": 21600,
        "TaskTimeLimitInSeconds": 3600,
        "NumberOfHumanWorkersPerDataObject": 1,
        "PreHumanTaskLambdaArn":  "arn:aws:lambda:us-
east-1:432418664414:function:PRE-tasktype",
        "WorkteamArn": "arn:aws:sagemaker:us-east-1:394669845002:workteam/public-crowd/
default",
        "PublicWorkforceTaskPrice": {
            "AmountInUsd": {
                "Dollars": 0,
                "TenthFractionsOfACent": 6,
                "Cents": 3
            }
        },
        "TaskDescription": "Select all labels that apply to the images shown",
        "MaxConcurrentTaskCount": 1000,
        "TaskTitle": "Multi-label image classification task",,
        "TaskKeywords": [
            "Images",
```

```
            "Classification",
            "Multi-label"
        ],
        "UiConfig": {
            "UiTemplateS3Uri": "s3://bucket/path/custom-worker-task-template.html"
        }
    }'
```

有关此操作的更多信息，请参阅 CreateLabelingJob。有关如何使用其他特定于语言的开发工具包的信息，请参阅。另请参阅中的CreateLabelingJobs主题。

创建直播标记 Job

通过流式标签作业，您可以将单个数据对象实时发送到永久运行的流式标签作业。要创建流式标记作业，您必须创建 Amazon SNS。输入主题并在中指定此主题CreateLabelingJob参数InputConfig的SnsDataSource. 或者，您还可以创建 Amazon SNS。输出主题然后在中指定OutputConfig如果要实时接收标记数据。

> **Important**
>
> 如果您是 Ground Truth 流式标记作业的新用户，建议您查阅。Ground Truth 直播标签职位 (p. 381)在创建直播标签作业之前。

使用以下部分创建您需要并可用于创建流式标签作业的资源：

- 按照中的步骤，了解如何使用 Ground Truth 流式传输标签作业所需的权限创建 SNS 主题创建 Amazon SNS 输入和输出主题 (p. 361). 您的 SNS 主题必须在同一时间内创建Amazon区域作为标记作业。
- 请参阅为终端节点订阅您的 Amazon SNS 输出主题 (p. 362)了解如何设置终端节点，以便在每次标记任务完成时在指定终端节点上接收标签任务输出数据。
- 要了解如何配置 Amazon S3 存储桶以向您的 Amazon SNS 输入主题发送通知，请参阅设置 Amazon S3 存储桶事件通知 (p. 362).
- 或者，在标签作业开始后立即将要标记的数据对象添加到输入清单中。有关更多信息，请参阅 创建清单文件（可选）(p. 363)。
- 创建标签作业还需要其他资源，例如 IAM 角色、Amazon S3 存储桶、工作人员任务模板和标签类别。有关创建标签作业的 Ground Truth 文档中对这些内容进行了描述。有关更多信息，请参阅 创建标记作业 (p. 350)。

> **Important**
>
> 在创建标记作业时，您必须提供 IAM 执行角色。将附加到Amazon管理的策略AmazonSageMakerGroundTruthExecution以确保其拥有执行标签作业所需的权限。

当您提交创建流式标签作业的请求时，标签作业的状态为Initializing. 标签作业处于活动状态后，状态将更改为InProgress. 不要向标签作业发送新的数据对象，也不要在标签作业处于Initializing状态。一旦状态变为InProgress，您可以使用 Amazon SNS 和 Amazon S3 配置开始发送新的数据对象。

主题

- 创建 Amazon SNS 输入和输出主题 (p. 361)
- 设置 Amazon S3 存储桶事件通知 (p. 362)
- 创建清单文件（可选）(p. 363)
- 例如：使用 SageMaker API 创建流式标签 Job (p. 363)
- 停止直播标签 Job (p. 364)

创建 Amazon SNS 输入和输出主题

要创建流式标记作业，您需要创建 Amazon SNS 输入。或者，您可以提供 Amazon SNS 输出主题。

在您创建要在直播标记作业中使用的 Amazon SNS 主题时，请记下 Amazon Resource Name (ARN) 该主题。ARN 将是参数的输入值`SnsTopicArn`在`InputConfig`和`OutputConfig`在创建标记作业时。

创建输入主题

您的输入主题用于向 Ground Truth 发送新的数据对象。要创建输入主题，请按照中的说明操作。创建 Amazon SNS 主题在 Amazon Simple Notification Service 开发人员指南中。

记下你的输入主题 ARN 并将其用作`CreateLabelingJob`参数`SnsTopicArn`在`InputConfig`.

创建输出主题

如果您提供输出主题，则用于在标记数据对象时发送通知。在创建主题时，您可以选择添加密密钥。使用该选项可添加Amazon Key Management Service客户管理的主题密钥，以便在标签作业的输出数据发布到输出主题之前对其进行加密。

要创建输出主题，请按照中的说明操作。创建 Amazon SNS 主题在 Amazon Simple Notification Service 开发人员指南中。

如果添加加密，则必须向该主题附加其他权限。请参阅向输出主题添加加密（可选）(p. 361)。了解更多信息。

> **Important**
>
> 要在控制台中创建主题时向输出主题添加客户托管密钥，请勿使用（默认）别名 /aws/SNS选项。选择您创建的客户托管密钥。

记下你的输入主题 ARN 并在你的`CreateLabelingJob`参数中的请求`SnsTopicArn`在`OutputConfig`.

向输出主题添加加密（可选）

要加密发布到输出主题的消息，您需要提供Amazon KMS客户管理的主题密钥。修改以下策略并将其添加到客户托管密钥中，以便在将输出数据发布到输出主题之前授予 Ground Truth 对输出数据加密的权限。

Replace*<account_id>*包含您用于创建主题的账户的 ID。要了解如何查找您的Amazon请参阅账户 ID查找您的Amazon账户 ID.

```
{
    "Id": "key-console-policy",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<account_id>:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow access for Key Administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<account_id>:role/Admin"
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
```

```
                "kms:List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion"
            ],
            "Resource": "*"
        }
    ]
}
```

此外，您必须修改以下策略并将其添加到用于创建标签作业的执行角色（`RoleArn`）。

Replace*<account_id>*包含您用于创建主题的账户的 ID。Replace*<region>*使用Amazon您用于创建标记作业的区域。Replace*<key_id>*使用客户管理的密钥 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "sid1",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "arn:aws:kms:<region>:<account_id>:key/<key_id>"
    }
  ]
}
```

有关创建和保护密钥的更多信息，请参阅。创建密钥和使用密钥策略中的Amazon Key Management Service 开发人员指南 的第一个版本。

为终端节点订阅您的 Amazon SNS 输出主题

当工作人员完成 Ground Truth 流式标签作业的标签任务时，Ground Truth 会使用输出主题将输出数据发布到您指定的一个或多个终端节点。要在工作人员完成标记任务时接收通知，您必须让一个终端节点订阅 Amazon SNS 输出主题。

要了解如何将终端节点添加到输出主题，请参阅。订阅 Amazon SNS 主题中的Amazon Simple Notification Service 开发人员指.

要了解发布到这些端点的输出数据格式的详细信息，请参阅。输出数据 (p. 414).

> **Important**
>
> 如果您没有为终端节点订阅 Amazon SNS 输出主题，则在标记了新数据对象时，您将不会收到通知。

设置 Amazon S3 存储桶事件通知

您可以使用 Amazon S3 控制台、API 和特定于语言的 API 向 Amazon S3 存储桶添加事件通知。AmazonSDK，或者Amazon Command Line Interface. 设置此事件以将通知发送到您使用指定的同一 Amazon SNS 输入主题`SnsTopicArn`在`InputConfig`在创建标记作业时。请勿使用您为其指定的相同 Amazon S3 位置设置事件通知`S3OutputPath`在`OutputConfig`— 这样做可能会导致 Ground Truth 处理不需要的数据对象以进行标记。

您可以决定要发送到 Amazon SNS 主题的事件的类型。Ground Truth 在你发送时创建标签作业对象创建事件.

发送到您的 Amazon SNS 输入主题的事件结构必须是 JSON 消息，格式化时使用中的相同结构事件消息结构.

要查看如何使用 Amazon S3 控制台为 Amazon S3 存储桶设置事件通知的示例，AmazonSDK for .NET，以及AmazonSDK for Java，请按照此演练操作，演练：为存储桶配置通知（SNS 主题或 SQS 队列）在《Amazon Simple Storage Service 用户指南》中。

创建清单文件（可选）

创建流式标签作业时，您可以一次性选择将对象（例如图像或文本）添加到您在中指定的输入清单文件`ManifestS3Uri`的`CreateLabelingJob`. 当流式标签作业开始时，如果对象总数超过，则这些对象将发送给工作人员，或将这些对象添加到 Amazon SQS 队列`MaxConcurrentTaskCount`. 结果将添加到您在工作人员完成贴标任务时定期创建标签任务时指定的 Amazon S3 路径中。输出数据将发送到您订阅的输出主题的任何终端节点。

如果您想提供要标记的初始对象，请创建一个标识这些对象的清单文件并将其放置在 Amazon S3 中。在中指定清单文件的 S3 URI。`ManifestS3Uri`之内`InputConfig`.

要了解如何格式清单文件，请参阅。输入数据 (p. 378). 要使用 SageMaker 控制台自动生成清单文件（不支持 3D 点云任务类型），请参阅自动数据设置 (p. 379).

例如：使用 SageMaker API 创建流式标签 Job

以下是一个示例：AmazonPython SDK (Boto3) 请求您可以使用该功能在美国东部（弗吉尼亚北部）区域启动内置任务类型的流标记作业。有关以下各参数的更多信息，请参阅`CreateLabelingJob`. 要了解如何使用此 API 和相关语言特定的 SDK 创建标签作业，请参阅创建标记作业 (API).

在此示例中，请注意以下参数：

- `SnsDataSource`— 此参数出现在`InputConfig`和`OutputConfig`并分别用于识别您的输入和输出 Amazon SNS 主题。要创建直播标签作业，您需要提供 Amazon SNS 输入主题。或者，您还可以提供 Amazon SNS 输出主题。
- `S3DataSource`— 此参数是可选的。如果要包含要在标签作业启动后立即标记的数据对象的输入清单文件，请使用此参数。
- `StoppingConditions`— 创建流式标签作业时，将忽略此参数。要了解有关停止流式标签作业的更多信息，请参阅停止直播标签 Job (p. 364).
- 流式标签作业不支持自动化数据标签。请勿包含`LabelingJobAlgorithmsConfig`参数。

```
response = client.create_labeling_job(
    LabelingJobName= 'example-labeling-job',
    LabelAttributeName='label',
    InputConfig={
        'DataSource': {
            'S3DataSource': {
                'ManifestS3Uri': 's3://bucket/path/manifest-with-input-data.json'
            },
            'SnsDataSource': {
                'SnsTopicArn': 'arn:aws:sns:us-east-1:123456789012:your-sns-input-topic'
            }
        },
        'DataAttributes': {
            'ContentClassifiers': [
                'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
            ]
        }
    },
    OutputConfig={
```

```
        'S3OutputPath': 's3://bucket/path/file-to-store-output-data',
        'KmsKeyId': 'string',
        'SnsTopicArn': 'arn:aws:sns:us-east-1:123456789012:your-sns-output-topic'
    },
    RoleArn='arn:aws:iam::*:role/*',
    LabelCategoryConfigS3Uri='s3://bucket/path/label-categories.json',
    HumanTaskConfig={
        'WorkteamArn': 'arn:aws:sagemaker:us-east-1:*:workteam/private-crowd/*',
        'UiConfig': {
            'UiTemplateS3Uri': 's3://bucket/path/custom-worker-task-template.html'
        },
        'PreHumanTaskLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:PRE-tasktype',
        'TaskKeywords': [
            'Example key word',
        ],
        'TaskTitle': 'Multi-label image classification task',
        'TaskDescription': 'Select all labels that apply to the images shown',
        'NumberOfHumanWorkersPerDataObject': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'MaxConcurrentTaskCount': 123,
        'AnnotationConsolidationConfig': {
            'AnnotationConsolidationLambdaArn': 'arn:aws:lambda:us-
east-1:432418664414:function:ACS-tasktype'
        },
    Tags=[
        {
            'Key': 'string',
            'Value': 'string'
        },
    ]
)
```

停止直播标签 Job

您可以使用操作手动停止直播标签作业StopIlLabelingJob.

如果您的贴标作业闲置了 10 天以上，那么 Ground Truth 将自动停止该作业。在这种情况下，考虑标签工作闲如果没有向 Amazon SNS 输入主题发送任何对象，且您的 Amazon SQS 队列中没有对象保留，等待标签。例如，如果没有向 Amazon SNS 输入主题提供任何数据对象，并且提供给标签作业的所有对象都已标记，则 Ground Truth 会启动计时器。计时器启动后，如果在 10 天内没有收到任何商品，则标签作业将停止。

如果此 10 天限制对您的企业使用案例造成问题，请联系Amazon支持.

当标签作业停止时，其状态为`STOPPING`而 Ground Truth 将清理标签作业资源并从 Amazon SQS 队列中取消订阅 Amazon SNS 主题。Amazon SQS 是不由 Ground Truth 删除，因为此队列可能包含未处理的数据对象。如果您想避免从 Amazon SQS 产生额外费用，则应手动删除队列。要了解更多信息，请参阅Amazon SQS 定价.

创建具有标签类别和框架属性的标记类别配置文件

在使用亚马逊创建 3D 点云或视频帧标记作业时 SageMaker API 操作`CreateLabelingJob`，您可以使用标签类别配置文件指定标签和工作人员说明。或者，您还可以在标签类别属性文件中提供以下内容：

- 你可以提供标签类别属性对于视频帧和 3D 点云对象跟踪和对象检测任务类型。工作人员可以使用一个或多个属性提供有关对象的更多信息。例如，您可能希望使用 occluded 属性，以使工作人员确定对象何时被部分遮挡。您可以使用 `categoryAttributes` 参数为单个标签指定标签类别属性，或者使用 `categoryGlobalAttributes` 参数为所有标签指定标签类别属性。
- 你可以提供帧属性对于视频帧和 3D 点云对象跟踪和对象检测任务类型，请使用`frameAttributes`. 创建帧属性时，该属性将显示在工作器任务的每个帧或点云上。在视频帧标记作业中，这些属性是工作人员分

配给整个视频帧的属性。对于 3D 点云标记作业，这些属性将应用于单点云。可以使用帧属性，以让工作人员提供有关特定帧或点云中场景的更多信息。

- 对于视频帧标注作业，可以使用标签类别配置文件指定发送给工作人员的任务类型（边界框、折线、多边形或关键点）。

对于工作人员，为标签类别属性和框架属性指定值是可选的。

> **Important**
>
> 您只应在中提供标签属性名称`auditLabelAttributeName`如果您运行审核作业以验证或调整标签。在生成您希望工作人员调整的注释的标记作业中，可以使用该参数以输入使用的 LabelAttributeName。在控制台中创建标记作业时，如果您未指定标签属性名称，则会将作业的 Name (名称) 作为 LabelAttributeName。

主题

- 标签类别配置文件架构 (p. 365)
- 例如：3D 点云标记作业的标签类别配置文件 (p. 370)
- 例如：视频帧标签作业的标签类别配置文件 (p. 374)
- 创建工作人员说明 (p. 377)

标签类别配置文件架构

下表列出了您可以或必须在标签类别配置文件中包含的元素。

> **Note**
>
> 该参数`annotationType`仅支持视频帧标签作业。

参数	必需	接受的值	描述

`frameAttributes`	否	JSON 对象列表。
		每个 JSON 对象中的必需参数：
		`name`, `type`, `description`
		`minimum`和`maximum`如果您必需`type`是`"number"`
		每个 JSON 对象中的可选参数：
		`enum`, `editsAllowed`, `isRequired`
			可以使用该参数以创建一个框架属性，该属性将应用于标记作业中的所有帧或 3D 点云。有关更多信息，请参阅本节中的第三表。

`categoryGlobalAttributes`	否	JSON 对象列表。
		每个 JSON 对象中的必需参数：
		`name`, `type`
		`minimum`和`maximum`如果您必需`type`是`"number"`
		每个 JSON 对象中的可选参数：
		`description`, `enum`, `editsAllowed`, `isRequired`
			可以使用该参数以创建将应用于您在中指定的所有标签的标签类别属性。`labels`.
			有关更多信息，请参阅本节中的第三表。

参数	必需	接受的值	描述

`labels`	是	最多包含 30 个 JSON 对象的列表
		
		每个 JSON 对象中的必需参数：
		
		`label`
		
		每个 JSON 对象中的可选参数：
		
		`categoryAttributes`, `editsAllowed`
			可以使用该参数以指定标签或类。为每个类添加一个 `label`。
			
			要将标签类别属性添加到一个标签中，请将 `categoryAttributes` 添加到该标签中。
			
			使用`editsAllowed`以指定是否可以在调整标签作业中编辑标签。Set`editsAllowed`到`"none"`用于验证标记作业。
			
			有关更多信息，请参阅下表。

`annotationType`（仅支持视频帧标签作业）	否	字符串
		
		接受的参数：
		
		`BoundingBox`, `Polyline`, `Polygon`, `Keypoint`
		
		默认值：
		
		`BoundingBox`
			使用此选项可以指定视频帧标记作业的任务类型。例如，对于多边形视频帧对象检测任务，请选择`Polygon`.
			
			如果您未指定`annotationType`当你创建视频帧标签作业时，Ground Truth 将使用`BoundingBox`默认情况下。

`instructions`	否	JSON 对象 每个 JSON 对象中的必需参数： `"shortInstruction"`, `"fullInstruction"`	可以使用该参数以添加工作人员说明，以帮助工作人员完成其任务。有关工作人员说明的更多信息，请参阅工作人员说明 (p. 288)。 短说明必须少于 255 个字符，长说明必须少于 2,048 个字符。 有关更多信息，请参阅 创建工作人员说明 (p. 377)。
`auditLabelAttributeName`	调整和验证任务类型必需	字符串	输入要调整注释的标记作业中使用的 LabelAttributeName。 只有在为视频帧和 3D 点云对象检测、对象跟踪或 3D 点云语义分割创建调整作业时，才会使用该参数。

下表介绍了您可以或必须在以下情况下使用以创建列表的参数：Labels. 每个参数都应包含在 JSON 对象中。

参数	必需	接受的值	描述
label	是	字符串	向工作人员显示的标签类别的名称。每个标签类别名称必须唯一。
categoryAttributes	否	JSON 对象列表。 每个 JSON 对象中的必需参数： name, type minimum和maximum如果您必需type是"number" 每个 JSON 对象中的可选参数： description, enum, editsAllowed, isRequired	可以使用该参数将标签类别属性添加到特定的标签中，labels. 要向标签添加一个或多个标签类别属性，请包括categoryAttributes同样的 JSON 对象labels作为 JSON 对象label. 有关更多信息，请参阅下表。
editsAllowed	否	字符串 支持的值： "none"：不允许进行任何修改。 或 "any"（默认）：允许进行所有修改。	指定工作人员是否可以编辑标签。 用于视频帧或 3D 点云调整为作业添加标签，请将此参数添加到labels列表以指定工作人员是否可以编辑标签。 对于 3D 点云和视频帧验证为作业添加标签，用值添加此参数"none"到中的每个 JSON 对象labels列表。这将使所有标签都不可编辑。

下表介绍了您可以或必须在以下情况下使用以下方式创建框架属性的参数：frameAttributes和标签类别属性categoryGlobalAttributes和categoryAttributes参数。

参数	必需	接受的值	描述
name	是	字符串	可以使用该参数为标签类别或帧属性分配名称。这是工作人员看到的属性名称。 标签类别配置文件中的每个标签类别属性名称必须是唯一的。全局标

参数	必需	接受的值	描述
			签类别属性和标签特定标签类别属性不能具有相同的名称。
type	是	字符串 必需的值： "string" 或者 "number"	可以使用该参数定义标签类别或帧属性类型。 如果你指定"string"为了type并提供enum该属性的值，工作人员将可以从您提供的选择之一中选择。 如果你指定"string"为了type而且不提供enum值，工作人员可以输入自由格式文本。 如果你指定number为了type，工作人员可以在minimum和maximum你指定的数字。
enum	否	字符串列表	可以使用该参数以定义工作人员可以从该标签类别或框架属性中选择的选项。工作人员可以选择在 enum 中指定的一个值。例如，如果您指定["foo", "buzz", "bar"]对于enum，工作人员可以选择foo、buzz，或者bar. 您必须指定"string"为了type要使用enum列表。
description	frameAttributes：是 categoryAttributes要么categoryGlobalAttributes：否	字符串	可以使用该参数添加标签类别或帧属性的描述。您可以使用此字段向工作人员提供有关属性的更多信息。 此字段仅对于帧属性是必需的。
minimum 和 maximum	如果属性为必需type是"number"	整数	使用这些参数来指定工作人员可以为数字标签类别或框架属性输入的最小值和最大值（包括）值。 您必须指定"number"为了type可以使用minimum和maximum.

参数	必需	接受的值	描述
editsAllowed	否	字符串 必需的值： "none"：不允许进行任何修改。 或 "any"（默认）：允许进行所有修改。	指定工作人员是否可以编辑标签类别或框架属性。 用于视频帧或 3D 点云调整和验证标记作业，将此参数添加到标签类别和框架属性 JSON 对象中，以指定工作人员是否可以编辑属性。
isRequired	否	布尔值	指定是否需要工作人员对属性进行注释。在注释所有必需的属性之前，员工才能提交作业。

标签和标签类别属性配额

您最多可以为每个类指定 10 个标签类别属性。该 10 属性配额包括全局标签类别属性。例如，如果您创建 4 个全局标签类别属性，然后将三个标签类别属性分配给标签 x，则该标签总共具有 4+3=7 个标签类别属性。有关所有标签类别和标签类别属性限制，请参阅下表。

类型	Min	最大值
标签 (Labels)	1	30
标签名称字符配额	1	16
每个标签的标签类别属性（总和categoryAttributes和categoryGlobalAttributes）	0	10
每个标签的自由格式文本输入标签类别属性（总和categoryAttributes和categoryGlobalAttributes)。	0	5
帧属性	0	10
中的自由格式文本输入属性frameAttributes.	0	5
属性名称字符配额 (name)	1	16
属性描述字符配额 (description)	0	128
属性类型字符配额 (type)	1	16
允许的值enum列出string属性	1	10
中某个值的字符配额enum列表	1	16
自由格式文本的自由格式文本响应中的最大字符数frameAttributes	0	1000

类型	Min	最大值
自由格式文本的自由格式文本响应中的最大字符数categoryAttributes和categoryGlobalAttributes	0	80

例如：3D 点云标记作业的标签类别配置文件

在下表中选择一个选项卡，以查看用于对象检测、对象跟踪、语义分割、调整和验证标签作业的 3D 点云标签类别配置文件示例。

3D Point Cloud Object Tracking and Object Detection

以下是标签类别配置文件的示例，该文件包含 3D 点云对象检测或对象跟踪标记作业的标签类别属性。此示例包括两个帧属性，这些属性将添加到提交给标注作业的所有点云中。这些区域有：Car标签将包括四个标签类别属性 —X、Y、Z，以及全局属性，W.

```
{
    "documentVersion": "2020-03-01",
    "frameAttributes": [
        {
            "name":"count players",
            "description":"How many players to you see in the scene?",
            "type":"number"
        },
        {
            "name":"select one",
            "description":"describe the scene",
            "type":"string",
            "enum":["clear","blurry"],
            "isRequired":true
        },
    ],
    "categoryGlobalAttributes": [
        {
            "name":"W",
            "description":"label-attributes-for-all-labels",
            "type":"string",
            "enum": ["foo", "buzz", "biz"]
        }
    ],
    "labels": [
        {
            "label": "Car",
            "categoryAttributes": [
                {
                    "name":"X",
                    "description":"enter a number",
                    "type":"number",
                },
                {
                    "name":"Y",
                    "description":"select an option",
                    "type":"string",
                    "enum":["y1", "y2"]
                },
                {
                    "name":"Z",
                    "description":"submit a free-form response",
                    "type":"string",
                }
            ]
```

```
        },
        {
            "label": "Pedestrian",
            "categoryAttributes": [...]
        }
    ],
    "instructions": {"shortInstruction":"Draw a tight Cuboid", "fullInstruction":"<html
 markup>"}
}
```

3D Point Cloud Semantic Segmentation

以下是 3D 点云语义分割标记作业的标签类别配置文件的示例。

3D 点云语义分割任务类型不支持标签类别属性。支持帧属性。如果您为语义分割标记作业提供标签类别属性，将忽略这些属性。

```
{
    "documentVersion": "2020-03-01",
    "frameAttributes": [
        {
            "name":"count players",
            "description":"How many players to you see in the scene?",
            "type":"number"
        },
        {
            "name":"select one",
            "description":"describe the scene",
            "type":"string",
            "enum":["clear","blurry"]
        },
    ],
    "labels": [
        {
            "label": "Car",
        },
        {
            "label": "Pedestrian",
        },
        {
            "label": "Cyclist",
        }
    ],
    "instructions": {"shortInstruction":"Select the appropriate label and
 paint all objects in the point cloud that it applies to the same color",
 "fullInstruction":"<html markup>"}
}
```

在下表中选择一个选项卡，以查看 3D 点云验证或调整标签作业的标签类别配置文件示例。

3D Point Cloud Adjustment

以下是 3D 点云对象检测或对象跟踪调整标记作业的标签类别配置文件。对于 3D 点云语义分割调整标记作业，`categoryGlobalAttributes`和`categoryAttributes`不支持。

您必须包括`auditLabelAttributeName`以指定用于创建调整标签作业的上一个标签作业的标签属性名称。或者，您也可以使用`editsAllowed`参数以指定是否可以编辑标签或帧属性。

```
{
    "documentVersion": "2020-03-01",
    "frameAttributes": [
        {
```

```
            "name":"count players",
            "description":"How many players to you see in the scene?",
            "type":"number"
        },
        {
            "name":"select one",
            "editsAllowed":"none",
            "description":"describe the scene",
            "type":"string",
            "enum":["clear","blurry"]
        },
    ],
    "categoryGlobalAttributes": [
        {
            "name":"W",
            "editsAllowed":"any",
            "description":"label-attributes-for-all-labels",
            "type":"string",
            "enum": ["foo", "buzz", "biz"]
        }
    ],
    "labels": [
        {
            "label": "Car",
            "editsAllowed":"any",
            "categoryAttributes": [
                {
                    "name":"X",
                    "description":"enter a number",
                    "type":"number"
                },
                {
                    "name":"Y",
                    "description":"select an option",
                    "type":"string",
                    "enum":["y1", "y2"],
                    "editsAllowed":"any"
                },
                {
                    "name":"Z",
                    "description":"submit a free-form response",
                    "type":"string",
                    "editsAllowed":"none"
                }
            ]
        },
        {
            "label": "Pedestrian",
            "categoryAttributes": [...]
        }
    ],
    "instructions": {"shortInstruction":"Draw a tight Cuboid", "fullInstruction":"<html
 markup>"},
    // include auditLabelAttributeName for label adjustment jobs
    "auditLabelAttributeName": "myPrevJobLabelAttributeName"
}
```

3D Point Cloud Verification

以下是您可以用于 3D 点云对象检测或对象跟踪验证标记作业的标签类别配置文件。对于 3D 点云语义分割验证标记作业，`categoryGlobalAttributes`和`categoryAttributes`不支持。

您必须包括`auditLabelAttributeName`以指定用于创建验证标签作业的上一个标签作业的标签属性名称。此外，您必须使用`editsAllowed`参数以指定不能编辑任何标签。

```
{
    "documentVersion": "2020-03-01",
    "frameAttributes": [
        {
            "name":"count players",
            "editsAllowed":"any",
            "description":"How many players to you see in the scene?",
            "type":"number"
        },
        {
            "name":"select one",
            "editsAllowed":"any",
            "description":"describe the scene",
            "type":"string",
            "enum":["clear","blurry"]
        },
    ],
    "categoryGlobalAttributes": [
        {
            "name":"W",
            "editsAllowed":"none",
            "description":"label-attributes-for-all-labels",
            "type":"string",
            "enum": ["foo", "buzz", "biz"]
        }
    ],
    "labels": [
        {
            "label": "Car",
            "editsAllowed":"none",
            "categoryAttributes": [
                {
                    "name":"X",
                    "description":"enter a number",
                    "type":"number",
                    "editsAllowed":"none"
                },
                {
                    "name":"Y",
                    "description":"select an option",
                    "type":"string",
                    "enum":["y1", "y2"],
                    "editsAllowed":"any"
                },
                {
                    "name":"Z",
                    "description":"submit a free-form response",
                    "type":"string",
                    "editsAllowed":"none"
                }
            ]
        },
        {
            "label": "Pedestrian",
            "editsAllowed":"none",
            "categoryAttributes": [...]
        }
    ],
    "instructions": {"shortInstruction":"Draw a tight Cuboid", "fullInstruction":"<html
 markup>"},
    // include auditLabelAttributeName for label verification jobs
    "auditLabelAttributeName": "myPrevJobLabelAttributeName"
}
```

例如：视频帧标签作业的标签类别配置文件

工作人员可用的注释工具和使用的任务类型取决于您为其指定的值annotationType. 例如，如果您希望工作人员使用关键点来跟踪多个帧中特定对象姿势的变化，则应指定Keypoint(对于)annotationType. 如果您未指定注释类型，BoundingBox默认使用。

以下是具有标签类别属性的视频帧关键点标签类别配置文件的示例。此示例包括两个帧属性，这两个属性将添加到提交给标注作业的所有帧中。这些区域有：Car标签将包括四个标签类别属性 —X、Y、Z，以及全局属性，W.

```
{
    "documentVersion": "2020-03-01",
    "frameAttributes": [
        {
            "name":"count players",
            "description":"How many players to you see in the scene?",
            "type":"number"
        },
        {
            "name":"select one",
            "description":"describe the scene",
            "type":"string",
            "enum":["clear","blurry"]
        },
    ],
    "categoryGlobalAttributes": [
        {
            "name":"W",
            "description":"label-attributes-for-all-labels",
            "type":"string",
            "enum": ["foo", "buz", "buz2"]
        }
    ],
    "labels": [
        {
            "label": "Car",
            "categoryAttributes": [
                {
                    "name":"X",
                    "description":"enter a number",
                    "type":"number",
                },
                {
                    "name":"Y",
                    "description":"select an option",
                    "type":"string",
                    "enum": ["y1", "y2"]
                },
                {
                    "name":"Z",
                    "description":"submit a free-form response",
                    "type":"string",
                }
            ]
        },
        {
            "label": "Pedestrian",
            "categoryAttributes": [...]
        }
    ],
    "annotationType":"Keypoint",
    "instructions": {"shortInstruction":"add example short instructions here",
 "fullInstruction":"<html markup>"}
}
```

从下表中选择一个选项卡，查看视频帧调整和验证标签作业的标签类别配置文件示例。

Video Frame Adjustment

以下是您可以用于视频帧调整标记作业的标签类别配置文件。

您必须包括`auditLabelAttributeName`以指定用于创建验证标签作业的上一个标签作业的标签属性名称。或者，您也可以使用`editsAllowed`参数来指定是否可以编辑标签、标签类别属性或框架属性。

```
{
    "documentVersion": "2020-03-01",
    "frameAttributes": [
        {
            "name":"count players",
            "editsAllowed":"none",
            "description":"How many players to you see in the scene?",
            "type":"number"
        },
        {
            "name":"select one",
            "description":"describe the scene",
            "type":"string",
            "enum":["clear","blurry"]
        },
    ],
    "categoryGlobalAttributes": [
        {
            "name":"W",
            "editsAllowed":"any",
            "description":"label-attributes-for-all-labels",
            "type":"string",
            "enum": ["foo", "buz", "buz2"]
        }
    ],
    "labels": [
        {
            "label": "Car",
            "editsAllowed":"any",
            "categoryAttributes": [
                {
                    "name":"X",
                    "description":"enter a number",
                    "type":"number",
                    "editsAllowed":"any"
                },
                {
                    "name":"Y",
                    "description":"select an option",
                    "type":"string",
                    "enum": ["y1", "y2"],
                    "editsAllowed":"any"
                },
                {
                    "name":"Z",
                    "description":"submit a free-form response",
                    "type":"string",
                    "editsAllowed":"none"
                }
            ]
        },
        {
            "label": "Pedestrian",
            "editsAllowed":"none",
            "categoryAttributes": [...]
        }
```

```
    ],
    "annotationType":"Keypoint",
    "instructions": {"shortInstruction":"add example short instructions here",
 "fullInstruction":"<html markup>"},
    // include auditLabelAttributeName for label adjustment jobs
    "auditLabelAttributeName": "myPrevJobLabelAttributeName"
}
```

Video Frame Verification

以下是视频帧标记作业的标签类别配置文件的示例。

您必须包括auditLabelAttributeName以指定用于创建验证标签作业的上一个标签作业的标签属性名称。此外，您必须使用editsAllowed参数以指定不能编辑任何标签。

```
{
    "documentVersion": "2020-03-01",
    "frameAttributes": [
        {
            "name":"count players",
            "editsAllowed":"none",
            "description":"How many players to you see in the scene?",
            "type":"number"
        },
        {
            "name":"select one",
            "editsAllowed":"any",
            "description":"describe the scene",
            "type":"string",
            "enum":["clear","blurry"]
        },
    ],
    "categoryGlobalAttributes": [
        {
            "name":"W",
            "editsAllowed":"none",
            "description":"label-attributes-for-all-labels",
            "type":"string",
            "enum": ["foo", "buz", "buz2"]
        }
    ],
    "labels": [
        {
            "label": "Car",
            "editsAllowed":"none",
            "categoryAttributes": [
                {
                    "name":"X",
                    "description":"enter a number",
                    "type":"number",
                    "editsAllowed":"any"
                },
                {
                    "name":"Y",
                    "description":"select an option",
                    "type":"string",
                    "enum": ["y1", "y2"],
                    "editsAllowed":"any"
                },
                {
                    "name":"Z",
                    "description":"submit a free-form response",
                    "type":"string",
                    "editsAllowed":"none"
```

```
                }
            ]
        },
        {
            "label": "Pedestrian",
            "editsAllowed":"none",
            "categoryAttributes": [...]
        }
    ],
    "annotationType":"Keypoint",
    "instructions": {"shortInstruction":"add example short instructions here",
 "fullInstruction":"<html markup>"},
    // include auditLabelAttributeName for label adjustment jobs
    "auditLabelAttributeName": "myPrevJobLabelAttributeName"
}
```

创建工作人员说明

创建有关标记作业的自定义说明，以提高工作人员完成其任务的准确性。当工作人员在工作人员 UI 中选择 Instructions (说明) 菜单选项时，便可以访问您的说明。短说明必须少于 255 个字符，长说明必须少于 2,048 个字符。

说明分为两种：

- 简短说明— 选择这些说明时，将向他们显示这些说明。说明在工作人员 UI 菜单中。它们应提供简单的参考内容，以向工作人员显示标记对象的正确方法。
- 完整说明— 当工作人员选择时会显示这些说明更多说明在说明中，弹出窗口。如果要完成的任务使用多个示例来显示边缘情况和其他难以标记对象的情况，我们建议您提供详细说明。

对于 3D 点云和视频帧标记作业，可以将工作人员说明添加到标签类别配置文件中。您可以使用单个字符串以创建说明，也可以添加 HTML 标记以自定义说明外观和添加图像。确保您在说明中包含的任何图像是公开可用的，或者如果说明位于 Amazon S3 中，请确保工作人员具有读取访问权限，以便他们可以查看这些说明。

使用输入和输出数据

您向 Amazon SageMaker Ground Truth 提供的输入数据将发送给工作人员进行标记。您可以通过创建单个清单文件（该文件定义所有需要标记的数据）或将输入数据对象发送到要实时标记的正在进行的流式标记作业，以选择要发送给工作人员的数据。

输出数据是标记作业的结果。输出数据文件，或增强清单文件，它们包含您发送给标记作业的每个对象的标签数据和有关分配给数据对象的标签的元数据。

使用任务类型内置的图像分类（单个和多标签）、文本分类（单标签和多标签）、对象检测和语义分段来创建标注作业时，可以使用生成的增强清单文件启动 SageMaker 培训作业。有关如何使用增强清单通过 Amazon SageMaker 训练对象检测机器学习模型的演示，请参阅ipynb. 有关更多信息，请参阅 通过增强清单文件，将数据集元数据提供给训练作业 (p. 2090)。

主题

- 输入数据 (p. 378)
- 3D 点云输入数据 (p. 388)
- 视频帧输入数据 (p. 409)
- 输出数据 (p. 414)

输入数据

输入数据是您发送给您的人力进行标记的数据对象。有两种方法可以将数据对象发送到 Ground Truth 以进行标记：

- 使用输入清单文件发送需要标记的数据对象的列表。
- 将单个数据对象实时发送到永久运行的流式标签作业。

如果您的数据集需要一次标记，并且不需要持续的标签作业，请使用输入清单文件创建标准标签作业。

如果要在标签作业开始后定期向其发送新数据对象，请创建一个流式标签作业。创建流式标签作业时，您可以选择使用输入清单文件来指定要在作业开始时立即标记的一组数据。只要流式标签作业处于活动状态，您就可以持续将新数据对象发送到该作业。

> Note
>
> 只有通过 SageMaker API 才支持流式标签作业。您无法使用 SageMaker 控制台创建流式标签作业。

以下任务类型有特殊的输入数据要求和选项：

- 适用于3D 点云标记作业输入数据要求，请参阅3D 点云输入数据 (p. 388).
- 适用于视频帧标记作业输入数据要求，请参阅视频帧输入数据 (p. 409).

主题

- 使用输入清单文件 (p. 378)
- 自动数据设置 (p. 379)
- 支持的数据格式 (p. 380)
- Ground Truth 直播标签职位 (p. 381)
- 输入数据配额 (p. 385)
- 筛选和选择要标记的数据 (p. 387)

使用输入清单文件

输入清单文件中的每一行都是包含要标签的对象或对象的引用的条目。条目还可以包含来自之前作业的标签，对于某些任务类型，还可以包含其他信息

输入数据和清单文件必须存储在 Amazon Simple Storage Service (Amazon S3) 中。它们具有特定的存储和访问要求，如下所示：

- 包含输入数据的 Amazon S3 存储桶必须位于同一位置。Amazon您运行 Amazon SageMaker Ground Truth 的地区。您必须授予 Amazon SageMaker 对存储在 Amazon S3 存储桶中的数据的访问权限，以便读取数据。有关 Amazon S3 存储桶的更多信息，请参阅使用 Amazon S3 存储桶.
- 清单文件必须位于同一位Amazon区域与数据文件一样，但不需要与数据文件位于同一位置。它可以存储在任何 Amazon S3 存储桶中，该存储桶可由Amazon Identity and Access Management在您创建标记作业时分配给 Ground Truth 的 (IAM) 角色。

> Note
>
> 3D 点云和视频帧任务类型具有不同的输入清单要求和属性。
> 适用于3D 点云任务类型请参阅为 3D 点云标记作业创建输入清单文件 (p. 389).
> 适用于视频帧任务类型请参阅创建视频帧输入清单文件 (p. 413).

清单是一个 UTF-8 编码的文件，其中每行都是完整且有效的 JSON 对象。每行都以标准换行符 \n 或 \r\n 分隔。由于每行都必须是有效的 JSON 对象，因此您不能使用未转义的换行符。有关数据格式的更多信息，请参阅 JSON 行。

清单文件中的每个 JSON 对象不得超过 100,000 个字符。对象中的任何单个属性都不能超过 20,000 个字符。属性名称不能以 $（美元符号）开头。

清单文件中的每个 JSON 对象必须包含以下键之一：source-ref 或 source。密钥的值说明如下：

- source-ref— 对象的源是在密钥值中指定的 Amazon S3 对象。当对象是二进制对象（如图像）时使用此值。
- source— 对象的源是密钥值。当对象为文本值时使用此值。

以下是 Amazon S3 存储桶中存储的文件的清单文件示例。

```
{"source-ref": "S3 bucket location 1"}
{"source-ref": "S3 bucket location 2"}
   ...
{"source-ref": "S3 bucket location n"}
```

使用source-ref用于边界框、图像分类（单标签和多标签）、语义分割和视频分类标记作业的视频剪辑的图像文件的键。3D 点云和视频帧标签作业还使用source-refkey 但是这些标签作业需要输入清单文件中的其他信息。有关更多信息，请参阅3D 点云输入数据 (p. 388)和视频帧输入数据 (p. 409).

下面是一个含有输入数据（存储在清单中）的清单文件示例：

```
{"source": "Lorem ipsum dolor sit amet"}
{"source": "consectetur adipiscing elit"}
   ...
{"source": "mollit anim id est laborum"}
```

使用source键用于单标签和多标签文本分类以及命名实体识别标记作业。

您可以在清单文件中包括其他密钥值对。这些对将传递到输出文件且保持不变。当您希望在应用程序之间传递信息时，此方法很有用。有关更多信息，请参阅 输出数据 (p. 414)。

自动数据设置

您可以使用自动数据设置，在 Ground Truth 控制台中，使用图像、视频、视频框架、文本 (.txt) 文件和存储在 Amazon S3 中的逗号分隔值 (.csv) 文件为标记作业创建清单文件。使用自动数据设置时，您可以指定存储输入数据的 Amazon S3 位置和输入数据类型，Ground Truth 会在指定位置中查找与该类型匹配的文件。

> Note
>
> Ground Truth 不使用Amazon KMS键可访问输入数据或在您指定的 Amazon S3 位置写入输入清单文件。创建标签作业的 IAM 用户或角色必须具有访问 Amazon S3 中输入数据对象的权限。

在使用以下过程之前，请确保输入图像或文件的格式正确：

- 图像文件 — 图像文件必须遵守中的表列出的大小和分辨率限制。输入文件大小限制 (p. 385).
- 文本文件 — 文本数据可以存储在一个或多个 .txt 文件中。要标记的每个项目必须用标准换行符分隔。
- CSV 文件 — 文本数据可以存储在一个或多个 .csv 文件中。要标记的每个项目必须位于单独的行中。
- 视频 — 视频文件可以是以下任意格式：.mp4、.ogg 和 .webm。如果要从视频文件中提取视频帧以进行对象检测或对象跟踪，请参阅提供视频文件 (p. 410).

- 视频帧 — 视频帧是从视频中提取的图像。从单个视频中提取的所有图像都称为视频帧序列. 在 Amazon S3 中，每个视频帧序列必须具有唯一的前缀键。请参阅 提供视频帧 (p. 409)。有关此数据类型，请参阅自动视频帧输入数据设置 (p. 411)

  Important

  有关视频帧对象检测和视频帧对象跟踪标记作业，请参阅自动视频帧输入数据设置 (p. 411)了解如何使用自动化数据设置。

使用这些说明自动设置与 Ground Truth 的输入数据集连接。

使用 Ground Truth 自动连接 Amazon S3 中的数据

1. 导航到创建标记作业在位于的 Amazon SageMaker 控制台中的页面https://console.aws.amazon.com/sagemaker/.

   此链接将您带到弗吉尼亚北部（us-east-1）Amazon区域。如果您的输入数据位于另一个区域的 Amazon S3 存储桶中，请切换到该区域。更改您的Amazon地区，在导航栏中，选择当前显示区域的名称。
2. Select创建标记作业.
3. 输入 Job name（任务名称）。
4. 在部分中输入数据设置，请选择自动化数据设置.
5. 输入 Amazon S3 URI输入数据集的 S3 位置.
6. 指定您的输出数据集的 S3 位置. 这是存储输出数据的地方。
7. 选择您的数据类型使用下拉列表。
8. 使用下拉菜单IAM 角色以选择执行角色。如果您选择创建新角色中，指定您希望授予此角色访问权限的 Amazon S3 存储桶。此角色必须有权访问您在步骤 5 和步骤 6 中指定的 S3 存储桶。
9. Select完成数据设置.

以下 GIF 演示了如何将自动化数据设置用于图像数据。这个例子将创建一个文件，`dataset-YYMMDDTHHMMSS.manifest`在 Amazon S3 存储桶中`example-groundtruth-images`哪里`YYMMDDTHHmmSS`表示年份（`YY`）、月 (`MM`)、1 天 (`DD`) 和时间（以小时为单位）`HH`)、分钟 (`mm`) 和秒 (`ss`)，输入清单文件已创建。

支持的数据格式

当您为内置任务类型对于相应输入数据类型，您的输入数据必须为以下支持文件格式之一。要了解自动数据设置的信息，请参阅自动数据设置 (p. 379).

Tip

使用自动数据设置时，可以使用其他数据格式为基于视频帧和文本的任务类型生成输入清单文件。

任务类型	输入数据类型	Support 格式	示例输入清单行
边界框、语义分割、图像分类（单标签和多标签）、验证和调整标签	映像	.jpg、.jpeg、.png	`{"source-ref": "s3://DOC-EXAMPLE-BUCKET1/example-image.png"}`
命名实体识别、文本分类（单个和多标签）	文本	原始文本	`{"source": "Lorem ipsum dolor sit amet"}`

任务类型	输入数据类型	Support 格式	示例输入清单行
视频分类	视频剪辑	.mp4、.ogg 和 .webm	`{"source-ref": "s3:///example-video.mp4"}`
视频帧对象检测、视频帧对象跟踪（边框、折线、多边形或关键点）	视频帧和视频帧序列文件（用于对象跟踪）	视频 帧: .jpg、.jpeg、.png 序列文件: .json	请参阅创建视频帧输入清单文件 (p. 413)。
3D 点云语义分割、3D 点云对象检测、3D 点云对象跟踪	点云和点云序列文件（用于对象跟踪）	点云：二进制包格式和 ASCII。有关更多信息，请参阅接受的原始 3D 数据格式 (p. 388)。 序列文件: .json	请参阅为 3D 点云标记作业创建输入清单文件 (p. 389)。

Ground Truth 直播标签职位

如果您想永久向 Amazon SageMaker Ground Truth 发送新数据对象以进行标签，请使用直播标签作业。直播标签作业允许您：

- 使用永久运行的标注作业将新的数据集对象实时发送给工作人员。只要标签作业处于活动状态且正在向其发送新对象，工作人员就会持续接收要标记的新数据对象。
- 了解已排队并等待标记的对象的数量。使用此信息控制发送到标签作业的数据对象的流。
- 工作人员完成标签后，实时接收单个数据对象的标签数据。

Ground Truth 直播标签作业将保持活动状态，直到手动停止或空闲超过 10 天。在标签作业处于活动状态时，您可以间歇性地向工作人员发送新的数据对象。

如果您是 Ground Truth 直播标记作业的新用户，建议您查阅。工作方式 (p. 381).

使用创建直播标记 Job (p. 360)以了解如何创建流式标记作业。

> Note
>
> Ground Truth 流标签作业只能通过 SageMaker API 支持。

主题

- 工作方式 (p. 381)
- 将数据发送到流式标记 Job (p. 382)
- 使用 Amazon SQS 队列管理贴标请求 (p. 383)
- 从流式标签 Job 接收输出数据 (p. 383)
- 处理重复消息 (p. 383)

工作方式

当您创建 Ground Truth 流式标签作业时，该作业将保持活动状态，直到手动停止、空闲超过 10 天或无法访问输入数据源。当工作人员处于活动状态时，您可以间歇性地将新数据对象发送给工作 只要工作人员当前可用的任务总数少于中的值，工作人员就可以继续实时接收新的数据对象`MaxConcurrentTaskCount`. 否则，数据对象将被发送到 Ground Truth 代表您创建的队列Amazon Simple Queue Service(Amazon SQS) 以便稍后处理。一旦工作人员当前可用的任务总数低于工作人员，这些任务就会立即发送给工作人员`MaxConcurrentTaskCount`. 如果数据对象在 14 天后没有发送给工作人员，则该数据对象将过期。您可

以查看队列中待处理的任务数，并调整发送到标记作业的对象数。例如，如果待处理对象的积压超过阈值，则可能会降低将对象发送到标记作业的速度。

将数据发送到流式标记 Job

在使用输入清单文件创建标签作业时，您可以选择性地将输入数据提交到流式标签作业一次。一旦标签作业开始并且状态为`InProgress`，您可以使用 Amazon SNS 输入主题和 Amazon S3 事件通知实时向标签作业提交新数据对象。

启动标签 Job 时提交数据对象（一次性）：

- 使用输入清单文件— 您可以选择在中指定输入清单文件 Amazon S3 URI`ManifestS3Uri`当你创建流媒体标签作业时。在标签作业开始后，Ground Truth 会立即将清单文件中的每个数据对象发送给工作人员进行标记。要了解更多信息，请参阅 创建清单文件（可选）(p. 363)。

  提交创建流式标签作业的请求后，其状态将为`Initializing`. 标签作业处于活动状态后，状态将更改为`InProgress`你可以开始使用实时选项来提交额外的数据对象以进行标记。

实时提交数据对象：

- 使用 Amazon SNS 消息发送数据对象— 您可以通过发送 Amazon SNS 消息将 Ground Truth 新的数据对象发送到标签。您将将此消息发送到您创建的 Amazon SNS 输入主题，并在创建流式标签作业时指定该主题。有关更多信息，请参阅 使用 Amazon SNS 发送数据对象 (p. 382)。
- 通过将数据对象放入 Amazon S3 存储桶中发送数据对象— 每次向 Amazon S3 存储桶添加新数据对象时，都可以提示 Ground Truth 处理该对象以进行标记。为此，您可以向存储桶添加事件通知，以便每次向 (或) 添加新对象时通知您的 Amazon SNS 输入主题创建于) 存储桶。有关更多信息，请参阅 使用 Amazon S3 发送数据对象 (p. 383)。此选项不适用于基于文本的标签作业，例如文本分类和命名实体识别。

  > **Important**
  >
  > 如果您使用 Amazon S3 配置，请勿将相同的 Amazon S3 位置用于输入数据配置和输出数据。在您创建标记作业时，您可以为输出数据指定 S3 前缀。

使用 Amazon SNS 发送数据对象

您可以使用 Amazon Simple Notification Service (Amazon SNS) 将数据对象发送到直播标记作业。Amazon SNS 是一项 Web 服务，用于协调和管理往返传输消息的过程。终端（例如，电子邮件地址或Amazon Lambda函数)。Amazon SNS话题充当两个或多个终端之间的通信渠道。您使用 Amazon SNS 发送，或发布中指定的主题的新数据对象`CreateLabelingJob`参数`SnsTopicArn`在`InputConfig`. 这些消息的格式与来自输入清单文件.

例如，您可以通过将文本发布到输入主题来将文本发送到活动文本分类标签作业。您发布的消息类似于以下内容：

```
{"source": "Lorem ipsum dolor sit amet"}
```

要将新图像对象发送到图像分类标签作业，您的消息可能类似于以下内容：

```
{"source-ref": "s3://awsexamplebucket/example-image.jpg"}
```

> **Note**
>
> 您还可以在 Amazon SNS 消息中包含自定义重复数据消除 ID 和重复数据消除密钥。要了解更多信息，请参阅 处理重复消息 (p. 383)。

当 Ground Truth 创建您的直播标签作业时，它会订阅您的 Amazon SNS 输入主题。

使用 Amazon S3 发送数据对象

您可以将一个或多个新数据对象发送到流式标签作业，方法是将它们放在配置了 Amazon SNS 事件通知的 Amazon S3 存储桶中。您可以设置一个事件，以便在存储桶中创建新对象时通知您的 Amazon SNS 输入主题。Amazon SNS 必须在`CreateLabelingJob`参数`SnsTopicArn`在`InputConfig`.

每当您将 Amazon S3 存储桶配置为向 Amazon SNS 发送通知时，Ground Truth 都会发布测试事件，`"s3:TestEvent"`，以确保主题存在，并且指定的 Amazon S3 存储桶的拥有者有权发布到指定主题。建议您在开始直播标签作业之前，先设置 Amazon S3 与 Amazon SNS 的连接。如果不这样做，此测试事件可能会注册为数据对象并发送到 Ground Truth 进行标签。

> Important
>
> 如果您使用 Amazon S3 配置，请勿将相同的 Amazon S3 位置用于输入数据配置和输出数据。在您创建标记作业时，您可以为输出数据指定 S3 前缀。
> 对于基于映像的标签作业，Ground Truth 要求所有 S3 存储桶都附加 CORS 策略。要了解更多信息，请参阅 CORS 许可要求 (p. 446)。

配置 Amazon S3 存储桶并创建标签作业后，您可以将对象添加到存储桶中，然后 Ground Truth 将该对象发送给工作人员或将其放在 Amazon SQS 队列中。

要了解更多信息，请参阅 设置 Amazon S3 存储桶事件通知 (p. 362)。

> Important
>
> 此选项不适用于基于文本的标签作业，例如文本分类和命名实体识别。

使用 Amazon SQS 队列管理贴标请求

当 Ground Truth 创建您的直播标签作业时，它会在Amazon用于创建标签作业的帐户。队列名称为`GroundTruth-`*`labeling_job_name`*哪里*`labeling_job_name`*是你的标签工作的名称，用小写字母表示。当您将数据对象发送到标记作业时，Ground Truth 要么将数据对象直接发送给工作人员，或者将任务放在队列中以便稍后处理。如果数据对象在 14 天后没有发送给工作人员，则该数据对象将过期并从队列中删除。您可以在 Amazon SQS 中设置警报以检测对象何时过期，并使用此机制控制发送到标签作业的对象的数量。

> Important
>
> 修改、删除对象或直接将对象发送到与您的流式标签作业关联的 Amazon SQS 队列可能会导致作业失败。

从流式标签 Job 接收输出数据

您的 Amazon S3 输出存储桶会定期使用流式标签作业中的新输出数据进行更新。

或者，您可以指定 Amazon SNS 输出主题。每次工作人员提交带标签的对象时，都会向该主题发送包含输出数据的通知。您可以为终端节点订阅 SNS 输出主题，以便在接收标签任务的输出数据时接收通知或触发事件。如果您想实时链接到另一个流式作业，并在工作人员每次提交数据对象时都会收到 Amazon SNS 通知，请使用 Amazon SNS 输出主题。

要了解更多信息，请参阅 为终端节点订阅您的 Amazon SNS 输出主题 (p. 362)。

处理重复消息

对于实时发送的数据对象，Ground Truth 通过确保每个唯一对象仅发送一次进行标记，即使指向该对象的输入消息被多次接收（重复消息），从而保证幂等性。为此，发送到流式标签作业的每个数据对象都会被分配重复数据删除 ID，其标识为重复数据删除键.

如果您使用 Amazon SNS 消息直接通过 Amazon SNS 输入主题发送标记数据对象的请求，则可以选择为对象选择自定义重复数据消除密钥和重复数据删除 ID。有关更多信息，请参阅 在 Amazon SNS 消息中指定重复数据删除密钥和 ID (p. 384)。

如果您没有提供自己的重复数据消除密钥，或者如果您使用 Amazon S3 配置将数据对象发送到标签作业，Ground Truth 将使用以下选项之一作为重复数据消除 ID：

- 对于直接发送到您的 Amazon SNS 输入主题的消息，Ground Truth 使用 SNS 消息 ID。
- 对于来自 Amazon S3 配置的消息，Ground Truth 将对象的 Amazon S3 URI 与序列器令牌在消息中。

在 Amazon SNS 消息中指定重复数据删除密钥和 ID

当您使用 Amazon SNS 消息将数据对象发送到直播标签作业时，您可以选择通过以下方式之一指定重复数据删除密钥和重复数据消除 ID。在所有这些情况下，请使用识别重复数据删除密钥`dataset-objectid-attribute-name`.

带上自己的重复数据删除密钥和 ID

通过按如下方式配置 Amazon SNS 消息，创建自己的重复数据消除密钥和重复数据消除 ID。Replace*byo-key*用你的钥匙*UniqueId*使用该数据对象的重复数据删除 ID。

```
{
    "source-ref":"s3://bucket/prefix/object1",
    "dataset-objectid-attribute-name":"byo-key",
    "byo-key":"UniqueId"
}
```

您的重复数据删除密钥最多 140 个字符。支持的模式包括：`"^[$a-zA-Z0-9](-*[a-zA-Z0-9])*"`.

您的重复数据删除 ID 最多可以有 1024 个字符。支持的模式包括：`^(https|s3)://([^/]+)/?(.*)$`.

将现有密钥用于重复数据删除密钥

您可以在消息中使用现有密钥作为重复数据消除密钥。执行此操作时，与该键关联的值将用于重复数据消除 ID。

例如，您可以指定使用`source-ref`键作为重复数据删除密钥，方法是将消息格式化如下：

```
{
    "source-ref":"s3://bucket/prefix/object1",
    "dataset-objectid-attribute-name":"source-ref"
}
```

在此示例中，Ground Truth 使用`"s3://bucket/prefix/object1"`对于重复数据删除 ID。

在输出数据中查找重复数据删除密钥和 ID

您可以在输出数据中看到重复数据消除密钥和 ID。重复数据删除密钥的标识方式是`dataset-objectid-attribute-name`.

当您使用自己的自定义重复数据删除密钥时，输出内容包含类似于以下内容：

```
"dataset-objectid-attribute-name": "byo-key",
"byo-key": "UniqueId",
```

如果不指定密钥，则可以找到 Ground Truth 分配给数据对象的重复数据删除 ID，如下所示。这些区域有：`$`*label-attribute-name*`-object-id`参数标识重复数据消除 ID。

```
{
    "source-ref":"s3://bucket/prefix/object1",
    "dataset-objectid-attribute-name":"$label-attribute-name-object-id"
    "label-attribute-name" :0,
    "label-attribute-name-metadata": {...},
    "$label-attribute-name-object-id":"<service-generated-key>"
}
```

适用于*<service-generated-key>*，如果数据对象是通过 Amazon S3 配置来的，Ground Truth 会添加服务使用的唯一值，并发出一个新字段$*sequencer*其中显示了使用的 Amazon S3 序列器。如果对象直接馈送到 SNS，Ground Truth 将使用 SNS 消息 ID。

Note

请勿使用$标签属性名称中的字符。

输入数据配额

语义分割标记作业中使用的输入数据集的配额为 20000 个项目。对于所有其他标注作业类型，数据集大小配额为 100,000 项。要请求增加标注除语义分割作业以外标记作业的配额，请查看中的过程。AmazonService Quotas请求提高配额。

主动学习标记作业和非主动学习标记作业的输入图像数据不得超过大小和分辨率限制。主动学习 是指使用自动化数据标记的标记作业。非主动学习 是指不使用自动化数据标记的标记作业。

其他配额适用于所有任务类型的标签类别，以及 3D 点云和视频帧任务类型的输入数据和标注类别属性。

输入文件大小限制

对于主动学习标记作业和非主动学习标记作业，输入文件不能超过以下大小限制。中使用的视频没有输入文件大小配额视频分类标记作业。

标记 Job 任务类型	输入文件大小限制
图像分类	40MB
边界框（物体检测）	40MB
语义分割	40MB
边界框（物体检测）标签调整	40MB
语义分割标签调整	40MB
边界框（物体检测）标签验证	40MB
语义分割标签验证	40MB

输入图像分辨率限制

图像文件分辨率是指图像中的像素数，决定了图像保存的细节量。图像分辨率限制因标记作业类型和使用的 SageMaker 内置算法而异。下表列出了主动和非主动学习标记作业中使用的图像的分辨率限制。

标记 Job 任务类型	分辨率限制 - 非主动学习	分辨率限制 - 主动学习
图像分类	1 亿像素	3840 x 2160 像素 (4 K)
边界框（物体检测）	1 亿像素	3840 x 2160 像素 (4 K)
语义分割	1 亿像素	1920 x 1080 像素 (1080 p)
对象检测标签调整	1 亿像素	3840 x 2160 像素 (4 K)
语义分割标签调整	1 亿像素	1920 x 1080 像素 (1080 p)
对象检测标签验证	1 亿像素	不可用
语义分割标签验证	1 亿像素	不可用

标记类别配额

每种标签作业任务类型都有可以指定的标签类别数量的配额。工作人员选择标签类别以创建注释。例如，您可以指定标签类别汽车、行人的，和骑自行车的人在创建边界框标记作业时，工作人员将选择汽车在汽车周围绘制边框之前的类别。

> **Important**
>
> 标签分类名称不能超过 256 个字符。
> 所有标签分类必须唯一。您不能指定重复的标签类别。

以下标签类别限制适用于标签作业。标签类别的配额取决于您是否使用 SageMaker API 操作`CreateLabelingJob`或者通过控制台来创建标记作业。

标记 Job 任务类型	标签类别配额-API	标签类别配额-控制台
图像分类（多标签）	50	50
图像分类（单个标签）	无限	30
边界框（物体检测）	50	50
标签验证	无限	30
语义分割（通过主动学习）	20	10
语义分割（没有主动学习）	无限	10
命名实体识别	无限	30
文本分类（多标签）	50	50
文本分类（单个标签）	无限	30
视频分类	30	30
视频帧对象检测	30	30
视频帧对象跟踪	30	30
3D 点云物体检测	30	30
3D 点云对象跟踪	30	30
3D 点云语义分割	30	30

3D 点云和视频帧标记 Job 配额

以下配额适用于 3D 点云和视频帧标记作业输入数据。

标记 Job 任务类型	输入数据配额
视频帧对象检测	每个序列 2,000 个视频帧（图像）
视频帧对象检测	每个清单文件 10 个视频帧序列
视频帧对象跟踪	每个序列 2,000 个视频帧（图像）
视频帧对象跟踪	每个清单文件 10 个视频帧序列

标记 Job 任务类型	输入数据配额
3D 点云物体检测	每个标签作业 100,000 个点云帧
3D 点云对象跟踪	每个标记作业 100,000 个点云帧序列
3D 点云对象跟踪	每个序列文件中有 500 个点云帧

在创建视频帧或 3D 点云标记作业时，您可以添加一个或多个标签类别属性您指定为让工作人员提供有关注释的更多信息的标签类别。

每个标签类别属性都有一个标签类别属性`name`，以及一个或多个选项（值）可供选择的列表。要了解更多信息，请参阅工作人员用户界面 (UI) (p. 287)对于 3D 点云标记作业和工作人员用户界面 (UI) (p. 246)对于视频帧标记作业。

以下配额适用于可以为标记作业指定的标签类别属性名称和值的数量。

标记 Job 任务类型	标签类别属性（名称）配额	标签类别属性值配额
视频帧对象检测	10	10
视频帧对象跟踪	10	10
3D 点云物体检测	10	10
3D 点云对象跟踪	10	10
3D 点云语义分割	10	10

筛选和选择要标记的数据

您可以使用 Amazon SageMaker 控制台选择数据集中要标记的部分。数据必须存储在 Amazon S3 存储桶中。您有三种选择：

- 使用完整的数据集。
- 选择数据集的一个随机选择样本。
- 使用查询指定数据集的一个子集。

以下选项可用于标记作业的 部分SageMaker 控制台选择之后创建标记作业. 要了解如何在控制台中创建标记作业，请参阅开始使用 (p. 206). 要配置用于标记的数据集，请在 Job overview (作业概述) 部分中选择 Additional configuration (其他配置)。

使用完整的数据集

当您选择使用完整数据集中，您必须为数据对象提供一个清单文件。您可以提供包含清单文件的 Amazon S3 存储桶的路径，也可以使用 SageMaker 控制台创建文件。要了解如何使用控制台创建清单文件，请参阅自动数据设置 (p. 379)。

选择随机样本

当您想标记数据集的随机子集时，请选择 Random sample (随机样本)。数据集存储在中指定的 Amazon S3 存储桶中。输入数据集位置字段中返回的子位置类型。

在您指定要包含在样本中的数据对象的百分比后，立即选择创建子集. SageMaker 会为您的标记作业随机选择数据对象。选定对象后，请选择 Use this subset (使用此子集)。

SageMaker 将为选定的数据对象创建清单文件。它还会修改 Input dataset location (输入数据集位置) 字段中的值以指向新的清单文件。

指定子集

您可以使用 Amazon S3 指定数据对象的一个子集。`SELECT`查询对象文件名。

为您定义了 SQL 查询的 `SELECT` 语句。您提供 `WHERE` 子句来指定应返回哪些数据对象。

有关 Amazon S3 的更多信息`SELECT`语句，请参阅从对象中选择内容.

选择 Create subset (创建子集) 开始选择，然后选择 Use this subset (使用此子集) 来使用选择的数据。

SageMaker 将为选定的数据对象创建清单文件。它还会更新 Input dataset location (输入数据集位置) 字段中的值以指向新的清单文件。

3D 点云输入数据

要创建 3D 点云标记作业，您必须创建输入清单文件。可以使用本主题以了解每种任务类型的输入清单文件的格式要求。要了解 Ground Truth 接受的 3D 点云标记作业的原始输入数据格式，请参阅部分接受的原始 3D 数据格式 (p. 388).

可以使用标记作业任务类型在为 3D 点云标记作业创建输入清单文件 (p. 389)中选择一个主题，以了解输入清单文件的每一行的格式要求。

主题

- 接受的原始 3D 数据格式 (p. 388)
- 为 3D 点云标记作业创建输入清单文件 (p. 389)
- 了解坐标系和传感器融合 (p. 400)

接受的原始 3D 数据格式

Ground Truth 使用 3D 点云数据以渲染工作人员注释的 3D 场景。本节介绍了点云帧的点云数据和传感器融合数据接受的原始数据格式。要了解如何创建输入清单文件以将原始输入数据文件与 Ground Truth 连接起来，请参阅为 3D 点云标记作业创建输入清单文件 (p. 389).

对于每个帧，Ground Truth 支持压缩二进制包格式 (.bin) 和 ASCII (.txt) 文件。这些文件包含有关组成该帧的所有点的位置（`x`、`y` 和 `z` 坐标）的信息，以及有关彩色点云的每个点的像素颜色的信息（可选）。在创建 3D 点云标记作业输入清单文件时，您可以在 `format` 参数中指定原始数据的格式。

下表列出了在点云帧文件中支持的元素以描述各个点。

符号	值
`x`	点的 x 坐标。
`y`	点的 y 坐标。
`z`	点的 z 坐标。
`i`	点的强度。
`r`	红色通道组件。8 位值 (0-255)。
`g`	绿色通道组件。8 位值 (0-255)。
`b`	蓝色通道组件。8 位值 (0-255)。

假 Ground Truth 关输入数据的以下内容：

- 所有位置坐标 (x, y, z) 以米为单位。
- 所有姿势方向 (qx, qy, qz, qw) 以空间四元数表示。

紧凑二进制包格式

紧凑二进制包格式将点云表示为有序的点流集合。流中的每个点是有序的 4 字节浮点值二进制包，格式为 `xyzirgb`。`x`、`y` 和 `z` 元素是必需的，可以使用 `i`、`r`、`g` 和 `b` 以各种方式包含有关该像素的其他信息。

要使用二进制文件将点云帧数据输入到 Ground Truth 3D 点云标记作业中，请输入`binary/`中的`format`输入清单文件的参数并替换每个二进制包中元素的顺序。例如，您可以为 `format` 参数输入以下内容之一。

- `binary/xyzi`— 在使用该格式时，点元素流将采用以下顺序：`x1y1z1i1x2y2z2i2...`
- `binary/xyzrgb`— 在使用该格式时，点元素流将采用以下顺序：`x1y1z1r1g1b1x2y2z2r2g2b2...`
- `binary/xyzirgb`— 在使用该格式时，点元素流将采用以下顺序：`x1y1z1i1r1g1b1x2y2z2i2r2g2b2...`

在将二进制文件用于点云帧数据时，如果未输入 `format` 的值，则使用默认包格式 `binary/xyzi`。

ASCII 格式

ASCII 格式使用文本文件以表示点云，其中，ASCII 点云文件中的每一行表示一个点。每个点是文本文件中的一行，并包含以空格分隔的值，每个值是 4 字节的浮点 ASCII 值。每个点的 `x`、`y` 和 `z` 元素是必需的，可以使用 `i`、`r`、`g` 和 `b` 以各种方式包含有关该点的其他信息。

要使用文本文件将点云帧数据输入到 Ground Truth 3D 点云标记作业中，请输入`text/`中的`format`输入清单文件的参数并替换每条线上点元素的顺序。

例如，如果为 `format` 输入 `text/xyzi`，每个点云帧的文本文件应类似于以下内容：

```
x1 y1 z1 i1
x2 y2 z2 i2
...
...
```

如果输入 `text/xyzrgb`，文本文件应类似于以下内容：

```
x1 y1 z1 r1 g1 b1
x2 y2 z2 r2 g2 b1
...
...
```

在将文本文件用于点云帧数据时，如果未输入 `format` 的值，则使用默认格式 `text/xyzi`。

点云分辨率限制

3D 点云帧没有分辨率限制。不过，我们建议您将每个点云帧限制为 500K 个点以获得最佳性能。在 Ground Truth 渲染 3D 点云可视化内容时，必须在工作人员的计算机上查看该内容，这取决于工作人员的计算机硬件。超过 100 万个点的点云帧可能无法在标准计算机上渲染，或者加载时间可能太长。

为 3D 点云标记作业创建输入清单文件

在创建标记作业时，您可以提供一个输入清单文件，其中，清单的每一行描述注释者要完成的任务单元。输入清单文件格式取决于任务类型。

- 如果要创建 3D 点云对象检测或语义分割标记作业，则输入清单文件中的每一行包含有关单个 3D 点云帧的信息。这称为点云帧输入清单。要了解更多信息，请参阅 创建点云帧输入清单文件 (p. 390)。

- 如果要创建 3D 点云对象跟踪标记作业，则输入清单文件的每一行包含一个 3D 点云帧序列和关联的数据。这称为点云序列输入清单。要了解更多信息，请参阅 创建点云序列输入清单 (p. 395)。

创建点云帧输入清单文件

清单是一个 UTF-8 编码的文件，其中每行都是完整且有效的 JSON 对象。每行都以标准换行符 \n 或 \r\n 分隔。由于每行都必须是有效的 JSON 对象，因此您不能使用未转义的换行符。在单帧输入清单文件中，清单中的每一行包含单个点云帧的数据。可以使用二进制或 ASCII 格式存储点云帧数据（请参阅接受的原始 3D 数据格式 (p. 388)）。这是 3D 点云对象检测和语义分割所需的清单文件格式。（可选）您也可以为每个点云帧提供摄像机传感器融合数据。

在中支持 Ground Truth 在中支持点云和摄像机传感器融合世界坐标系 (p. 401)对于所有模式。如果您可以获取 3D 传感器外部矩阵（例如 LiDAR 外部矩阵），我们建议您使用外部矩阵将 3D 点云帧变换为世界坐标系。有关更多信息，请参阅 传感器融合 (p. 402)。

不过，如果无法在世界坐标系中获取点云，您可以在捕获数据的原始坐标系中提供坐标。如果提供用于传感器融合的摄像机数据，建议您在世界坐标系中提供 LiDAR 传感器和摄像机姿势。

要创建单帧输入清单文件，可以使用 `source-ref` 键指定您希望工作人员标记的每个点云帧的位置。此外，您必须使用 `source-ref-metadata` 键指定数据集的格式、该帧的时间戳以及传感器融合数据和视频摄像机图像（可选）。

以下示例说明了用于单帧点云标记作业的输入清单文件的语法。该示例包含两个点云帧。有关每个参数的详细信息，请参阅该示例后面的表。

> **Important**
>
> 输入清单文件中的每一行必须在JSON 行格式的日期和时间。下面的代码块显示了包含两个 JSON 对象的输入清单文件。每个 JSON 对象都用于指向单点云帧并提供有关单点云帧的详细信息。为了便于阅读，JSON 对象已经扩展，但是在创建输入清单文件时，您必须最小化每个 JSON 对象以适应一行。此代码块下提供了一个示例。

```
{
    "source-ref": "s3://awsexamplebucket/examplefolder/frame1.bin",
    "source-ref-metadata":{
        "format": "binary/xyzi",
        "unix-timestamp": 1566861644.759115,
        "ego-vehicle-pose":{
            "position": {
                "x": -2.7161461413869947,
                "y": 116.25822288149078,
                "z": 1.8348751887989483
            },
            "heading": {
                "qx": -0.02111296123795955,
                "qy": -0.006495469416730261,
                "qz": -0.008024565904865688,
                "qw": 0.9997181192298087
            }
        },
        "prefix": "s3://awsexamplebucket/lidar_singleframe_dataset/someprefix/",
        "images": [
        {
            "image-path": "images/frame300.bin_camera0.jpg",
            "unix-timestamp": 1566861644.759115,
            "fx": 847.7962624528487,
            "fy": 850.0340893791985,
            "cx": 576.2129134707038,
            "cy": 317.2423573573745,
            "k1": 0,
            "k2": 0,
            "k3": 0,
```

```
            "k4": 0,
            "p1": 0,
            "p2": 0,
            "skew": 0,
            "position": {
                "x": -2.2722515189268138,
                "y": 116.86003310568965,
                "z": 1.454614668542299
            },
            "heading": {
                "qx": 0.7594754093069037,
                "qy": 0.02181790885672969,
                "qz": -0.02461725233103356,
                "qw": -0.6496916273040025
            },
            "camera-model": "pinhole"
        }]
    }
}
{
    "source-ref": "s3://awsexamplebucket/examplefolder/frame2.bin",
    "source-ref-metadata":{
        "format": "binary/xyzi",
        "unix-timestamp": 1566861632.759133,
        "ego-vehicle-pose":{
            "position": {
                "x": -2.7161461413869947,
                "y": 116.25822288149078,
                "z": 1.8348751887989483
            },
            "heading": {
                "qx": -0.02111296123795955,
                "qy": -0.006495469416730261,
                "qz": -0.008024565904865688,
                "qw": 0.9997181192298087
            }
        },
        "prefix": "s3://awsexamplebucket/lidar_singleframe_dataset/someprefix/",
        "images": [
        {
            "image-path": "images/frame300.bin_camera0.jpg",
            "unix-timestamp": 1566861644.759115,
            "fx": 847.7962624528487,
            "fy": 850.0340893791985,
            "cx": 576.2129134707038,
            "cy": 317.2423573573745,
            "k1": 0,
            "k2": 0,
            "k3": 0,
            "k4": 0,
            "p1": 0,
            "p2": 0,
            "skew": 0,
            "position": {
                "x": -2.2722515189268138,
                "y": 116.86003310568965,
                "z": 1.454614668542299
            },
            "heading": {
                "qx": 0.7594754093069037,
                "qy": 0.02181790885672969,
                "qz": -0.02461725233103356,
                "qw": -0.6496916273040025
            },
            "camera-model": "pinhole"
        }]
```

```
    }
}
```

创建输入清单文件时，必须折叠 JSON 对象才能适应单行。例如，上面的代码块将显示在输入清单文件中如下所示：

```
{"source-ref":"s3://awsexamplebucket/examplefolder/frame1.bin","source-ref-metadata":
{"format":"binary/xyzi","unix-timestamp":1566861644.759115,"ego-vehicle-pose":{"position":
{"x":-2.7161461413869947,"y":116.25822288149078,"z":1.8348751887989483},"heading":
{"qx":-0.02111296123795955,"qy":-0.006495469416730261,"qz":-0.008024565904865688,"qw":0.999718119229808
awsexamplebucket/lidar_singleframe_dataset/someprefix/","images":
[{"image-path":"images/frame300.bin_camera0.jpg","unix-
timestamp":1566861644.759115,"fx":847.7962624528487,"fy":850.0340893791985,"cx":576.2129134707038,"cy":
{"x":-2.2722515189268138,"y":116.86003310568965,"z":1.454614668542299},"heading":
{"qx":0.7594754093069037,"qy":0.02181790885672969,"qz":-0.02461725233103356,"qw":-0.6496916273040025},"
model":"pinhole"}]}}
{"source-ref":"s3://awsexamplebucket/examplefolder/frame2.bin","source-ref-metadata":
{"format":"binary/xyzi","unix-timestamp":1566861632.759133,"ego-vehicle-pose":{"position":
{"x":-2.7161461413869947,"y":116.25822288149078,"z":1.8348751887989483},"heading":
{"qx":-0.02111296123795955,"qy":-0.006495469416730261,"qz":-0.008024565904865688,"qw":0.999718119229808
awsexamplebucket/lidar_singleframe_dataset/someprefix/","images":
[{"image-path":"images/frame300.bin_camera0.jpg","unix-
timestamp":1566861644.759115,"fx":847.7962624528487,"fy":850.0340893791985,"cx":576.2129134707038,"cy":
{"x":-2.2722515189268138,"y":116.86003310568965,"z":1.454614668542299},"heading":
{"qx":0.7594754093069037,"qy":0.02181790885672969,"qz":-0.02461725233103356,"qw":-0.6496916273040025},"
model":"pinhole"}]}}
```

下表显示了可包含在输入清单文件中的参数：

参数	必需	接受的值	描述
`source-ref`	是	字符串 接受的字符串值格式： `s3://<bucket-name>/<folder-name>/point-cloud-frame-file`	单个点云帧的 Amazon S3 位置。
`source-ref-metadata`	是	JSON 对象 接受的参数： `format`, `unix-timestamp`, `ego-vehicle-pose`, `position`, `prefix`, `images`	可以使用该参数在 `source-ref` 中包含有关点云的其他信息，并提供用于传感器融合的摄像机数据。
`format`	否	字符串 接受的字符串值："`binary/xyz`"、"`binary/xyzi`"、"`binary/xyzrgb`"、"`binary/xyzirgb`"、"`text/xyz`"、"`text/xyzi`"、"`text/`	可以使用该参数以指定点云数据的格式。有关更多信息，请参阅接受的原始 3D 数据格式 (p. 388)。

参数	必需	接受的值	描述
		`xyzrgb"、"text/xyzirgb"` 默认值： `binary/xyzi`（在 `source-ref` 中指定的文件具有 .bin 扩展名时） `text/xyzi`（在 `source-ref` 中指定的文件具有 .txt 扩展名时）	
`unix-timestamp`	是	Number Unix 时间戳。	Unix 时间戳是 1970 年 1 月 1 日到传感器收集数据的 UTC 时间之间的秒数。
`ego-vehicle-pose`	否	JSON 对象	用于收集点云数据的设备的姿势。有关该参数的更多信息，请参阅在输入清单中包含车辆姿势信息 (p. 393)。
`prefix`	否	字符串 接受的字符串值格式： `s3://<bucket-name>/<folder-name>/`	在 Amazon S3 中存储该帧的元数据（例如摄像机图像）的位置。 前缀必须以正斜杠结尾：`/`。
`images`	否	List	描述用于传感器融合的彩色摄像机图像的参数列表。您最多可以在该列表中包含 8 个图像。有关每个图像所需的参数的更多信息，请参阅在输入清单中包含摄像机数据 (p. 394)。

在输入清单中包含车辆姿势信息

可以使用自主车辆位置以提供有关用于捕获点云数据的车辆的位置的信息。Ground Truth 使用此信息来计算 LiDAR 外在矩阵。

Ground Truth 使用外部矩阵以在 3D 场景和 2D 图像之间投影标签。有关更多信息，请参阅 传感器融合 (p. 402)。

下表提供了有关在提供自主车辆信息时所需的 `position` 和方向 (`heading`) 参数的更多信息。

参数	必需	接受的值	描述
`position`	是	JSON 对象	自主车辆在世界坐标系中的平移向量。

参数	必需	接受的值	描述
		必需的参数： x, y, 和 z. 为这些参数输入数字。	
heading	是	JSON 对象 必需的参数： qx、qy、qz 和 qw。为这些参数输入数字。	安装在车辆上的设备或传感器（用于检测周围环境）的参照系的方向，在坐标系中以四元数 (qx, qy, qz, qw) 表示。

在输入清单中包含摄像机数据

如果要将视频摄像机数据与帧包含在一起，请使用以下参数以提供有关每个图像的信息。这些区域有：必需下面的列适用于images参数包含在下面的输入清单文件中source-ref-metadata. 您无需在输入清单文件中包含图像。

如果包含摄像机图像，您必须包含有关用于在世界坐标系中捕获图像的摄像机的 position 和 heading 的信息。

如果图像发生畸变，可以使用您在输入清单文件中提供的图像 Ground Truth 关信息以自动校正畸变，包括畸变系数 (畸变系数 (k1、k2、k3、k4、p1、p1)、相机模型和相机内在矩阵。内部矩阵由焦距 (fx, fy) 和主点 (cx, cy)) 组成。请参阅内部矩阵 (p. 404)以了解 Ground Truth 如何使用摄像机内部矩阵。如果不包含畸变系数 Ground Truth 则不会校正图像畸变。

参数	必需	接受的值	描述
image-path	是	字符串 格式示例： *<folder-name>/<imagefile.png>*	图像文件在 Amazon S3 中的相对位置。该相对路径将附加到您在 prefix 中指定的路径后面。
unix-timestamp	是	Number	Unix 时间戳是 1970 年 1 月 1 日到摄像机收集数据的 UTC 时间之间的秒数。
camera-model	否	字符串: 接受的值： "pinhole", "fisheye" 默认值： "pinhole"	用于捕获图像的摄像机型号。该信息用于校正摄像机图像畸变。
fx, fy	是	数字	摄像机在 x (fx) 和 y (fy) 方向上的焦距。
cx, cy	是	数字	主点的 x (cx) 和 y (cy) 坐标。

参数	必需	接受的值	描述
k1, k2, k3, k4	否	Number	径向畸变系数。鱼眼和针孔摄像机型号支持。
p1, p2	否	Number	切向畸变系数。针孔摄像机型号支持。
skew	否	Number	测量图像偏斜的参数。
position	是	JSON 对象 必需的参数： x, y, 和 z. 为这些参数输入数字。	安装在车辆上的摄像机（用于捕获图像）的参照系的位置或原点。
heading	是	JSON 对象 必需的参数： qx、qy、qz 和 qw。为这些参数输入数字。	安装在车辆上的摄像机（用于在世界坐标系中捕获图像）的参照系的方向，以四元数 (qx, qy, qz, qw) 表示。

点云帧限制

您最多可以在输入清单文件中包含 100,000 个点云帧。3D 点云标记作业的预处理时间比其他 Ground Truth 任务类型长。有关更多信息，请参阅 作业预处理时间 (p. 286)。

创建点云序列输入清单

清单是一个 UTF-8 编码的文件，其中每行都是完整且有效的 JSON 对象。每行都以标准换行符 \n 或 \r\n 分隔。由于每行都必须是有效的 JSON 对象，因此您不能使用未转义的换行符。在点云序列输入清单文件中，清单中的每一行包含一个点云帧序列。可以使用二进制或 ASCII 格式存储序列中的每个帧的点云数据。有关更多信息，请参阅 接受的原始 3D 数据格式 (p. 388)。这是 3D 点云对象跟踪所需的清单文件格式。（可选）您也可以为每个点云帧提供点属性和摄像机传感器融合数据。在创建序列输入清单文件时，您必须在世界坐标系 (p. 401)中提供 LiDAR 和视频摄像机传感器融合数据。

以下示例说明了在清单中的每一行是序列文件时用于输入清单文件的语法。输入清单文件中的每一行必须在JSON 行格式的日期和时间。

```
{"source-ref": "s3://awsexamplebucket/example-folder/seq1.json"}
{"source-ref": "s3://awsexamplebucket/example-folder/seq2.json"}
```

每个点云帧序列的数据需要存储在 JSON 数据对象中。以下是用于序列文件的格式示例。有关每个帧的信息将包含为 JSON 对象并在 frames 列表中列出。这是一个包含两个点云帧文件的序列文件的示例，frame300.bin和frame303.bin. 这些区域有：...用于指示应在何处包含其他帧的信息。为序列中的每个帧添加一个 JSON 对象。

以下代码块包括单个序列文件的 JSON 对象。为便于阅读，JSON 对象已经过扩展。

```
{
  "seq-no": 1,
  "prefix": "s3://awsexamplebucket/example_lidar_sequence_dataset/seq1/",
  "number-of-frames": 100,
  "frames":[
    {
        "frame-no": 300,
```

```
        "unix-timestamp": 1566861644.759115,
        "frame": "example_lidar_frames/frame300.bin",
        "format": "binary/xyzi",
        "ego-vehicle-pose":{
            "position": {
                "x": -2.7161461413869947,
                "y": 116.25822288149078,
                "z": 1.8348751887989483
            },
            "heading": {
                "qx": -0.02111296123795955,
                "qy": -0.006495469416730261,
                "qz": -0.008024565904865688,
                "qw": 0.9997181192298087
            }
        },
        "images": [
        {
            "image-path": "example_images/frame300.bin_camera0.jpg",
            "unix-timestamp": 1566861644.759115,
            "fx": 847.7962624528487,
            "fy": 850.0340893791985,
            "cx": 576.2129134707038,
            "cy": 317.2423573573745,
            "k1": 0,
            "k2": 0,
            "k3": 0,
            "k4": 0,
            "p1": 0,
            "p2": 0,
            "skew": 0,
            "position": {
                "x": -2.2722515189268138,
                "y": 116.86003310568965,
                "z": 1.454614668542299
            },
            "heading": {
                "qx": 0.7594754093069037,
                "qy": 0.02181790885672969,
                "qz": -0.02461725233103356,
                "qw": -0.6496916273040025
            },
            "camera-model": "pinhole"
        }]
    },
    {
        "frame-no": 303,
        "unix-timestamp": 1566861644.759115,
        "frame": "example_lidar_frames/frame303.bin",
        "format": "text/xyzi",
        "ego-vehicle-pose":{...},
        "images":[{...}]
    },
     ...
  ]
}
```

下表提供了序列文件的顶级参数的详细信息。有关序列文件中的各个帧所需的参数的详细信息，请参阅各个点云帧的参数 (p. 397)。

参数	必需	接受的值	描述
seq-no	是	整数	序列的有序编号。

参数	必需	接受的值	描述
prefix	是	字符串 接受的值： s3://<bucket-name>/<prefix>/	序列文件所在的 Amazon S3 位置。 前缀必须以正斜杠结尾：/。
number-of-frames	是	整数	序列文件中包含的总帧数。该数字必须与下一行中的 frames 参数列出的总帧数匹配。
frames	是	JSON 对象列表	帧数据列表。列表长度必须等于 number-of-frames。在工作人员 UI 中，序列中的帧与该数组中的帧顺序相同。 有关每个帧的格式的详细信息，请参阅各个点云帧的参数 (p. 397)。

各个点云帧的参数

下表显示了可包含在输入清单文件中的参数。

参数	必需	接受的值	描述
frame-no	否	整数	帧编号。这是客户指定的用于识别序列中帧的可选标识符。Ground Truth 没有使用它。
unix-timestamp	是	Number	Unix 时间戳是 1970 年 1 月 1 日到传感器收集数据的 UTC 时间之间的秒数。 每个帧的时间戳必须不同，时间戳必须是连续的，因为它们用于立方体插值。理想情况下，这应该是收集数据时的真实时间戳。如果不可用，则必须使用增量时间戳序列，其中序列文件中的第一个帧对应于序列中的第一个时间戳。
frame	是	字符串 格式示例	序列文件在 Amazon S3 中的相对位置。该相对路径将附加到您在 prefix 中指定的路径后面。

参数	必需	接受的值	描述
		<folder-name>/<sequence-file.json>	
format	否	字符串 接受的字符串值："binary/xyz"、"binary/xyzi"、"binary/xyzrgb"、"binary/xyzirgb"、"text/xyz"、"text/xyzi"、"text/xyzrgb"、"text/xyzirgb" 默认值： binary/xyzi（在 source-ref 中指定的文件具有 .bin 扩展名时） text/xyzi（在 source-ref 中指定的文件具有 .txt 扩展名时）	可以使用该参数以指定点云数据的格式。有关更多信息，请参阅接受的原始 3D 数据格式 (p. 388)。
ego-vehicle-pose	否	JSON 对象	用于收集点云数据的设备的姿势。有关该参数的更多信息，请参阅在输入清单中包含车辆姿势信息 (p. 398)。
prefix	否	字符串 接受的字符串值格式： s3://*<bucket-name>/<folder-name>/*	在 Amazon S3 中存储该帧的元数据（例如摄像机图像）的位置。 前缀必须以正斜杠结尾：/。
images	否	List	描述用于传感器融合的彩色摄像机图像的列表参数。您最多可以在该列表中包含 8 个图像。有关每个图像所需的参数的更多信息，请参阅在输入清单中包含摄像机数据 (p. 399)。

在输入清单中包含车辆姿势信息

可以使用自主车辆位置以提供有关用于捕获点云数据的车辆的姿势的信息。Ground Truth 使用此信息来计算 LiDAR 外在矩阵。

Ground Truth 使用外部矩阵以在 3D 场景和 2D 图像之间投影标签。有关更多信息，请参阅 传感器融合 (p. 402)。

下表提供了有关在提供自主车辆信息时所需的 `position` 和方向 (`heading`) 参数的更多信息。

参数	必需	接受的值	描述
`position`	是	JSON 对象 必需的参数： `x`, `y`, 和 `z`. 为这些参数输入数字。	自主车辆在世界坐标系中的平移向量。
`heading`	是	JSON 对象 必需的参数： `qx`、`qy`、`qz` 和 `qw`。为这些参数输入数字。	安装在车辆上的设备或传感器（用于检测周围环境）的参照系的方向，在坐标系中以四元数 (`qx`, `qy`, `qz`, `qw`) 表示。

在输入清单中包含摄像机数据

如果要将彩色摄像机数据与帧包含在一起，请使用以下参数以提供有关每个图像的信息。这些区域有：必需下表中的列适用于`images`输入清单文件包含参数。您无需在输入清单文件中包含图像。

如果包含摄像机图像，您必须包含有关用于捕获图像的摄像机的 `position` 和位置 (`heading`) 的信息。

如果图像发生畸变，可以使用您在输入清单文件中提供的图像 Ground Truth 关信息以自动校正畸变，包括畸变系数 (畸变系数 (`k1`、`k2`、`k3`、`k4`、`p1`、`p1`)、相机型号和焦距 (`fx`、`fy`)，以及主要点 (`cx`、`cy`). 要了解这些系数和校正图像畸变的更多信息，请参阅使用 OpenCV 校准摄像机。如果不包含畸变系数 Ground Truth 则不会校正图像畸变。

参数	必需	接受的值	描述
`image-path`	是	字符串 格式示例： *`<folder-name>/<imagefile.png>`*	图像文件在 Amazon S3 中的相对位置。该相对路径将附加到您在 `prefix` 中指定的路径后面。
`unix-timestamp`	是	Number	图像的时间戳。

`camera-model`	否	字符串:
		
		接受的值：
		
		`"pinhole"`, `"fisheye"`
		
		默认值：
		
		`"pinhole"`
			用于捕获图像的摄像机型号。该信息用于校正摄像机图像畸变。

`fx`, `fy`	是	数字	摄像机在 x (`fx`) 和 y (`fy`) 方向上的焦距。

参数	必需	接受的值	描述
`cx, cy`	是	数字	主点的 x (cx) 和 y (cy) 坐标。
`k1, k2, k3, k4`	否	Number	径向畸变系数。鱼眼和针孔摄像机型号支持。
`p1, p2`	否	Number	切向畸变系数。针孔摄像机型号支持。
`skew`	否	Number	测量图像中的任何已知偏斜的参数。
`position`	是	JSON 对象 必需的参数： `x`, `y`, 和 `z`. 为这些参数输入数字。	安装在车辆上的摄像机（用于捕获图像）的参照系的位置或原点。
`heading`	是	JSON 对象 必需的参数： `qx`、`qy`、`qz` 和 `qw`。为这些参数输入数字。	安装在车辆上的摄像机（用于捕获图像）的参照系的方向，以四元数 (`qx`, `qy`, `qz`, `qw`) 表示。

序列文件和点云帧限制

您最多可以在输入清单文件中包含 100,000 个点云帧序列。您最多可以在每个序列文件中包含 500 个点云帧。

请注意，3D 点云标记作业的预处理时间比其他 Ground Truth 任务类型长。有关更多信息，请参阅作业预处理时间 (p. 286)。

了解坐标系和传感器融合

点云数据始终位于坐标系中。该坐标系可能是检测周围环境的车辆或设备的局部坐标系，也可能是世界坐标系。在使用 Ground Truth 3D 点云标记作业时，所有注释是使用输入数据的坐标系生成的。对于某些标记作业任务类型和功能，您必须在世界坐标系中提供数据。

在本主题中，您将了解以下内容：

- 当您 时是必需的在世界坐标系或全局参照系中提供输入数据。
- 什么是世界坐标，以及如何将点云数据转换为世界坐标系？
- 在使用传感器融合时，如何使用传感器和摄像机外部矩阵提供姿势数据？

标记作业的坐标系要求

如果点云数据是在局部坐标系中收集的，您可以使用用于收集数据的传感器的外部矩阵将其转换为世界坐标系或全局参照系。如果无法为您的点云数据获取外部矩阵，因而无法在世界坐标系中获取点云，您可以在局部坐标系中为 3D 点云对象检测和语义分割任务类型提供点云数据。

对于对象跟踪，您必须在世界坐标系中提供点云数据。这是因为在多个帧中跟踪对象时，自主车辆本身正在世界中移动，因此，所有帧都需要一个参照点。

如果包含用于传感器融合的摄像机数据，建议您在与 3D 传感器（例如 LiDAR 传感器）相同的世界坐标系中提供摄像机姿势。

在世界坐标系中使用点云数据

本节说明了什么是世界坐标系 (WCS)（也称为全局参照系），并说明了如何在世界坐标系中提供点云数据。

什么是世界坐标系？

WCS 或全局参照系是固定的通用坐标系，车辆和传感器坐标系放置在该坐标系中。例如，如果从两个传感器中收集多个点云帧而导致它们位于不同的坐标系中，可以使用 WCS 将这些点云帧中的所有坐标转换为单个坐标系，其中，所有帧具有相同的原点 (0,0,0)。这种变换是通过以下方法完成的：使用平移向量将每个帧的原点平移到 WCS 的原点，并使用旋转矩阵将三个轴（通常为 x、y 和 z）旋转到正确的方向。这种刚体变换称为齐次变换。

世界坐标系对于全球路径规划、本地化、制图和驾驶场景模拟非常重要。Ground Truth 使用右手笛卡尔世界坐标系，例如ISO 8855，其中 x 轴朝向汽车的移动方向，y 轴在左侧，z 轴从地面指向上面。

全局参照系取决于数据。某些数据集将第一帧中的 LiDAR 位置作为原点。在这种情况下，所有帧将第一帧作为参照，并且设备方向和位置在第一帧的原点附近。例如，KITTI 数据集将第一帧作为世界坐标的参照。其他数据集使用不同于原点的设备位置。

请注意，这不是 GPS/IMU 坐标系，后者通常沿 z 轴旋转 90 度。如果点云数据位于 GPS/IMU 坐标系中（例如，开源 AV KITTI 数据集中的 OxTS），则需要将原点变换为世界坐标系（通常是车辆的参照坐标系）。您可以将数据乘以变换指标（旋转矩阵和平移向量）以应用这种变换。这会将数据从原始坐标系变换为全局参照坐标系。请在下一节中了解这种变换的更多信息。

将 3D 点云数据转换为 WCS

Ground Truth 假设点云数据已变换为所选的参照坐标系。例如，您可以选择传感器（例如 LiDAR）的参照坐标系以作为全局参照坐标系。您也可以从不同传感器中获取点云，并将其从传感器的视图变换为车辆的参照坐标系视图。您可以使用传感器的外部矩阵（由旋转矩阵和平移向量组成）将点云数据转换为 WCS 或全局参照系。

总而言之，平移向量和旋转矩阵可用于组成外部矩阵，该矩阵可用于将数据从局部坐标系转换为 WCS。例如，LiDAR 外部矩阵可能由以下内容组成，其中 `R` 是旋转矩阵，`T` 是平移向量：

```
LiDAR_extrinsic = [R T;0 0 0 1]
```

例如，自动驾驶 KITTI 数据集包含每个帧的 LiDAR 外部变换矩阵的旋转矩阵和平移向量。pykitti python 模块可用于加载 KITTI 数据，在数据集中，`dataset.oxts[i].T_w_imu` 为第 `i` 帧提供 LiDAR 外部变换，可以将其与该帧中的点相乘以转换为世界坐标系 (`np.matmul(lidar_transform_matrix, points)`)。通过将 LiDAR 帧中的点与 LiDAR 外部矩阵相乘，可以将其变换为世界坐标。通过将世界坐标系中的点与摄像机外部矩阵相乘，可以得出摄像机的参照系中的点坐标。

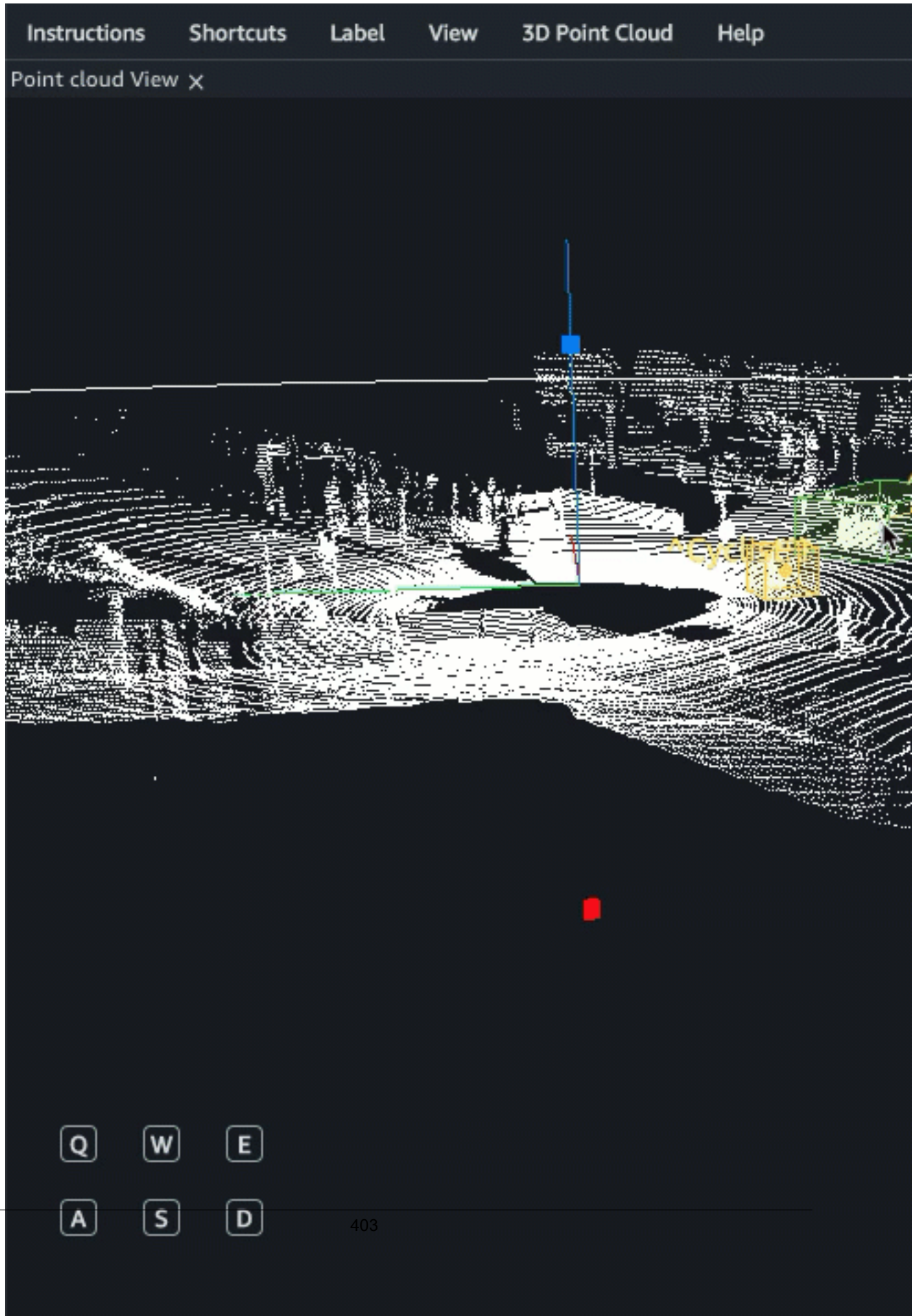
以下代码示例说明了如何将点云帧从 KITTI 数据集转换为 WCS。

```
import pykitti
import numpy as np

basedir = '/Users/nameofuser/kitti-data'
date = '2011_09_26'
drive = '0079'

# The 'frames' argument is optional - default: None, which loads the whole dataset.
# Calibration, timestamps, and IMU data are read automatically.
# Camera and velodyne data are available via properties that create generators
# when accessed, or through getter methods that provide random access.
```

```
data = pykitti.raw(basedir, date, drive, frames=range(0, 50, 5))

# i is frame number
i = 0

# lidar extrinsic for the ith frame
lidar_extrinsic_matrix = data.oxts[i].T_w_imu

# velodyne raw point cloud in lidar scanners own coordinate system
points = data.get_velo(i)

# transform points from lidar to global frame using lidar_extrinsic_matrix
def generate_transformed_pcd_from_point_cloud(points, lidar_extrinsic_matrix):
    tps = []
    for point in points:
        transformed_points = np.matmul(lidar_extrinsic_matrix, np.array([point[0],
 point[1], point[2], 1], dtype=np.float32).reshape(4,1)).tolist()
        if len(point) > 3 and point[3] is not None:
            tps.append([transformed_points[0][0], transformed_points[1][0],
 transformed_points[2][0], point[3]])

    return tps

# customer transforms points from lidar to global frame using lidar_extrinsic_matrix
transformed_pcl = generate_transformed_pcd_from_point_cloud(points, lidar_extrinsic_matrix)
```

传感器融合

支持 Ground Truth 云数据与最多 8 个摄像机输入进行传感器融合。通过使用该功能，标记人员可以并排查看 3D 点云帧和同步的视频帧。除了为标记提供更多视觉上下文以外，工作人员还可以通过传感器融合在 3D 场景和 2D 图像中调整注释，调整内容将投影到另一个视图中。以下视频说明了使用 LiDAR 和摄像机传感器融合的 3D 点云标记作业。


Instructions
Shortcuts
Label
View
3D Point Cloud
Help
Point cloud View
Q
W
E
A
S
D

为了获得最佳效果，在使用传感器融合时，点云应该位于 WCS 中。Ground Truth 获得传感器（例如 LiDAR）、摄像机和自主车辆姿势信息，以计算传感器融合的外部和内部矩阵。

外部矩阵

使 Ground Truth 传感器（例如 LiDAR）外部矩阵以及摄像机外部和内部矩阵，以便在点云数据的参照系和摄像机的参照系之间投影对象。

例如，为了将标签从 3D 点云投影到摄像机图像平面，Ground Truth 将 3D 点从 LiDAR 自己的坐标系变换为摄像机的坐标系。为了执行此操作，首先使用 LiDAR 外部矩阵将 3D 点从 LiDAR 自己的坐标系变换为世界坐标系（或全局参照系）。然后，Ground Truth 使用摄像机逆外部矩阵（将点从全局参照系变换为摄像机的参照系），以便将上一步中获得的世界坐标系的 3D 点变换为摄像机图像平面。也可以使用 LiDAR 外部矩阵以将 3D 数据变换为世界坐标系。如果 3D 数据已变换为世界坐标系，则第一次变换不会对标签平移产生任何影响，标签平移仅取决于摄像机逆外部矩阵。视图矩阵用于将投影的标签可视化。要了解这些变换和视图矩阵的更多信息，请参阅 Ground Truth 传感器融合变换 (p. 407)。

Ground Truth 使用激光雷达和摄像头计算这些外在矩阵姿势数据您提供的：`heading`（以四元数为单位：`qx`、`qy`、`qz`, 和`qw`) 和`position`(`x`、`y`、`z`)。对于车辆，通常通过世界坐标系中的车辆参照系描述方向和位置，它们称为自主车辆姿势。对于每个摄像机外部矩阵，您可以为该摄像机添加姿势信息。有关更多信息，请参阅 Pose (p. 404)。

内部矩阵

使用 Ground Truth 像机外部和内部矩阵以计算视图指标，以便在 3D 场景和摄像机图像之间变换标签。Ground Truth 使用摄像机焦距计算相机内在矩阵 (`fx`、`fy`) 和光学中心坐标 (`cx`、`cy`) 您提供的。有关更多信息，请参阅 内部矩阵和畸变 (p. 408)。

映像畸变

图像畸变可能是由于多种原因而发生的。例如，由于桶状或鱼眼效果，图像可能会被扭曲。使用内部参数以及畸变系数以校 Ground Truth 您在创建 3D 点云标记作业时提供的图像的畸变。如果已校正摄像机图像畸变，则所有畸变系数应设置为 0。

有关为校正图像畸变 Ground Truth 执行的变换的更多信息，请参阅相机校准：外部矩阵、内部矩阵和畸变 (p. 407).

自主车辆

为了收集用于自动驾驶应用程序的数据，将从安装在车辆或自主车辆 上的传感器中获取用于生成点云数据的测量值。为了在 3D 场景和 2D 图像之间投影标签调整，需要使用世界坐标系中的自主车辆姿势。自主车辆姿势由位置坐标和方向四元数组成。

使用 Ground Truth（自主车辆姿势）以计算旋转和变换矩阵。三维旋转可以表示为一系列围绕一些轴的三次旋转。理论上，跨 3D 欧几里德空间的任意三个轴就足够了。在实践中，将选择旋转轴以作为基本向量。预计这三次轮调将在全球参考框架内（外部）。Ground Truth 不支持以物体为中心的参照系（内部），该参照系附加到旋转的对象并随之旋转。为了跟踪对象，需要从所有车辆行驶时所在的全局参照中进行测量。在使用 Ground Truth 3D 点云标记作业时，z 指定旋转轴（外部旋转），偏转欧拉角以弧度（旋转角）表示。

Pose

将姿势信息用于 3D 可视化和传感器融合。您通过清单文件输入的姿势信息用于计算外部矩阵。如果您已具有外部矩阵，您可以使用该矩阵提取传感器和摄像机姿势数据。

例如，在自动驾驶 KITTI 数据集中，可以使用 pykitti python 模块以加载 KITTI 数据。在数据集中，`dataset.oxts[i].T_w_imu` 为第 `i` 帧提供 LiDAR 外部变换，可以将其与点相乘以转换为世界坐标系 (`matmul(lidar_transform_matrix, points)`)。对于输入清单文件 JSON 格式，可以将该变换转换为 LiDAR 的位置（平移向量）和方向（以四元数表示）。`inv(matmul(dataset.calib.T_cam0_velo, inv(dataset.oxts[i].T_w_imu)))` 可以计算第 `i` 帧中的 `cam0` 的摄像机外部变换，并将其转换为 `cam0` 方向和位置。

```
import numpy

rotation = [[ 9.96714314e-01, -8.09890350e-02,  1.16333982e-03],
 [ 8.09967396e-02,  9.96661051e-01, -1.03090934e-02],
 [-3.24531964e-04,  1.03694477e-02,  9.99946183e-01]]

origin= [1.71104606e+00,
          5.80000039e-01,
          9.43144935e-01]


from scipy.spatial.transform import Rotation as R

# position is the origin
position = origin
r = R.from_matrix(np.asarray(rotation))

# heading in WCS using scipy
heading = r.as_quat()
print(f"pose:{position}\nheading: {heading}")
```

位置

在输入清单文件中，`position` 指的是传感器相对于世界坐标系的位置。如果无法将设备位置放置在世界坐标系中，您可以将 LiDAR 数据与局部坐标一起使用。同样，对于安装的视频摄像机，您可以在世界坐标系中指定位置和方向。对于摄像机，如果您没有位置信息，请使用 (0, 0, 0)。

以下是 position 对象中的字段：

1. x（浮点值）自主车辆、传感器或摄像机位置的 x 坐标（以米为单位）。
2. y（浮点值）自主车辆、传感器或摄像机位置的 y 坐标（以米为单位）。
3. z（浮点值）自主车辆、传感器或摄像机位置的 z 坐标（以米为单位）。

以下是 `position` JSON 对象的示例：

```
{
    "position": {
        "y": -152.77584902657554,
        "x": 311.21505956090624,
        "z": -10.854137529636024
      }
}
```

方向

在输入清单文件中，`heading` 是一个对象，它表示设备相对于世界坐标系的方向。heading 值应以四元数表示。四元数是与测地线球形属性一致的方向的表示形式。如果您无法将传感器方向放置在世界坐标中，请使用标识四元数（`qx = 0, qy = 0, qz = 0, qw = 1`）。同样，对于摄像机，请以四元数形式指定方向。如果您无法获取外部摄像机校准参数，请也使用标识四元数。

`heading` 对象中的字段如下所示：

1. qx（浮点值）- 自主车辆、传感器或摄像机方向的 x 分量。
2. qy（浮点值）- 自主车辆、传感器或摄像机方向的 y 分量。
3. qz（浮点值）- 自主车辆、传感器或摄像机方向的 z 分量。
4. qw（浮点值）- 自主车辆、传感器或摄像机方向的 w 分量。

以下是 `heading` JSON 对象的示例：

```
{
    "heading": {
        "qy": -0.7046155108831117,
        "qx": 0.034278837280808494,
        "qz": 0.7070617895701465,
        "qw": -0.04904659893885366
      }
}
```

要了解更多信息，请参阅 计算方向四元数和位置 (p. 406)。

计算方向四元数和位置

Ground Truth 必须以四元数形式提供所有方向或方位数据。四元数是与测地线球形属性一致的方向的表示形式，可用于对旋转进行近似计算。与欧拉角相比，它们的组成更简单，并且避免了万向节死锁问题。与旋转矩阵相比，它们更紧凑，并且数值稳定性和效率更高。

您可以从旋转矩阵或变换矩阵中计算四元数。

如果在世界坐标系中具有旋转矩阵（由轴旋转组成）和平移向量（或原点），而不是单个 4x4 刚性变换矩阵，则可以直接使用旋转矩阵和平移向量以计算四元数。scipy 和 pyqaternion 之类的库可以提供帮助。以下代码块说明了使用这些库从旋转矩阵中计算四元数的示例。

```
import numpy

rotation = [[ 9.96714314e-01, -8.09890350e-02,  1.16333982e-03],
 [ 8.09967396e-02,  9.96661051e-01, -1.03090934e-02],
 [-3.24531964e-04,  1.03694477e-02,  9.99946183e-01]]

origin = [1.71104606e+00,
          5.80000039e-01,
          9.43144935e-01]


from scipy.spatial.transform import Rotation as R
# position is the origin
position = origin
r = R.from_matrix(np.asarray(rotation))
# heading in WCS using scipy
heading = r.as_quat()
print(f"position:{position}\nheading: {heading}")
```

3D 旋转转换器之类的 UI 工具也可能是非常有用的。

如果您具有 4x4 外部变换矩阵，请注意，该变换矩阵采用形式。`[R T; 0 0 0 1]`哪里`R`是旋转矩阵`T`是原点翻译向量。这意味着，您可以按以下方式从变换矩阵中提取旋转矩阵和平移向量。

```
import numpy as np

transformation
= [[ 9.96714314e-01, -8.09890350e-02,  1.16333982e-03, 1.71104606e+00],
   [ 8.09967396e-02,  9.96661051e-01, -1.03090934e-02, 5.80000039e-01],
   [-3.24531964e-04,  1.03694477e-02,  9.99946183e-01, 9.43144935e-01],
   [              0,               0,               0,              1]]

transformation  = np.array(transformation )
rotation = transformation[0:3][0:3]
translation= transformation[0:3][3]

from scipy.spatial.transform import Rotation as R
# position is the origin translation
position = translation
```

```
r = R.from_matrix(np.asarray(rotation))
# heading in WCS using scipy
heading = r.as_quat()
print(f"position:{position}\nheading: {heading}")
```

对于您自己的设置，您可以使用相对于自主车辆上的 LiDAR 传感器的 GPS/IMU 位置和方向（纬度、经度和高度以及翻滚、俯仰和偏转）以计算外部变换矩阵。例如，您可以使用 `pose = convertOxtsToPose(oxts)` 将 oxts 数据变换为局部欧几里德姿势（由 4x4 刚性变换矩阵指定），以从 KITTI 原始数据中计算姿势。然后，您可以使用世界坐标系中的参照系变换矩阵，以将该姿势变换矩阵变换为全局参照系。

```
struct Quaternion
{
    double w, x, y, z;
};

Quaternion ToQuaternion(double yaw, double pitch, double roll) // yaw (Z), pitch (Y), roll
 (X)
{
    // Abbreviations for the various angular functions
    double cy = cos(yaw * 0.5);
    double sy = sin(yaw * 0.5);
    double cp = cos(pitch * 0.5);
    double sp = sin(pitch * 0.5);
    double cr = cos(roll * 0.5);
    double sr = sin(roll * 0.5);

    Quaternion q;
    q.w = cr * cp * cy + sr * sp * sy;
    q.x = sr * cp * cy - cr * sp * sy;
    q.y = cr * sp * cy + sr * cp * sy;
    q.z = cr * cp * sy - sr * sp * cy;

    return q;
}
```

Ground Truth 传感器融合变换

以下几节更详细地介绍了使用您提供的姿势数据执行的 Ground Truth 传感器融合变换。

LiDAR 外部矩阵

为了在 3D LiDAR 场景和 2D 摄像机图像之间进行投影，Ground Truth 使用自主车辆姿势和方向计算刚性变换投影指标。执行一系列简单的旋转和平移，Ground Truth 以计算世界坐标到 3D 平面的旋转和平移。

使用方向四元数以计算旋转指标，如下所示：

$$M = \begin{pmatrix} 1-2y^2-2z^2 & 2xy+2zw & 2xz-2yw \\ 2xy-2zw & 1-2x^2-2z^2 & 2yz+2xw \\ 2xz+2yw & 2yz-2xw & 1-2x^2-2y^2 \end{pmatrix}$$

在这里，`[x, y, z, w]`与中的参数相对应`heading`JSON 对象，`[qx, qy, qz, qw]`. Ground Truth 将翻译列矢量计算为`T = [poseX, poseY, poseZ]`. 然后，外部指标就如同下面所示：

```
LiDAR_extrinsic = [R T;0 0 0 1]
```

相机校准：外部矩阵、内部矩阵和畸变

几何摄像机校准（也称为摄像机标定）估算图像或视频摄像机的镜头和图像传感器的参数。您可以使用这些参数校正镜头畸变，以世界单位测量对象的大小或确定摄像机在场景中的位置。摄像机参数包括内部和畸变系数。

摄像机外部矩阵

如果提供了摄像机姿势，则根据从 3D 平 Ground Truth 到摄像机平面的刚性变换以计算摄像机外部矩阵。该计算方法与用于LiDAR 外部矩阵 (p. 407)，除了 Ground Truth 使用相机姿势（`position`和`heading`) 并计算反向的外部信息。

```
camera_inverse_extrinsic = inv([Rc Tc;0 0 0 1]) #where Rc and Tc are camera pose
components
```

内部矩阵和畸变

某些摄像机，例如针孔或鱼眼相机，可能会导致照片显著失真。这种失真可以使用失真系数和摄像机焦距来纠正。要了解更多信息，请参阅使用 OpenCV 进行相机校准(在 OpenCV 文档中)。

Ground Truth 可以纠正两种类型的失真：径向失真和切向失真。

如果光线在镜头边缘附近的弯曲程度超过光学中心，就会发生径向畸变。镜头越小，畸变越大。径向畸变的存在以桶要么鱼眼效果和 Ground Truth 使用一级方程式来解除它。

公式 1：

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

切向畸变发生这种情况是因为，用于拍摄图像的镜头未完全平行于成像平面。可以使用公式 2 校正这种畸变。

公式 2：

$$x_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

在输入清单文件中，您可以提供畸变系数，Ground Truth 将校正图像畸变。所有畸变系数均为浮点值。

- `k1`、`k2`、`k3`、`k4`— 径向畸变系数。鱼眼和针孔摄像机型号支持。
- `p1`、`p2`— 切向畸变系数。针孔摄像机型号支持。

如果已校正图像畸变，输入清单中的所有畸变系数应为 0。

为了正确重建校正的图像，Ground Truth 根据焦距对图像进行单位转换。如果在两个轴中将相同的焦距用于给定宽高比（例如 1），在上面的公式中，我们将具有单个焦距。包含这四个参数的矩阵称为摄像机内部校准矩阵。

$$\begin{Bmatrix} x \\ y \\ w \end{Bmatrix} = \begin{Bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{Bmatrix} \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}$$

尽管畸变系数是相同的（而不管使用的摄像机分辨率如何），但应使用校准的分辨率缩放当前分辨率的系数。

以下是浮点值。

- `fx` - x 方向的焦距。

- `fy` - y 方向的焦距。
- `cx` - 主点的 x 坐标。
- `cy` - 主点的 y 坐标。

使用 Ground Truth 像机外部和内部矩阵以计算视图指标（如以下代码块中所示），以便在 3D 场景和 2D 图像之间变换标签。

```
def generate_view_matrix(intrinsic_matrix, extrinsic_matrix):
    intrinsic_matrix = np.c_[intrinsic_matrix, np.zeros(3)]
    view_matrix = np.matmul(intrinsic_matrix, extrinsic_matrix)
    view_matrix = np.insert(view_matrix, 2, np.array((0, 0, 0, 1)), 0)
    return view_matrix
```

视频帧输入数据

创建视频帧对象检测或对象跟踪标签作业时，可以选择视频文件（MP4 文件）或视频帧作为输入数据。所有工作人员任务都是使用视频帧创建的，因此如果选择视频文件，请使用 Ground Truth 帧提取工具从视频文件中提取视频帧（图像）。

对于这两个选项，您可以使用自动化数据设置Amazon SageMaker 控制台的 Ground Truth 部分中的选项，用于在 Ground Truth 和 Amazon S3 中的输入数据之间建立连接，以便 Ground Truth 知道在创建标签任务时在何处查找输入数据。这将在 Amazon S3 输入数据集位置创建并存储输入清单文件。要了解更多信息，请参阅 自动视频帧输入数据设置 (p. 411)。

或者，您可以为要标记的每个视频帧序列手动创建序列文件，并使用`source-ref`键。要了解更多信息，请参阅 创建视频帧输入清单文件 (p. 413)。

主题

- 为输入数据选择视频文件或视频帧 (p. 409)
- 输入数据设置 (p. 410)

为输入数据选择视频文件或视频帧

当您创建视频帧对象检测或对象跟踪标签作业时，您可以提供一系列视频帧（图像），也可以使用 Amazon SageMaker 控制台让 Ground Truth 自动从视频文件中提取视频帧。使用以下部分了解有关这些选项的更多信息。

提供视频帧

视频帧是从视频文件中提取的图像序列。您可以创建 Ground Truth 标签作业，让工作人员标记多个视频帧序列。每个序列都由从单个视频中提取的图像组成。

要使用视频帧序列创建标签作业，必须使用唯一的键名称前缀在 Amazon S3 中。在 Amazon S3 控制台中，密钥名称前缀是文件夹。因此，在 Amazon S3 控制台中，每个视频帧序列都必须位于 Amazon S3 中自己的文件夹中。

例如，如果您有两个视频帧序列，则可以使用键名称前缀。`sequence1/`和`sequence2/`来识别你的序列。在此示例中，您的序列可能位于`s3://DOC-EXAMPLE-BUCKET/video-frames/sequence1/`和`s3://DOC-EXAMPLE-BUCKET/video-frames/sequence2/`.

如果您使用 Ground Truth 控制台创建输入清单文件，则所有序列键名称前缀应位于 Amazon S3 中的同一位置。例如，在 Amazon S3 控制台中，每个序列可以位于`s3://DOC-EXAMPLE-BUCKET/video-frames/`. 在此示例中，您的第一个视频帧（图像）序列可能位于`s3://DOC-EXAMPLE-BUCKET/video-frames/sequence1/`你的第二个序列可能位于`s3://DOC-EXAMPLE-BUCKET/video-frames/sequence2/`.

> **Important**
>
> 即使您只有希望工作人员标记的单个视频帧序列，该序列必须在 Amazon S3 中有密钥名称前缀。如果您使用的是 Amazon S3 控制台，这意味着您的序列位于文件夹中。它不能位于 S3 存储桶的根目录中。

使用视频帧序列创建工作人员任务时，Ground Truth 每个任务使用一个序列。在每个任务中，Ground Truth 都会使用以下方式订购UTF-8二进制顺序。

例如，在 Amazon S3 中，视频帧可能按以下顺序排列：

```
[0001.jpg, 0002.jpg, 0003.jpg, ..., 0011.jpg]
```

他们在工作人员的任务中按照相同的顺序排列：`0001.jpg, 0002.jpg, 0003.jpg, ..., 0011.jpg`.

也可以使用如下命名约定对帧进行排序：

```
[frame1.jpg, frame2.jpg, ..., frame11.jpg]
```

在本例中，`frame10.jpg`和`frame11.jpg`之前来`frame2.jpg`在工作人员任务中。你的工作人员按以下顺序查看你的视频帧：`frame1.jpg, frame10.jpg, frame11.jpg, frame2.jpg, ..., frame9.jpg`.

提供视频文件

在控制台中创建新的标注作业时，可以使用 Ground Truth 帧拆分功能从视频文件（MP4 文件）中提取视频帧。从单个视频文件中提取的一系列视频帧被称为视频帧序列.

您可以让 Ground Truth 自动从视频中提取所有帧（最多 2,000 帧），也可以指定帧提取频率。例如，您可以每 10 个获取 Ground Truth 提取。th来自你的视频中的帧。

使用自动数据设置提取帧时，最多可以提供 50 个视频，但是在创建视频帧对象跟踪和视频帧对象检测标记作业时，输入清单文件不能引用 10 个以上的视频帧序列文件。如果您使用自动数据设置控制台工具从 10 个以上的视频文件中提取视频帧，则需要修改该工具生成的清单文件或创建一个新的清单文件以包含 10 个或更少的视频帧序列文件。要了解这些配额的更多信息，请参阅3D 点云和视频帧标记 Job 配额 (p. 386).

要使用视频帧提取工具，请参阅自动视频帧输入数据设置 (p. 411).

成功从视频中提取所有视频帧后，您将在 S3 输入数据集位置中看到以下内容：

- 以每个视频命名的密钥名称前缀（Amazon S3 控制台中的一个文件夹）。这些前缀中的每一个都导致：
  - 从视频中提取的视频帧序列，用于命名该前缀。
  - 用于标识组成该序列的所有图像的序列文件。
- 扩展名为 .manifest 的输入清单文件。这标识了将用于创建标签作业的所有序列文件。

从单个视频文件中提取的所有帧都用于标签任务。如果从多个视频文件中提取视频帧，则会为标签作业创建多个任务，每个视频帧序列一个任务。

Ground Truth 将每个视频帧序列存储在 Amazon S3 位置中提取的输入数据集中使用唯一的键名称前缀. 在 Amazon S3 控制台中，密钥名称前缀是文件夹。

输入数据设置

在创建视频帧标记作业时，您需要让 Ground Truth 知道在哪里查找输入数据。您可以通过两种方式之一来执行此操作：

- 您可以将输入数据存储在 Amazon S3 中，并让 Ground Truth 自动检测用于标签作业的输入数据集。请参阅自动视频帧输入数据设置 (p. 411)了解有关此选项的更多信息。

- 您可以创建输入清单文件和序列文件并将其上传到 Amazon S3。请参阅手动输入数据设置 (p. 412)了解有关此选项的更多信息。

主题

- 自动视频帧输入数据设置 (p. 411)
- 手动输入数据设置 (p. 412)

自动视频帧输入数据设置

您可以使用 Ground Truth 自动数据设置自动检测 Amazon S3 存储桶中的视频文件并从这些文件中提取视频帧。要了解如何操作，请参阅提供视频文件 (p. 410)。

如果 Amazon S3 中已有视频帧，则可以使用自动数据设置在标签作业中使用这些视频帧。对于此选项，来自单个视频的所有视频帧必须使用唯一的前缀来存储。要了解使用此选项的要求，请参阅提供视频帧 (p. 409).

选择以下部分之一，了解如何使用 Ground Truth 设置自动输入数据集连接。

提供视频文件和提取帧

使用以下过程将视频文件与 Ground Truth 连接起来，并自动从这些文件中提取视频帧以进行视频帧对象检测和对象跟踪标记作业。

> Note
>
> 如果您使用自动数据设置控制台工具从 10 个以上的视频文件中提取视频帧，则需要修改该工具生成的清单文件或创建一个新的清单文件以包含 10 个或更少的视频帧序列文件。要了解更多信息，请参阅 提供视频文件 (p. 410)。

确保您的视频文件存储在同一存储桶中的 Amazon S3 存储桶中。Amazon您在其中执行自动数据设置的区域。

使用 Ground Truth 自动连接 Amazon S3 中的视频文件并提取视频帧：

1. 导航到创建标记作业Amazon SageMaker 控制台中的页面：https://console.aws.amazon.com/sagemaker/groundtruth.

   您的输入和输出 S3 存储桶必须位于同一位置中。Amazon您在其中创建标签作业的地区。这个链接将您带到弗吉尼亚北部（us-east-1）Amazon区域。如果您的输入数据位于另一个区域的 Amazon S3 存储桶中，请切换到该区域。更改Amazon地区，在导航栏中，选择当前显示区域的名称。
2. Select创建标记作业.
3. 输入 Job name（任务名称）。
4. 在部分中输入数据设置，选择自动化数据设置.
5. 输入以下内容的 Amazon S3 URI输入数据集的 S3 位置. S3 URI 看上去与下类似：`s3://DOC-EXAMPLE-BUCKET/path-to-files/`. 此 URI 应指向存储视频文件的 Amazon S3 位置。
6. 指定您的输出数据集的 S3 位置. 这是您存放输出数据的地方。您可以选择将输出数据存储在与输入数据集相同的位置要么指定新位置然后输入要存储输出数据的位置的 S3 URI。
7. 选择视频文件为了您的数据类型使用下拉列表。
8. 选择是的，为对象跟踪和检测任务提取帧.
9. 选择帧提取.

   - 选择时使用从视频中提取的所有帧来创建标签任务，Ground Truth 从你的每个视频中提取所有帧输入数据集的 S3 位置，最多 2,000 帧。如果输入数据集中的视频包含 2,000 个以上的帧，则提取前 2,000 帧并用于该标注任务。

- 选择时使用每个*x*从视频中进行帧以创建标签任务，Ground Truth 提取每一个*x*th从你的每个视频中的帧输入数据集的 S3 位置.

  例如，如果你的视频长度为 2 秒，并且有帧率在每秒 30 帧中，视频中有 60 帧。如果你在这里指定 10，Ground Truth 每 10 次提取一次th来自视频的帧。这意味着 1st，10th，20th，30th，40th，50th，和 60th帧被提取。

10. 选择或创建 IAM 执行角色。确保此角色有权访问您的 Amazon S3 位置以获取步骤 5 和 6 中指定的输入和输出数据。
11. Select完成数据设置.

提供视频帧

使用以下过程将视频帧序列与 Ground Truth 连接起来，以进行视频帧对象检测和对象跟踪标记作业。

确保您的视频帧存储在同一存储桶中的 Amazon S3 存储桶中。Amazon您在其中执行自动数据设置的区域。每个视频帧序列都应该有唯一的前缀。例如，如果你有两个序列存储在`s3://DOC-EXAMPLE-BUCKET/video-frames/sequences/`，每个人应具有唯一的前缀，如`sequence1`和`sequence2`并且都应该直接位于`/sequences/prefix`. 在以上示例中，这两个序列的位置是：`s3://DOC-EXAMPLE-BUCKET/video-frames/sequences/sequence1/`和`s3://DOC-EXAMPLE-BUCKET/video-frames/sequences/sequence2/`.

使用 Ground Truth 自动连接 Amazon S3 中的视频帧：

1. 导航到创建标记作业在 Amazon SageMaker 控制台中的页面：https://console.aws.amazon.com/sagemaker/groundtruth.

   您的输入和输出 S3 存储桶必须位于同一位置中。Amazon您在其中创建标签作业的地区。这个链接将您带到弗吉尼亚北部（us-east-1）Amazon区域。如果您的输入数据位于另一个区域的 Amazon S3 存储桶中，请切换到该区域。更改Amazon地区，在导航栏中，选择当前显示区域的名称。
2. Select创建标记作业.
3. 输入 Job name（任务名称）。
4. 在部分中输入数据设置，选择自动化数据设置.
5. 输入以下内容的 Amazon S3 URI输入数据集的 S3 位置.

   这应是存储序列的 Amazon S3 位置。例如，如果你有两个序列存储在`s3://DOC-EXAMPLE-BUCKET/video-frames/sequences/sequence1/`、`s3://DOC-EXAMPLE-BUCKET/video-frames/sequences/sequence2/`输入`s3://DOC-EXAMPLE-BUCKET/video-frames/sequences/`在这里。
6. 指定您的输出数据集的 S3 位置. 这是您存放输出数据的地方。您可以选择将输出数据存储在与输入数据集相同的位置要么指定新位置然后输入要存储输出数据的位置的 S3 URI。
7. 选择视频帧数为了您的数据类型使用下拉列表。
8. 选择或创建 IAM 执行角色。确保此角色有权访问您的 Amazon S3 位置以获取步骤 5 和 6 中指定的输入和输出数据。
9. Select完成数据设置.

这些过程将在 Amazon S3 位置为您在步骤 5 中指定的输入数据集创建输入清单。如果要使用 SageMaker API 创建标记作业，或者，Amazon CLI，或者Amazon开发工具包，使用此输入清单文件的 Amazon S3 URI 作为参数的输入`ManifestS3Uri`.

手动输入数据设置

如果您已经为每个视频帧序列创建了序列文件，请选择手动数据设置选项，以及列出对这些序列文件的引用的清单文件。

创建视频帧输入清单文件

在创建标注任务时，Ground Truth 使用输入清单文件来标识输入数据集的位置。对于视频帧对象检测和对象跟踪标记作业，输入清单文件中的每一行都标识视频帧序列文件的位置。每个序列文件标识单个视频帧序列中包含的图像。

使用本页可以了解如何为视频帧对象跟踪和对象检测标记作业创建视频帧序列文件和输入清单文件。

如果您希望 Ground Truth 自动生成序列文件并输入清单文件，请参阅自动视频帧输入数据设置 (p. 411).

创建视频帧序列输入清单

在视频帧序列输入清单文件中，清单中的每一行是 JSON 对象，其中包含`"source-ref"`引用序列文件的键。每个序列文件都标识一系列视频帧的位置。这是所有视频帧标记作业所需的清单文件格式。

以下示例说明了用于输入清单文件的语法：

```
{"source-ref": "s3://DOC-EXAMPLE-BUCKET/example-folder/seq1.json"}
{"source-ref": "s3://DOC-EXAMPLE-BUCKET/example-folder/seq2.json"}
```

创建视频帧序列文件

每个视频帧序列的数据需要存储在 JSON 数据对象中。以下是用于序列文件的格式示例。有关每个帧的信息将包含为 JSON 对象并在 `frames` 列表中列出。为了便于阅读，下面的 JSON 已经扩展。

```
{
 "seq-no": 1,
 "prefix": "s3://mybucket/prefix/video1/",
 "number-of-frames": 3,
 "frames":[
   {"frame-no": 1, "unix-timestamp": 1566861644, "frame": "frame0001.jpg" },
   {"frame-no": 2, "unix-timestamp": 1566861644, "frame": "frame0002.jpg" },
   {"frame-no": 3, "unix-timestamp": 1566861644, "frame": "frame0003.jpg" }
 ]
}
```

下表提供了有关此代码示例中显示的参数的详细信息。

参数	必需	接受的值	描述
`seq-no`	是	整数	序列的有序编号。
`prefix`	是	字符串 接受的值： s3://*<bucket-name>*/*<prefix>*/	序列文件所在的 Amazon S3 位置。 前缀必须以正斜杠结尾：`/`。
`number-of-frames`	是	整数	序列文件中包含的总帧数。该数字必须与下一行中的 `frames` 参数列出的总帧数匹配。
`frames`	是	JSON 对象列表 必需： `frame-no`, `frame`	帧数据列表。列表长度必须等于 `number-of-frames`。在工作人员 UI 中，序列中的帧是在中

参数	必需	接受的值	描述
		可选： unix-timestamp	排序的。UTF-8二进制顺序。要了解有关此订单的更多信息，请参提供视频帧 (p. 409).
frame-no	是	整数	帧数。这将决定序列中帧的顺序。
unix-timestamp	否	整数	帧的 unix 时间戳。从 1970 年 1 月 1 日到捕获帧的 UTC 时间之间的秒数。
frame	是	字符串	视频帧图像文件的名称。

输出数据

标记作业的输出 Amazon S3 置于您在控制台中或对CreateLabelingJoboperation. 当工作人员提交一个或多个任务或任务过期时，输出数据将显示在此位置。请注意，在工作人员提交任务或任务过期后，输出数据可能需要几分钟时间才能显示在 Amazon S3 中。

输出数据文件的每一行都与清单文件完全相同，此外添加了分配给输入对象的标记的属性和值。该值的属性名称在控制台中或在对 `CreateLabelingJob` 操作的调用中定义。您不能在标签属性名称中使用 `-metadata`。如果您运行图像语义分割、3D 点云语义分割或 3D 点云对象跟踪作业，则标签属性必须以 `-ref` 结尾。对于任何其他类型的作业，属性名称不能以 `-ref` 结尾。

标记作业的输出是标签键值对的值。标签和值将用新值覆盖输入文件中任何现有的 JSON 数据。

例如，以下是一个图像分类标记作业的输出，其中输入数据文件存储在 Amazon S3 中。*AWSDOC-EXAMPLE-BUCKET*标签属性名称被定义为*sport*. 在本示例中，JSON 对象设置为便于阅读的格式；在实际输出文件中，JSON 对象位于单行上。有关数据格式的更多信息，请参阅 JSON 行。

```
{
    "source-ref": "s3://AWSDOC-EXAMPLE-BUCKET/image_example.png",
    "sport":0,
    "sport-metadata":
    {
        "class-name": "football",
        "confidence": 0.00,
        "type":"groundtruth/image-classification",
        "job-name": "identify-sport",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256"
    }
}
```

标签的值可以是任意有效的 JSON。在这种情况下，标签的值是分类表中此类的索引。其他作业类型（如边界框）具有更复杂的值。

输入清单文件中除标签属性之外的任何密钥值对在输出文件中都保持不变。您可以使用此功能将数据传递到您的应用程序。

一个标记作业的输出可以用作另一个标记作业的输入。在将标记作业连接在一起时，您可以使用此功能。例如，您可以发送一个标记作业以确定正在开展的运动。然后，您可以发送另一个使用相同数据的作业，以确定此运动是室内还是室外的。通过将第一个作业的输出数据作为第二个作业的清单，您可以将两个作业的结果整合为一个输出文件，以便应用程序可以更轻松地进行处理。

当作业正在进行时，输出数据文件将定期地写入到输出位置。这些中间文件对于清单文件中的每一行都包含一行。如果标记了一个对象，则会包含标签。如果未标记该对象，则将其写入到中间输出文件中，写入方式与清单文件完全相同。

输出目录

Ground Truth 在您的 Amazon S3 输出路径中创建几个目录。这些目录包含标记作业的结果和该作业的其他项目。标记作业顶层目录的名称与标记作业名称相同；输出目录放在此目录之下。例如，如果您将标记作业命名为。**`find-people`**，您的输出将在以下目录中：

```
s3://AWSDOC-EXAMPLE-BUCKET/find-people/activelearning
s3://AWSDOC-EXAMPLE-BUCKET/find-people/annotations
s3://AWSDOC-EXAMPLE-BUCKET/find-people/inference
s3://AWSDOC-EXAMPLE-BUCKET/find-people/manifests
s3://AWSDOC-EXAMPLE-BUCKET/find-people/training
```

每个目录包含以下输出：

Active Learning 目录

`activelearning` 目录仅在您使用自动数据标记时才出现。它包含为自动数据标记设置的输入和输出验证以及自动标记数据的输入和输出文件夹。

Annotations 目录

`annotations` 目录包含人力所做的所有注释。这些是来自各个工作人员的响应，这些响应尚未整合到数据对象的单个标签中。

`annotations` 目录中有三个子目录。

- 第一个子目录是 `worker-response`，它包含来自各个工作人员的响应。这包含用于每次迭代的一个子目录，其中反过来又包含该迭代中每个数据对象的一个子目录。每个数据对象的工作人员响应数据存储在带时间戳的 JSON 文件中，该文件包含每个工作人员为该数据对象提交的答案，如果您使用私人员工队伍，则包含有关这些工作人员的元数据。要详细了解此元数据的信息，请参阅员工元数据 (p. 416).
- 第二个子目录是 `consolidated-annotation`，它包含将当前批次中的注释合并到数据对象标签所需的信息。
- 第三个子目录是 `intermediate`，其中包含当前批次的输出清单以及任何已完成的标签。当完成每个数据对象的标签时，该文件会更新。

> **Note**
>
> 我们建议您不要使用文档中未提到的文件。

Inference 目录

`inference` 目录仅在您使用自动数据标记时才出现。此目录包含在标记数据对象时，SageMaker 批量转换使用的输入和输出文件。

Manifest 目录

`manifest` 目录包含来自标记作业的输出清单。清单目录 `output` 中有一个子目录。`output` 目录包含标记作业的输出清单文件。此文件命名为 `output.manifest`。

Training 目录

`training` 目录仅在您使用自动数据标记时才出现。此目录包含用于训练自动数据标记模型的输入和输出文件。

置信度得分

如果多个工作人员对单个任务进行注释，则您的标签来自于注释整合。Ground Truth 将计算每个标签的置信度得分。一个置信度得分是介于 0 和 1 之间的数字，表示在标签中具有多大的可信度。您可以使用置信度得分相互比较已标记的数据对象，并确定最不可信和最可信的标签。

您不应将置信度得分的值解释为绝对值，也不能比较标记作业的置信度得分。例如，如果所有置信度得分都在 0.98 和 0.998 之间，您只应在数据对象之间进行比较，而不应依赖于这些高的置信度得分。

您不能将人工标记的数据对象和自动标记的数据对象的置信度得分进行比较。人工标记的置信度得分是使用针对相应任务的注释整合函数计算得出的，而自动标记的置信度得分是使用一个纳入对象功能的模型计算得出的。两个模型通常具有不同的规模和平均置信度。

对于边界框标记作业，Ground Truth 将计算每个框的置信度得分。对于相同的标记类型（人工或自动），您可以在一个图像中或跨图像比较置信度得分。您不能跨标记作业比较置信度得分。

如果单个工作人员对任务进行注释（`NumberOfHumanWorkersPerDataObject`设置为`1`或者在控制台中，输入 1每个数据集对象的工作线程数)，置信度得分设置为`0.00`.

员工元数据

Ground Truth 提供的信息可用于跟踪任务输出数据中的个别工作人员。以下数据位于`worker-response`位于Annotations 目录 (p. 415)：

- 这些区域有：`acceptanceTime`是工作人员接受任务的时间。此日期和时间戳的格式为`YYYY-MM-DDTHH:MM:SS.mmmZ`年份 (`YYYY`)、月 (`MM`)、1 天 (`DD`)、小时 (`HH`)、分钟 (`MM`), 第二 (`SS`）和毫秒（`mmm`)。日期和时间由分隔`T`.
- 这些区域有：`submissionTime`是工作人员使用提交按钮。此日期和时间戳的格式为`YYYY-MM-DDTHH:MM:SS.mmmZ`年份 (`YYYY`)、月 (`MM`)、1 天 (`DD`)、小时 (`HH`)、分钟 (`MM`), 第二 (`SS`）和毫秒（`mmm`)。日期和时间由分隔`T`.
- `timeSpentInSeconds`报告工作人员积极处理该任务的总时间（以秒为单位）。此指标不包括工作人员暂停或休息的时间。
- `workerId` 对于每个工作人员都是唯一的。
- 如果您使用私有人力，在`workerMetadata`，您将看到以下内容。
  - 这些区域有：`identityProviderType`是用于管理私有人力的服务。
  - 这些区域有：`issuer`是与分配给此人工审核任务的工作团队关联的 Cognito 用户池或 OIDC 身份提供商 (IdP) 发行人。
  - 独一无二的`sub`标识符是指工作线程。如果您使用 Amazon Cognito 创建工作人员，您可以使用 Amazon Cognito 使用此 ID 检索有关此工作人员的详细信息（例如姓名或用户名）。要了解如何操作，请参阅管理和搜索用户账户在Amazon Cognito 开发人员指南.

以下是您在使用 Amazon Cognito 创建私人工作人员时可能看到的输出示例。这在`identityProviderType`.

```
"submissionTime": "2020-12-28T18:59:58.321Z",
"acceptanceTime": "2020-12-28T18:59:15.191Z",
"timeSpentInSeconds": 40.543,
"workerId": "a12b3cdefg4h5i67",
"workerMetadata": {
    "identityData": {
        "identityProviderType": "Cognito",
        "issuer": "https://cognito-idp.aws-region.amazonaws.com/aws-region_123456789",
        "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
    }
}
```

以下是的示例`workerMetadata`你可能会看看你是否使用自己的 OIDC IdP 来创建私人劳动力：

```
"workerMetadata": {
        "identityData": {
            "identityProviderType": "Oidc",
            "issuer": "https://example-oidc-ipd.com/adfs",
            "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
        }
}
```

要了解有关使用私有人力的更多信息，请参阅使用私有人力 (p. 467).

输出元数据

每个作业的输出包含有关分配给数据对象的标签的元数据。对于所有具有微小差异的作业，这些元素都是一样的。以下示例显示了元数据元素：

```
    "confidence": 0.00,
    "type": "groundtruth/image-classification",
    "job-name": "identify-animal-species",
    "human-annotated": "yes",
    "creation-date": "2020-10-18T22:18:13.527256"
```

元素有以下含义：

- `confidence`— Ground Truth 对于标签正确的可信度。有关更多信息，请参阅 置信度得分 (p. 416)。
- `type`— 分类作业的类型。有关作业类型，请参阅内置任务类型 (p. 351)。
- `job-name`— 创建作业时分配给作业的名称。
- `human-annotated`— 数据对象是由人工标记的，还是由自动数据标记所标记的。有关更多信息，请参阅自动数据标记 (p. 438)。
- `creation-date`— 创建标签的日期和时间。

分类作业输出

以下是一个图像分类作业和一个文本分类作业的示例输出（输出清单文件）。它们包括 Ground Truth 分配给数据对象的标签、标签的值和描述标签的元数据。

除了标准元数据元素外，分类作业的元数据还包括标签的类的文本值。有关更多信息，请参阅 图像分类算法 (p. 1620)。

以下示例中的红色斜体文本 取决于标记作业规范和输出数据。

```
{
    "source-ref":"s3://AWSDOC-EXAMPLE-BUCKET/example_image.jpg",
    "species":"0",
    "species-metadata":
    {
        "class-name": "dog",
        "confidence": 0.00,
        "type": "groundtruth/image-classification",
        "job-name": "identify-animal-species",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256"
    }
}
```

```
{
    "source":"The food was delicious",
    "mood":"1",
    "mood-metadata":
```

```
    {
        "class-name": "positive",
        "confidence": 0.8,
        "type": "groundtruth/text-classification",
        "job-name": "label-sentiment",
        "human-annotated": "yes",
        "creation-date": "2020-10-18T22:18:13.527256"
    }
}
```

多标签分类作业输出

以下是多标签图像分类作业和多标签文本分类作业的示例输出清单文件。它们包括 Ground Truth 分配给数据对象的标签（例如，图像或文本片段）以及用于描述工作人员在完成标记任务时看到的标签的元数据。

label 属性名称参数（例如，image-label-attribute-name) 包含由完成此任务的至少一个工作人员选择的所有标签的数组。此数组包含与 class-map 中找到的标签对应的整数键（例如 [1,0,8]）。在多标签图像分类示例中，至少一个完成图像标记任务的工作人员已为图像 exampleimage.jpg 选择 bicycle、person 和 clothing。

这些区域有：confidence-map显示了分配给由工作人员选择的每个标签的置信度得分。要了解有关 Ground Truth 置信度得分的更多信息置信度得分 (p. 416).

以下示例中的红色斜体文本 取决于标记作业规范和输出数据。

以下是多标签图像分类输出清单文件的示例。

```
{
    "source-ref": "s3://AWSDOC-EXAMPLE-BUCKET/example_image.jpg",
    "image-label-attribute-name":[1,0,8],
    "image-label-attribute-name-metadata":
       {
        "job-name":"labeling-job/image-label-attribute-name",
        "class-map":
            {
                "1":"bicycle","0":"person","8":"clothing"
            },
        "human-annotated":"yes",
        "creation-date":"2020-02-27T21:36:25.000201",
        "confidence-map":
            {
                "1":0.95,"0":0.77,"8":0.2
            },
        "type":"groundtruth/image-classification-multilabel"
        }
}
```

以下是多标签文本分类输出清单文件的示例。在该示例中，approving、sad 和 critical 是由完成位于 AWSDOC-EXAMPLE-BUCKET 中的 exampletext.txt 对象的标记任务的至少一个工作人员选择的。

```
{
    "source-ref": "AWSDOC-EXAMPLE-BUCKET/text_file.txt",
    "text-label-attribute-name":[1,0,4],
    "text-label-attribute-name-metadata":
       {
        "job-name":"labeling-job/text-label-attribute-name",
        "class-map":
            {
                "1":"approving","0":"sad","4":"critical"
            },
        "human-annotated":"yes",
        "creation-date":"2020-02-20T21:36:25.000201",
```

```
        "confidence-map":
            {
                "1":0.95,"0":0.77,"4":0.2
            },
        "type":"groundtruth/text-classification-multilabel"
        }
}
```

边界框作业输出

以下是边界框作业中的示例输出（输出清单文件）。对于此任务，返回三个边界框。标签值包含有关图像大小和边界框位置的信息。

`class_id` 元素是框的类在该任务的可用类列表中的索引。`class-map` 元数据元素包含类的文本。

对于每个边界框，元数据有一个单独的置信度得分。元数据还包含 `class-map` 元素，该元素将 `class_id` 映射到类的文本值。有关更多信息，请参阅 对象检测算法 (p. 1691)。

以下示例中的红色斜体文本 取决于标记作业规范和输出数据。

```
{
    "source-ref": "s3://AWSDOC-EXAMPLE-BUCKET/example_image.png",
    "bounding-box-attribute-name":
    {
        "image_size": [{ "width": 500, "height": 400, "depth":3}],
        "annotations":
        [
            {"class_id": 0, "left": 111, "top": 134,
                    "width": 61, "height": 128},
            {"class_id": 5, "left": 161, "top": 250,
                     "width": 30, "height": 30},
            {"class_id": 5, "left": 20, "top": 20,
                     "width": 30, "height": 30}
        ]
    },
    "bounding-box-attribute-name-metadata":
    {
        "objects":
        [
            {"confidence": 0.8},
            {"confidence": 0.9},
            {"confidence": 0.9}
        ],
        "class-map":
        {
            "0": "dog",
            "5": "bone"
        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256",
        "job-name": "identify-dogs-and-toys"
    }
 }
```

边界框调整作业的输出如以下 JSON 所示。请注意，原始 JSON 保持不变并列出了两个新作业，每个作业都在原始属性名称前加上“adjust-”。

```
{
    "source-ref": "S3 bucket location",
    "bounding-box-attribute-name":
    {
        "image_size": [{ "width": 500, "height": 400, "depth":3}],
```

```
        "annotations":
        [
            {"class_id": 0, "left": 111, "top": 134,
                    "width": 61, "height": 128},
            {"class_id": 5, "left": 161, "top": 250,
                     "width": 30, "height": 30},
            {"class_id": 5, "left": 20, "top": 20,
                     "width": 30, "height": 30}
        ]
    },
    "bounding-box-attribute-name-metadata":
    {
        "objects":
        [
            {"confidence": 0.8},
            {"confidence": 0.9},
            {"confidence": 0.9}
        ],
        "class-map":
        {
            "0": "dog",
            "5": "bone"
        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256",
        "job-name": "identify-dogs-and-toys"
    },
    "adjusted-bounding-box":
    {
        "image_size": [{ "width": 500, "height": 400, "depth":3}],
        "annotations":
        [
            {"class_id": 0, "left": 110, "top": 135,
                    "width": 61, "height": 128},
            {"class_id": 5, "left": 161, "top": 250,
                     "width": 30, "height": 30},
            {"class_id": 5, "left": 10, "top": 10,
                     "width": 30, "height": 30}
        ]
    },
    "adjusted-bounding-box-metadata":
    {
        "objects":
        [
            {"confidence": 0.8},
            {"confidence": 0.9},
            {"confidence": 0.9}
        ],
        "class-map":
        {
            "0": "dog",
            "5": "bone"
        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2018-11-20T22:18:13.527256",
        "job-name": "adjust-bounding-boxes-on-dogs-and-toys",
        "adjustment-status": "adjusted"
    }
}
```

在此输出中，作业的 type 不会更改，但会添加一个 adjustment-status 字段。此字段的值为 adjusted 或 unadjusted。如果多个工作人员已查看了对象，并且至少有一个工作人员调整了标签，则状态为 adjusted。

命名实体识别

以下是命名实体识别 (NER) 标记任务中的输出清单文件示例。对于这个任务，七entities返回。

在输出清单中，JSON 对象annotations，包括labels您提供的（标签类别）。

工作人员的响应在名为的列表中entities. 此列表中的每个实体都是一个 JSON 对象，其中包含label与中的一个匹配的值labels列出，整数startOffset标记 span 的起始 Unicode 偏移量的值和整数endOffset结束 Unicode 偏移量的值。

对于每个实体，元数据有一个单独的置信度得分。如果单个工作人员标记了每个数据对象，则每个实体的置信度值将为零。

以下示例中的红色斜体文本取决于标记作业输入和工作人员响应。

```
{
    "source": "Amazon SageMaker is a cloud machine-learning platform that was launched
 in November 2017. SageMaker enables developers to create, train, and deploy machine-
learning (ML) models in the cloud. SageMaker also enables developers to deploy ML models on
 embedded systems and edge-devices",
    "ner-labeling-job-attribute-name": {
        "annotations": {
            "labels": [
                {
                    "label": "Date",
                    "shortDisplayName": "dt"
                },
                {
                    "label": "Verb",
                    "shortDisplayName": "vb"
                },
                {
                    "label": "Thing",
                    "shortDisplayName": "tng"
                },
                {
                    "label": "People",
                    "shortDisplayName": "ppl"
                }
            ],
            "entities": [
                {
                    "label": "Thing",
                    "startOffset": 22,
                    "endOffset": 53
                },
                {
                    "label": "Thing",
                    "startOffset": 269,
                    "endOffset": 281
                },
                {
                    "label": "Verb",
                    "startOffset": 63,
                    "endOffset": 71
                },
                {
                    "label": "Verb",
                    "startOffset": 228,
                    "endOffset": 234
                },
                {
                    "label": "Date",
                    "startOffset": 75,
                    "endOffset": 88
```

```
                },
                {
                    "label": "People",
                    "startOffset": 108,
                    "endOffset": 118
                },
                {
                    "label": "People",
                    "startOffset": 214,
                    "endOffset": 224
                }
            ]
        }
    },
    "ner-labeling-job-attribute-name-metadata": {
        "job-name": "labeling-job/example-ner-labeling-job",
        "type": "groundtruth/text-span",
        "creation-date": "2020-10-29T00:40:39.398470",
        "human-annotated": "yes",
        "entities": [
            {
                "confidence": 0
            },
            {
                "confidence": 0
            },
            {
                "confidence": 0
            },
            {
                "confidence": 0
            },
            {
                "confidence": 0
            },
            {
                "confidence": 0
            },
            {
                "confidence": 0
            }
        ]
    }
}
```

标签验证作业输出

边界框验证作业的输出（输出清单文件）看起来与边界框注释作业的输出不相同。这是由于工作人员有不同类型的任务。他们并不标记对象，而是评估先前标记的准确性，做出判断，然后提供判断结果，或许还有一些评论。

如果人工正在验证或调整先前的边界框标签，则验证作业的输出将类似于以下 JSON。以下示例中的红色斜体文本 取决于标记作业规范和输出数据。

```
{
    "source-ref":"s3://AWSDOC-EXAMPLE-BUCKET/image_example.png",
    "bounding-box-attribute-name":
    {
        "image_size": [{ "width": 500, "height": 400, "depth":3}],
        "annotations":
        [
            {"class_id": 0, "left": 111, "top": 134,
                    "width": 61, "height": 128},
            {"class_id": 5, "left": 161, "top": 250,
```

```
                    "width": 30, "height": 30},
                {"class_id": 5, "left": 20, "top": 20,
                    "width": 30, "height": 30}
            ]
        },
        "bounding-box-attribute-name-metadata":
        {
            "objects":
            [
                {"confidence": 0.8},
                {"confidence": 0.9},
                {"confidence": 0.9}
            ],
            "class-map":
            {
                "0": "dog",
                "5": "bone"
            },
            "type": "groundtruth/object-detection",
            "human-annotated": "yes",
            "creation-date": "2018-10-18T22:18:13.527256",
            "job-name": "identify-dogs-and-toys"
        },
        "verify-bounding-box-attribute-name":"1",
        "verify-bounding-box-attribute-name-metadata":
        {
            "class-name": "bad",
            "confidence": 0.93,
            "type": "groundtruth/label-verification",
            "job-name": "verify-bounding-boxes",
            "human-annotated": "yes",
            "creation-date": "2018-11-20T22:18:13.527256",
            "worker-feedback": [
                {"comment": "The bounding box on the bird is too wide on the right side."},
                {"comment": "The bird on the upper right is not labeled."}
            ]
        }
}
```

虽然原始边界框输出上的 `type` 是 `groundtruth/object-detection`，但新的 `type` 是 `groundtruth/label-verification`。另请注意，`worker-feedback` 数组提供工作人员注释。如果工作人员不提供注释，则合并期间将排除空字段。

语义分割作业输出

以下是语义分割标记作业中的输出清单文件。此作业的标签的值是对 Amazon S3 存储桶中一个 PNG 文件的引用。

除了标准元素以外，标签的元数据还包括一个颜色映射，旨在定义用于标记图像的颜色、与该颜色关联的类名称以及每种颜色的置信度得分。有关更多信息，请参阅 语义分割算法 (p. 1711)。

以下示例中的红色斜体文本 取决于标记作业规范和输出数据。

```
{
    "source-ref": "s3://AWSDOC-EXAMPLE-BUCKET/example_city_image.png",
    "city-streets-ref": "S3 bucket location",
    "city-streets-ref-metadata": {
      "internal-color-map": {
        "0": {
           "class-name": "BACKGROUND",
           "confidence": 0.9,
           "hex-color": "#ffffff"
        },
        "1": {
```

```
            "class-name": "buildings",
            "confidence": 0.9,
            "hex-color": "#2acf59"
         },
         "2":  {
            "class-name": "road",
            "confidence": 0.9,
            "hex-color": "#f28333"
        }
      },
      "type": "groundtruth/semantic-segmentation",
      "human-annotated": "yes",
      "creation-date": "2018-10-18T22:18:13.527256",
      "job-name": "label-city-streets",
      },
      "verify-city-streets-ref":"1",
      "verify-city-streets-ref-metadata":
      {
         "class-name": "bad",
         "confidence": 0.93,
         "type": "groundtruth/label-verification",
         "job-name": "verify-city-streets",
         "human-annotated": "yes",
         "creation-date": "2018-11-20T22:18:13.527256",
         "worker-feedback": [
             {"comment": "The mask on the leftmost building is assigned the wrong side of
 the road."},
             {"comment": "The curb of the road is not labeled but the instructions say
 otherwise."}
         ]
      }
}
```

置信度是按每个图像评分的。一个图像中所有分类的置信度得分都将相同。

语义分割调整作业的输出类似于以下 JSON。

```
{
    "source-ref": "s3://AWSDOC-EXAMPLE-BUCKET/example_city_image.png",
    "city-streets-ref": "S3 bucket location",
    "city-streets-ref-metadata": {
      "internal-color-map": {
        "0": {
            "class-name": "BACKGROUND",
            "confidence": 0.9,
            "hex-color": "#ffffff"
        },
        "1": {
            "class-name": "buildings",
            "confidence": 0.9,
            "hex-color": "#2acf59"
        },
        "2":  {
            "class-name": "road",
            "confidence": 0.9,
            "hex-color": "#f28333"
        }
      },
      "type": "groundtruth/semantic-segmentation",
      "human-annotated": "yes",
      "creation-date": "2018-10-18T22:18:13.527256",
      "job-name": "label-city-streets",
      },
      "adjusted-city-streets-ref": "s3://AWSDOC-EXAMPLE-BUCKET/example_city_image.png",
      "adjusted-city-streets-ref-metadata": {
```

```
      "internal-color-map": {
        "0": {
           "class-name": "BACKGROUND",
           "confidence": 0.9,
           "hex-color": "#ffffff"
        },
        "1": {
           "class-name": "buildings",
           "confidence": 0.9,
           "hex-color": "#2acf59"
        },
        "2":  {
           "class-name": "road",
           "confidence": 0.9,
           "hex-color": "#f28333"
       }
     },
     "type": "groundtruth/semantic-segmentation",
     "human-annotated": "yes",
     "creation-date": "2018-11-20T22:18:13.527256",
     "job-name": "adjust-label-city-streets",
     }
}
```

视频帧物体检测输出

以下是视频帧对象检测标记作业中的输出清单文件。以下示例中的###### 取决于标记作业规范和输出数据。

除了标准元素以外，元数据还包含一个类映射，该映射列出每个在序列中至少具有一个标签的类。元数据还包括job-name这是您分配给标签作业的名称。对于调整任务，如果修改了一个或多个边界框，则有一个adjustment-status审核工作流的元数据中设置为adjusted.

```
{
    "source-ref": "s3://DOC-EXAMPLE-BUCKET/example-path/input-manifest.json",
    "CarObjectDetection-ref": "s3://AWSDOC-EXAMPLE-BUCKET/output/labeling-job-name/
annotations/consolidated-annotation/output/0/SeqLabel.json",
    "CarObjectDetection-ref-metadata": {
        "class-map": {
            "0": "car",
            "1": "bus"
        },
        "job-name": "labeling-job/labeling-job-name",
        "human-annotated": "yes",
        "creation-date": "2021-09-29T05:50:35.566000",
        "type": "groundtruth/video-object-detection"
        }
}
```

Ground Truth 为标记的每个视频帧序列创建一个输出序列文件。每个输出序列文件包含以下内容：

- 中序列中所有帧的所有注释detection-annotationsJSON 对象列表。
- 对于由工作程序进行注释的每个帧，帧文件名称 (frame)、数字 (frame-no)，包含注释的 JSON 对象的列表 (annotations)，如果适用，frame-attributes. 此列表的名称由您使用的任务类型定义：polylines、polygons、keypoints，对于边界框，annotations.

  每个 JSON 对象都包含有关单个注释和关联标签的信息。下表概述了您将为每种视频帧任务类型看到的参数。

任务类型	参数
Bounding Box	箱的尺寸：`height`和`width` 框顶、左角像素位置：`top`和`left`
Keypoint	关键点顶点：`{ "x": int, "y": int }`
面	多边形顶点列表：`vertices` 多边形顶点：`{ "x": int, "y": int }` 多边形是封闭的形状，因此第一个点也将代表最后一个点。
多边形	折线顶点列表：`vertices` 折线顶点：`{ "x": int, "y": int }`

除了特定于任务类型的值之外，您还会在每个 JSON 对象中看到以下内容：

- 任何的价值`label-category-attributes`这是为该标签指定的。
- 这些区域有：`class-id`箱的内容。使用`class-map`在输出清单文件中查看此 ID 映射到的标签类别。

以下是一个示例。`SeqLabel.json`来自边界框视频帧对象检测标签作业的文件。此文件将位于`s3://`*`your-output-bucket`*`/`*`output-prefix`*`/annotations/consolidated-annotation/output/`*`annotation-number`*`/`

```
{
    "detection-annotations": [
        {
            "annotations": [
                {
                    "height": 41,
                    "width": 53,
                    "top": 152,
                    "left": 339,
                    "class-id": "1",
                    "label-category-attributes": {
                        "occluded": "no",
                        "size": "medium"
                    }
                },
                {
                    "height": 24,
                    "width": 37,
                    "top": 148,
                    "left": 183,
                    "class-id": "0",
                    "label-category-attributes": {
                        "occluded": "no",
                    }
                }
            ],
            "frame-no": "0",
            "frame": "frame_0000.jpeg",
            "frame-attributes": {name: value, name: value}
        },
        {
            "annotations": [
                {
```

```
                    "height": 41,
                    "width": 53,
                    "top": 152,
                    "left": 341,
                    "class-id": "0",
                    "label-category-attributes": {}
                },
                {
                    "height": 24,
                    "width": 37,
                    "top": 141,
                    "left": 177,
                    "class-id": "0",
                    "label-category-attributes": {
                        "occluded": "no",
                    }
                }
            ],
            "frame-no": "1",
            "frame": "frame_0001.jpeg",
            "frame-attributes": {name: value, name: value}
        }
    ]
}
```

视频帧对象跟踪输出

以下是视频帧对象跟踪标记作业中的输出清单文件。以下示例中的###### 取决于标记作业规范和输出数据。

除了标准元素以外，元数据还包含一个类映射，该映射列出每个在帧序列中至少具有一个标签的类。元数据还包括job-name这是您分配给标签作业的名称。对于调整任务，如果修改了一个或多个边界框，则有一个adjustment-status审核工作流的元数据中设置为adjusted.

```
{
    "source-ref": "s3://DOC-EXAMPLE-BUCKET/example-path/input-manifest.json",
    "CarObjectTracking-ref": "s3://AWSDOC-EXAMPLE-BUCKET/output/labeling-job-name/
annotations/consolidated-annotation/output/0/SeqLabel.json",
    "CarObjectTracking-ref-metadata": {
        "class-map": {
            "0": "car",
            "1": "bus"
        },
        "job-name": "labeling-job/labeling-job-name",
        "human-annotated": "yes",
        "creation-date": "2021-09-29T05:50:35.566000",
        "type": "groundtruth/video-object-tracking"
        }
 }
```

Ground Truth 为标记的每个视频帧序列创建一个输出序列文件。每个输出序列文件包含以下内容：

- 中序列中所有帧的所有注释tracking-annotationsJSON 对象列表。
- 对于由工作人员注释的每个帧，该框架 (frame)、数字 (frame-no)，包含注释的 JSON 对象的列表（annotations），如果适用，框架属性（frame-attributes)。此列表的名称由您使用的任务类型定义：polylines、polygons、keypoints，对于边界框，annotations.

  每个 JSON 对象都包含有关单个注释和关联标签的信息。下表概述了您将为每种视频帧任务类型看到的参数。

任务类型	参数
Bounding Box	箱的尺寸：`height`和`width` 框顶、左角像素位置：`top`和`left`
Keypoint	关键点顶点：`{ "x": int, "y": int }`
面	多边形顶点列表：`vertices` 多边形顶点：`{ "x": int, "y": int }` 多边形是封闭的形状，因此第一个点也将代表最后一个点。
多边形	折线顶点列表：`vertices` 折线顶点：`{ "x": int, "y": int }`

除了特定于任务类型的值之外，您还会在每个 JSON 对象中看到以下内容：

- 任何的价值`label-category-attributes`这是为该标签指定的。
- 这些区域有：`class-id`箱的内容。使用`class-map`在输出清单文件中查看此 ID 映射到的标签类别。
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`object-id`它标识标签的实例。如果工作人员在多个帧中识别了对象的同一实例，则跨帧的 ID 将是相同的。例如，如果一辆汽车出现在多个帧中，则所有边框用来识别该车的所有边框都将具有相同的`object-id`.
- 这些区域有：`object-name`这是该注释的实例 ID。

以下是一个示例。`SeqLabel.json`文件来自边界框视频帧对象跟踪标记作业。此文件将位于`s3://`*`your-output-bucket`*`/`*`output-prefix`*`/annotations/consolidated-annotation/output/`*`annotation-number`*`/`

```
{
    "tracking-annotations": [
        {
            "annotations": [
                {
                    "height": 36,
                    "width": 46,
                    "top": 178,
                    "left": 315,
                    "class-id": "0",
                    "label-category-attributes": {
                        "occluded": "no"
                    },
                    "object-id": "480dc450-c0ca-11ea-961f-a9b1c5c97972",
                    "object-name": "car:1"
                }
            ],
            "frame-no": "0",
            "frame": "frame_0001.jpeg",
            "frame-attributes": {}
        },
        {
            "annotations": [
                {
                    "height": 30,
                    "width": 47,
                    "top": 163,
                    "left": 344,
```

```
                    "class-id": "1",
                    "label-category-attributes": {
                        "occluded": "no",
                        "size": "medium"
                    },
                    "object-id": "98f2b0b0-c0ca-11ea-961f-a9b1c5c97972",
                    "object-name": "bus:1"
                },
                {
                    "height": 28,
                    "width": 33,
                    "top": 150,
                    "left": 192,
                    "class-id": "0",
                    "label-category-attributes": {
                        "occluded": "partially"
                    },
                    "object-id": "480dc450-c0ca-11ea-961f-a9b1c5c97972",
                    "object-name": "car:1"
                }
            ],
            "frame-no": "1",
            "frame": "frame_0002.jpeg",
            "frame-attributes": {name: value, name: value}
        }
    ]
}
```

3D 点云语义分割输出

以下是 3D 点云语义分割标记作业中的输出清单文件。

除了标准元素以外，标签的元数据还包括一个颜色映射，旨在定义用于标记图像的颜色、与该颜色关联的类名称以及每种颜色的置信度得分。此外，还有一个adjustment-status审核工作流的元数据中设置为adjusted是否修改了颜色蒙版。如果你添加了一个或多个frameAttributes对于标签类别配置文件，工作人员对帧属性的响应位于 JSON 对象中，dataset-object-attributes.

your-label-attribute-ref 参数包含具有 .zlib 扩展名的压缩文件的位置。解压缩此文件时，它包含一个数组。数组中的每个索引对应于输入点云中注释点的索引。在给定索引处，数组的值根据位于color-map的参数metadata.

您可以使用类似于以下内容的 Python 代码来解压缩 .zlib 文件：

```
import zlib
from array import array

# read the label file
compressed_binary_file = open(zlib_file_path/file.zlib, 'rb').read()

# uncompress the label file
binary_content = zlib.decompress(compressed_binary_file)

# load labels to an array
my_int_array_data = array('B', binary_content);

print(my_int_array_data)
```

上面的代码块将生成类似于以下内容的输出。打印数组的每个元素都包含点云中该索引处的点的类。例如，my_int_array_data[0] = 1意味着point[0]在输入点云中有一个类1. 在以下输出清单文件示例中，class0对应于"Background"、1和Car, 和2和Pedestrian.

```
>> array('B', [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

以下是语义分割 3D 点云标记作业输出清单文件的示例。以下示例中的红色斜体文本 取决于标记作业规范和输出数据。

```
{
   "source-ref": "s3://AWSDOC-EXAMPLE-BUCKET/examplefolder/frame1.bin",
   "source-ref-metadata":{
      "format": "binary/xyzi",
      "unix-timestamp": 1566861644.759115,
      "ego-vehicle-pose":{...},
      "prefix": "s3://AWSDOC-EXAMPLE-BUCKET/lidar_singleframe_dataset/prefix",
      "images": [{...}]
    },
    "lidar-ss-label-attribute-ref": "s3://your-output-bucket/labeling-job-name/annotations/
consolidated-annotation/output/dataset-object-id/filename.zlib",
    "lidar-ss-label-attribute-ref-metadata": {
        'color-map': {
            "0": {
                "class-name": "Background",
                "hex-color": "#ffffff",
                "confidence": 0.00
            },
            "1": {
                "class-name": "Car",
                "hex-color": "#2ca02c",
                "confidence": 0.00
            },
            "2": {
                "class-name": "Pedestrian",
                "hex-color": "#1f77b4",
                "confidence": 0.00
            },
            "3": {
                "class-name": "Tree",
                "hex-color": "#ff7f0e",
                "confidence": 0.00
            }
        },
        'type': 'groundtruth/point_cloud_single_frame_semantic_segmentation',
        'human-annotated': 'yes',
        'creation-date': '2019-11-12T01:18:14.271944',
        'job-name': 'labeling-job-name',
        //only present for adjustment audit workflow
        "adjustment-status": "adjusted", // "adjusted" means the label was adjusted
        "dataset-object-attributes": {name: value, name: value}
    }
}
```

3D 点云对象检测输出

以下是 3D 点云对象检测作业的示例输出。对于该任务类型，有关 3D 长方体的数据是在名为 `annotations` 的列表上的 `3d-bounding-box` 参数中返回的。在该列表中，每个 3D 长方体是使用以下信息描述的。

- 您在输入清单中指定的每个类或标签类别与关联`class-id`. 可以使用 `class-map` 标识与每个类 ID 关联的类。
- 这些类用于为每个 3D 长方体提供一个采用 `<class>:<integer>` 格式的 `object-name`，其中 `integer` 是用于在帧中标识该长方体的唯一数字。
- `center-x`、`center-y`, 和`center-z`是立方体中心的坐标，与标注作业中使用的 3D 点云输入数据位于同一坐标系中。

- length、width, 和height描述长方体的尺寸。
- yaw用于描述长方体的方向（方位），以弧度为单位。

  这些区域有：yaw输出数据中的测量值为 180 度，或以弧度为单位的 pi 值，减yaw向下看方体时，在右手的世界坐标系中。换句话说，从自上而下看一个立方体时，yaw_in_output_data是顺时针正数的（与右手世界坐标系相反，其中自上而下的视图与逆时针正旋转相关联）。当从长方体上看时，yaw_in_output_data逆时针为阳性。

  转换yaw_in_output_data对于右手世界坐标系的更常见的方向，请使用以下内容（所有单位均以弧度为单位）：

```
yaw_right_handed_cartesian_system = pi - yaw_in_output_data
```

- 如果您创建了 3D 点云调整标记作业并包含在内的作业pitch和roll在输入清单文件中，相同pitch和roll测量值将显示在输出清单文件中。否则为pitch和role将始终为 0。
- 如果在输入清单文件中包含给定类的标签属性，则label-category-attributes对于工作人员选择了标签属性的所有立方体，都包括参数。

在审核工作流的元数据中具有 adjustment-status 参数；如果修改了一个或多个长方体，该参数将设置为 adjusted。如果你添加了一个或多个frameAttributes对于标签类别配置文件，工作人员对帧属性的响应位于 JSON 对象中，dataset-object-attributes.

以下示例中的###### 取决于标记作业规范和输出数据。省略号 (...) 表示该列表的延续，其中可能显示与前面对象格式相同的其他对象。

```
{
   "source-ref": "s3://AWSDOC-EXAMPLE-BUCKET/examplefolder/frame1.txt",
   "source-ref-metadata":{
      "format": "text/xyzi",
      "unix-timestamp": 1566861644.759115,
      "prefix": "s3://AWSDOC-EXAMPLE-BUCKET/lidar_singleframe_dataset/prefix",
      "ego-vehicle-pose": {
            "heading": {
                "qx": -0.02111296123795955,
                "qy": -0.006495469416730261,
                "qz": -0.008024565904865688,
                "qw": 0.9997181192298087
            },
            "position": {
                "x": -2.7161461413869947,
                "y": 116.25822288149078,
                "z": 1.8348751887989483
            }
       },
       "images": [
            {
                "fx": 847.7962624528487,
                "fy": 850.0340893791985,
                "cx": 576.2129134707038,
                "cy": 317.2423573573745,
                "k1": 0,
                "k2": 0,
                "k3": 0,
                "k4": 0,
                "p1": 0,
                "p2": 0,
                "skew": 0,
                "unix-timestamp": 1566861644.759115,
                "image-path": "images/frame_0_camera_0.jpg",
                "position": {
                    "x": -2.2722515189268138,
```

```
                    "y": 116.86003310568965,
                    "z": 1.454614668542299
                },
                "heading": {
                    "qx": 0.7594754093069037,
                    "qy": 0.02181790885672969,
                    "qz": -0.02461725233103356,
                    "qw": -0.6496916273040025
                },
                "camera_model": "pinhole"
            }
        ]
    },
   "3d-bounding-box":
    {
       "annotations": [
            {
                "label-category-attributes": {
                    "Occlusion": "Partial",
                    "Type": "Sedan"
                },
                "object-name": "Car:1",
                "class-id": 0,
                "center-x": -2.616382013657516,
                "center-y": 125.04149850484193,
                "center-z": 0.311272296465834,
                "length": 2.993000265181146,
                "width": 1.8355260519692056,
                "height": 1.3233490884304047,
                "roll": 0,
                "pitch": 0,
                "yaw": 1.6479308313703527
            },
            {
                "label-category-attributes": {
                    "Occlusion": "Partial",
                    "Type": "Sedan"
                },
                "object-name": "Car:2",
                "class-id": 0,
                "center-x": -5.188984560617168,
                "center-y": 99.7954483288783,
                "center-z": 0.2226435567445657,
                "length": 4,
                "width": 2,
                "height": 2,
                "roll": 0,
                "pitch": 0,
                "yaw": 1.6243170732068055
            }
        ]
    },
    "3d-bounding-box-metadata":
    {
        "objects": [],
        "class_map":
        {
            "0": "Car",
        },
        "type": "groundtruth/point_cloud_object_detection",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256",
        "job-name": "identify-3d-objects",
        "adjustment-status": "adjusted",
        "dataset-object-attributes": {name: value, name: value}
    }
```

```
}
```

3D 点云对象跟踪输出

以下是 3D 点云对象跟踪标记作业中的输出清单文件的示例。以下示例中的`######` 取决于标记作业规范和输出数据。省略号 (`...`) 表示该列表的延续，其中可能显示与前面对象格式相同的其他对象。

除了标准元素以外，元数据还包含一个类映射，该映射列出每个在序列中至少具有一个标签的类。在审核工作流的元数据中具有 `adjustment-status` 参数；如果修改了一个或多个长方体，该参数将设置为 `adjusted`。

```
{
   "source-ref": "s3://AWSDOC-EXAMPLE-BUCKET/myfolder/seq1.json",
    "lidar-label-attribute-ref": "s3://<CustomerOutputLocation>/<labelingJobName>/
annotations/consolidated-annotation/output/<datasetObjectId>/SeqLabel.json",
    "lidar-label-attribute-ref-metadata": {
        "objects":
        [
            {
                "frame-no": 300,
                "confidence": []
            },
            {
                "frame-no": 301,
                "confidence": []
            },
            ...
        ],
        'class-map': {'0': 'Car', '1': 'Person'},
        'type': 'groundtruth/point_cloud_object_tracking',
        'human-annotated': 'yes',
        'creation-date': '2019-11-12T01:18:14.271944',
        'job-name': 'identify-3d-objects',
        "adjustment-status": "adjusted"
    }
}
```

在上面的示例中，中的每个帧的长方体数据`seq1.json`在`SeqLabel.json`在 Amazon S3 位置，`s3://<customerOutputLocation>/<labelingJobName>/annotations/consolidated-annotation/output/<datasetObjectId>/SeqLabel.json`. 以下是此标签序列文件的示例。

对于序列中的每个帧，您会看到`frame-number`、`frame-name`，如果适用，`frame-attributes`，以及一份清单`annotations`. 此列表包含为该帧绘制的 3D 长方体。每条注释包含以下信息：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`object-name`格式`<class>:<integer>`哪里`class`识别标签类别和`integer`是整个数据集的唯一 ID。
- 在工作人员绘制长方体时，它与唯一的 `object-id` 相关联，后者与在多个帧中标识同一对象的所有长方体相关联。
- 您在输入清单中指定的每个类或标签类别与 `class-id` 关联。可以使用 `class-map` 标识与每个类 ID 关联的类。
- `center-x`、`center-y`, 和`center-z`是立方体中心的坐标，与标注作业中使用的 3D 点云输入数据位于同一坐标系中。
- `length`、`width`, 和`height`描述长方体的尺寸。
- `yaw`用于描述长方体的方向（方位），以弧度为单位。

  这些区域有：`yaw`输出数据中的测量值为 180 度，或以弧度为单位的 pi 值，减`yaw`向下看方体时，在右手的世界坐标系中。换句话说，从自上而下看一个立方体时，`yaw_in_output_data`是顺时针正数的（与右手世界坐标系相反，其中自上而下的视图与逆时针正旋转相关联）。当从长方体上看时，`yaw_in_output_data`逆时针为阳性。

转换yaw_in_output_data对于右手世界坐标系的更常见的方向，请使用以下内容（所有单位均以弧度为单位）：

```
yaw_right_handed_cartesian_system = pi - yaw_in_output_data
```

- 如果您创建了 3D 点云调整标记作业并包含在内的作业pitch和roll在输入清单文件中，相同pitch和roll测量值将显示在输出清单文件中。否则为pitch和role将始终为 0。
- 如果在输入清单文件中包含给定类的标签属性，则label-category-attributes对于工作人员选择了标签属性的所有立方体，都包括参数。


```
{
    "tracking-annotations": [
        {
            "frame-number": 0,
            "frame-name": "0.txt.pcd",
            "frame-attributes": {name: value, name: value},
            "annotations": [
                {
                    "label-category-attributes": {},
                    "object-name": "Car:4",
                    "class-id": 0,
                    "center-x": -2.2906369208300674,
                    "center-y": 103.73924823843463,
                    "center-z": 0.37634114027023313,
                    "length": 4,
                    "width": 2,
                    "height": 2,
                    "roll": 0,
                    "pitch": 0,
                    "yaw": 1.5827222214406014,
                    "object-id": "ae5dc770-a782-11ea-b57d-67c51a0561a1"
                },
                {
                    "label-category-attributes": {
                        "Occlusion": "Partial",
                        "Type": "Sedan"
                    },
                    "object-name": "Car:1",
                    "class-id": 0,
                    "center-x": -2.6451293634707413,
                    "center-y": 124.9534455706848,
                    "center-z": 0.5020834081743839,
                    "length": 4,
                    "width": 2,
                    "height": 2.080488827301309,
                    "roll": 0,
                    "pitch": 0,
                    "yaw": -1.5963335581398077,
                    "object-id": "06efb020-a782-11ea-b57d-67c51a0561a1"
                },
                {
                    "label-category-attributes": {
                        "Occlusion": "Partial",
                        "Type": "Sedan"
                    },
                    "object-name": "Car:2",
                    "class-id": 0,
                    "center-x": -5.205611313118477,
                    "center-y": 99.91731932137061,
                    "center-z": 0.22917217081212138,
                    "length": 3.8747142207671956,
```

```
                "width": 1.9999999999999918,
                "height": 2,
                "roll": 0,
                "pitch": 0,
                "yaw": 1.5672228760316775,
                "object-id": "26fad020-a782-11ea-b57d-67c51a0561a1"
            }
        ]
    },
    {
        "frame-number": 1,
        "frame-name": "1.txt.pcd",
        "frame-attributes": {},
        "annotations": [
            {
                "label-category-attributes": {},
                "object-name": "Car:4",
                "class-id": 0,
                "center-x": -2.2906369208300674,
                "center-y": 103.73924823843463,
                "center-z": 0.37634114027023313,
                "length": 4,
                "width": 2,
                "height": 2,
                "roll": 0,
                "pitch": 0,
                "yaw": 1.5827222214406014,
                "object-id": "ae5dc770-a782-11ea-b57d-67c51a0561a1"
            },
            {
                "label-category-attributes": {
                    "Occlusion": "Partial",
                    "Type": "Sedan"
                },
                "object-name": "Car:1",
                "class-id": 0,
                "center-x": -2.6451293634707413,
                "center-y": 124.9534455706848,
                "center-z": 0.5020834081743839,
                "length": 4,
                "width": 2,
                "height": 2.080488827301309,
                "roll": 0,
                "pitch": 0,
                "yaw": -1.5963335581398077,
                "object-id": "06efb020-a782-11ea-b57d-67c51a0561a1"
            },
            {
                "label-category-attributes": {
                    "Occlusion": "Partial",
                    "Type": "Sedan"
                },
                "object-name": "Car:2",
                "class-id": 0,
                "center-x": -5.221311072916759,
                "center-y": 100.4639841045424,
                "center-z": 0.22917217081212138,
                "length": 3.8747142207671956,
                "width": 1.9999999999999918,
                "height": 2,
                "roll": 0,
                "pitch": 0,
                "yaw": 1.5672228760316775,
                "object-id": "26fad020-a782-11ea-b57d-67c51a0561a1"
            }
        ]
```

```
        }
    ]
}
```

增强的数据标记

Amazon SageMaker Ground Truth 管理将数据对象发送给工作人员进行标记的过程 标记每个数据对象的过程都是一个任务。工作人员将完成每个任务，直到整个标记作业完成。Ground Truth 将任务总数划分为较小的任务批次这些都发给工作人员。前一批次完成后，向工作人员发送新批次的作业。

Ground Truth 提供两种功能来帮助提高数据标签的准确性并减少标记数据的总成本：

- 注释整合有助于提高数据对象标签的准确率。它将多个工作人员注释任务的结果合并为一个高保真标签。
- 自动数据标记 使用机器学习来自动标记数据的组成部分，而无需将其发送给工作人员。

主题

- 控制发送给工作人员的数据对象的流 (p. 436)
- 合并注释 (p. 437)
- 自动数据标记 (p. 438)
- 链接标记作业 (p. 444)

控制发送给工作人员的数据对象的流

根据您创建的标签作业的类型，Amazon SageMaker Ground Truth 会批量或以流式传输方式向工作人员发送数据对象。您可以通过以下方式控制向工作人员的数据对象流：

- 对于这两种类型的标签作业，您都可以使用`MaxConcurrentTaskCount`以控制标记作业运行的给定时间点所有工作人员可用的数据对象的总数。
- 对于流式标签作业，您可以通过监控和控制发送到与标签作业关联的 Amazon SQS 的数据对象的数量来控制数据对象流向工作人员的流。

有关这些选项的更多信息，请参阅以下部分。要了解有关流式标记作业的更多信息，请Ground Truth 直播标签职位 (p. 381).

主题

- 使用 maxconcurTaskCount 控制数据对象的流 (p. 436)
- 使用 Amazon SQS 控制数据对象流向流式标签作业的流 (p. 437)

使用 maxconcurTaskCount 控制数据对象的流

`MaxConcurrentTaskCount`定义可以同时标记的最大数据对象数。如果使用控制台，则此参数设置为 1,000。如果您使用`CreateLabelingJob`，您可以将此参数设置为介于 1 到 1,000 之间的任何整数（含）。

当您使用输入清单文件启动标记作业时，Ground Truth 将执行以下操作：

1. 对于输入清单文件中列出的每个数据对象，将创建一个或多个任务，具体取决于您为其指定的值`NumberOfHumanWorkersPerDataObject`. 例如，如果将每个数据对象的工作人员数设置为 3，则将为每个数据集对象创建 3 个任务。要标记为成功标记，至少必须有一个工作人员为对象添加标签。或者，这些任务可能会过期或被拒绝。
2. 如果您使用的是 Mechanical Turk 工作人员，Ground Truth 首先将 10 个数据集对象发送给您的员工。它使用这个小批次来设置标记作业，并确保该作业得到正确配置。

3. 接下来，Ground Truth 发送MaxConcurrentTaskCount工作人员的数据集对象的数量。例如，如果输入清单文件中有 2,000 个输入数据对象，并且已将每个数据对象的 worker 数设置为 3 并设置MaxConcurrentTaskCount至 900 个，输入清单中的前 900 个数据对象将发送给工作人员，相当于 2,700 个任务（900 x 3）。这是发送给工作人员的第一组全尺寸对象。
4. 接下来会发生的操作取决于您创建的标记作业的类型。此步骤假定输入清单文件中的一个或多个数据集对象，或者使用 Amazon SNS 输入数据源（在流式标签作业中）发送的数据集中未包含在步骤 3 中发送给工作人员的集合中。
   - 流式标记作业：只要工作人员可用的对象总数等于MaxConcurrentTaskCount，输入清单文件中以及您使用 Amazon SNS 实时发送的所有剩余数据集对象都放在 Amazon SQS 队列中。当工作人员可用的对象总数低于时MaxConcurrentTaskCount减去NumberOfHumanWorkersPerDataObject，则使用队列中的新数据对象来创建NumberOfHumanWorkersPerDataObject-任务，实时发送给工作人员。
   - 非流式标记作业：随着工作人员完成对一组对象的标签，最多MaxConcurrentTaskCount倍NumberOfHumanWorkersPerDataObject将发送给工作人员的新任务数量。重复此过程，直到输入清单文件中的所有数据对象都被标记为止。

使用 Amazon SQS 控制数据对象流向流式标签作业的流

当您创建流式贴标任务时，系统会在您的账户中自动创建 Amazon SQS 队列。仅当发送给工作人员的对象总数超过时，数据对象才会添加到 Amazon SQS 队列MaxConcurrentTaskCount. 否则，对象将直接发送给工作人员。

您可以使用此队列来管理数据对象流向标记作业的流。要了解更多信息，请参阅 使用 Amazon SQS 队列管理贴标请求 (p. 383)。

合并注释

一个注释是单个工作人员的标记作业的成果。注释整合 将两个及更多工作人员的注释整合到数据对象的单个标签中。分配给数据集中每个对象的标签是对标签的真实情况的概率估计。数据集中的每个对象通常具有多个注释，但只有一个或一组标签。

您可以决定数据集中的每个对象添加注释的数量。使用更多的工作人员可以提高标签的准确性，但也会增加标记的成本。有关 Ground Truth 定价的更多信息，请参阅Amazon SageMaker Ground Truth 定价.

如果您使用 Amazon SageMaker 控制台创建标记作业，则以下是可以对对象进行注释的工作人员数的默认值：

- 文本分类 —3 个工作人员
- 图像分类 —3 个工作人员
- 边界框 —5 个工作人员
- 语义分割 —3 个工作人员
- 指定实体识别 —3 个工作人员

当您使用 CreateLabelingJob 操作时，使用 NumberOfHumanWorkersPerDataObject 参数设置对每个数据对象进行注释的工作人员数量。您可以使用控制台或 CreateLabelingJob 操作来覆盖对数据对象进行注释的默认工作人员数量。

Ground Truth 为其每个预定义标记任务提供一个注释整合函数：边界框、图像分类、名称实体识别、语义分割和文本分类。这些函数如下所示：

- 图像和文本分类的多类别注释整合对注释使用期望最大化方法的变体。它估计每个工作人员的参数，并使用贝叶斯推断根据来自各个员工的类别注释来估计真正的类别。
- 边界框注释整合来自多个工作人员的边界框。该函数根据框的 Jaccard 指数（或交并比）查找来自不同工作人员的最相似的框，并计算它们的平均值。

- 语义分割注释整合将单个图像中的每个像素视为一个多类别分类。该函数将来自工作人员的像素注释视为“投票”，并将平滑函数应用于图像以包含来自周围像素的更多信息。
- 指定实体识别按照 Jaccard 相似度聚类文本选择，并根据模式计算选择边界，如果模式不明确，则计算中值。标签解析为聚类中具有最多分配的实体标签，通过随机选择来中断绑定。

您可以使用其他算法来合并注释。有关信息，请参阅创建您自己的注释整合函数 (p. 438)。

创建您自己的注释整合函数

您可以选择使用自己的注释合并函数来确定标记对象的最终标签。编写函数有很多可能的方法，您采用的方法取决于要合并的注释的性质。广义地说，注释函数查看来自工作人员的注释，衡量它们之间的相似性，然后使用某种形式的概率判断来确定最可能的标签应该是怎样的。

如果要使用其他算法创建注释整合函数，可以在`[project-name]`/annotations/worker-response指导作业输出的 Amazon S3 存储桶的文件夹。

评估相似性

要评估标签之间的相似性，您可以使用下列策略之一，也可以使用满足您的数据标记需要的策略：

- 对于由离散、互斥的类别（如多类别分类）组成的标签空间，评估相似性的过程可能很简单。离散标签要么匹配，要么不匹配。
- 对于没有离散值的标签空间（例如，边界框注释），请查找广泛的相似性度量。对于边界框，一个此类度量是 Jaccard 指数。此指数衡量两个框的交并比以评估它们的相似程度。例如，如果具有三个注释，则可以使用一个函数确定哪些注释表示同一对象并且应进行整合。

评估最可能的标签

考虑到前面章节中详述的策略之一，对整合的标签表示的内容做出某种概率判断。对于离散、互斥的类别，这一过程可能很简单。最常用的一种实现方法是采用注释之间的大多数投票结果。这种方法将各个注释的权重视为相同。

一些方法尝试评估不同的注释者的准确率，并按照正确性概率比例对其注释进行加权。一个此类示例是期望最大化方法，将在多类别注释的默认 Ground Truth 整合函数中使用该方法。

有关创建注释整合函数的更多信息，请参阅第 3 步：使用处理Amazon Lambda (p. 330)。

自动数据标记

如果您选择，Amazon SageMaker Ground Truth 可以使用主动学习自动标记某些内置任务类型的输入数据。主动学习 是一种机器学习技术，可识别应由您的工作人员标记的数据。在 Ground Truth 中，此功能称为自动数据标记。自动数据标记与只用人工相比，有助于减少标记数据集所需的成本和时间。当您使用自动标记时，您会产生 SageMaker 训练和推理成本。

我们推荐在大型数据集上使用自动标签，因为用于主动学习的神经网络对于每个新的数据集需要大量的数据。通常，随着您提供的数据增长，高精度预测的可能性也会增长。只有在自动标记模型中使用的神经网络能够达到可接受的高精度时，数据才会被自动标记。因此，对于较大的数据集，能自动标记数据的可能性更大，因为神经网络可以实现自动标记所需的足够高的准确性。当您拥有数千个数据对象时，自动数据标记最为合适。自动数据标记允许的最小对象数量为 1,250，但我们强烈建议至少提供 5,000 个对象。

自动数据标记仅适用于以下 Ground Truth 内置任务类型：

- 图像分类（单个标签） (p. 218)
- 图像语义分割 (p. 215)
- 对象检测 (Bounding Box (p. 211))
- 文本分类（单标签） (p. 228)

流式标记作业不支持自动数据标记。

要了解如何使用自己的模型创建自定义活跃学习工作流，请参阅。使用自己的模型设置主动学习工作流 (p. 444).

输入数据配额适用于自动数据标记作业。请参阅输入数据配额 (p. 385)有关数据集大小、输入数据大小和分辨率限制的信息。

> Note
>
> 在生产中使用自动标记模型之前，您需要对其进行优化和/或测试。您可以针对由您的标记作业生成的数据集来优化模型（或创建并优化您选择的另一个监管式模型），以优化模型的架构和超参数。如果您决定使用该模型进行推理而不进行优化，我们强烈建议确保在使用 Ground Truth 标记的数据集的代表性（例如随机选择的）子集上评估其准确率，且准确率与您的期望目标匹配。

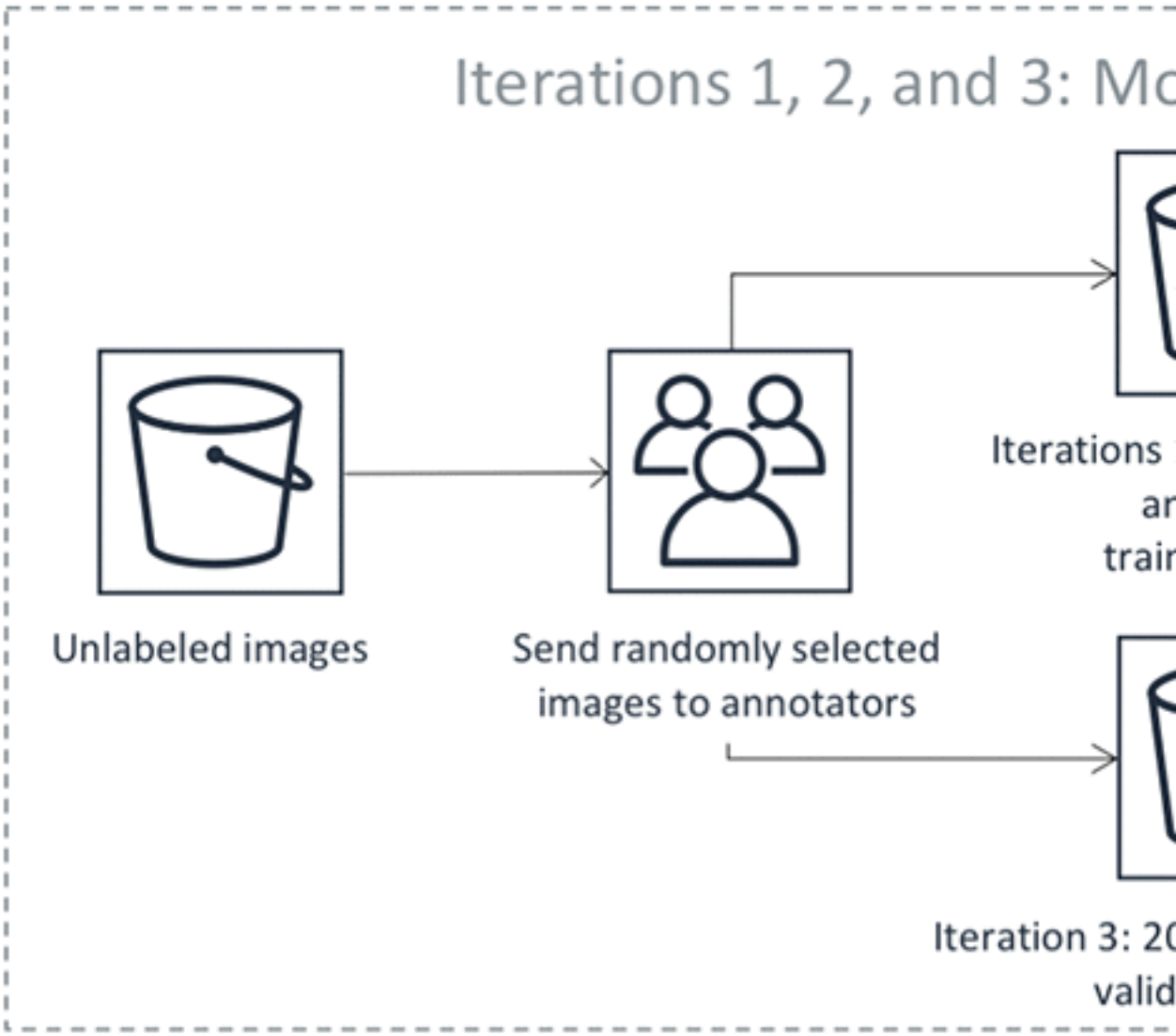
工作方式

在您创建标记作业时开启自动数据标记功能。以下是具体工作原理：

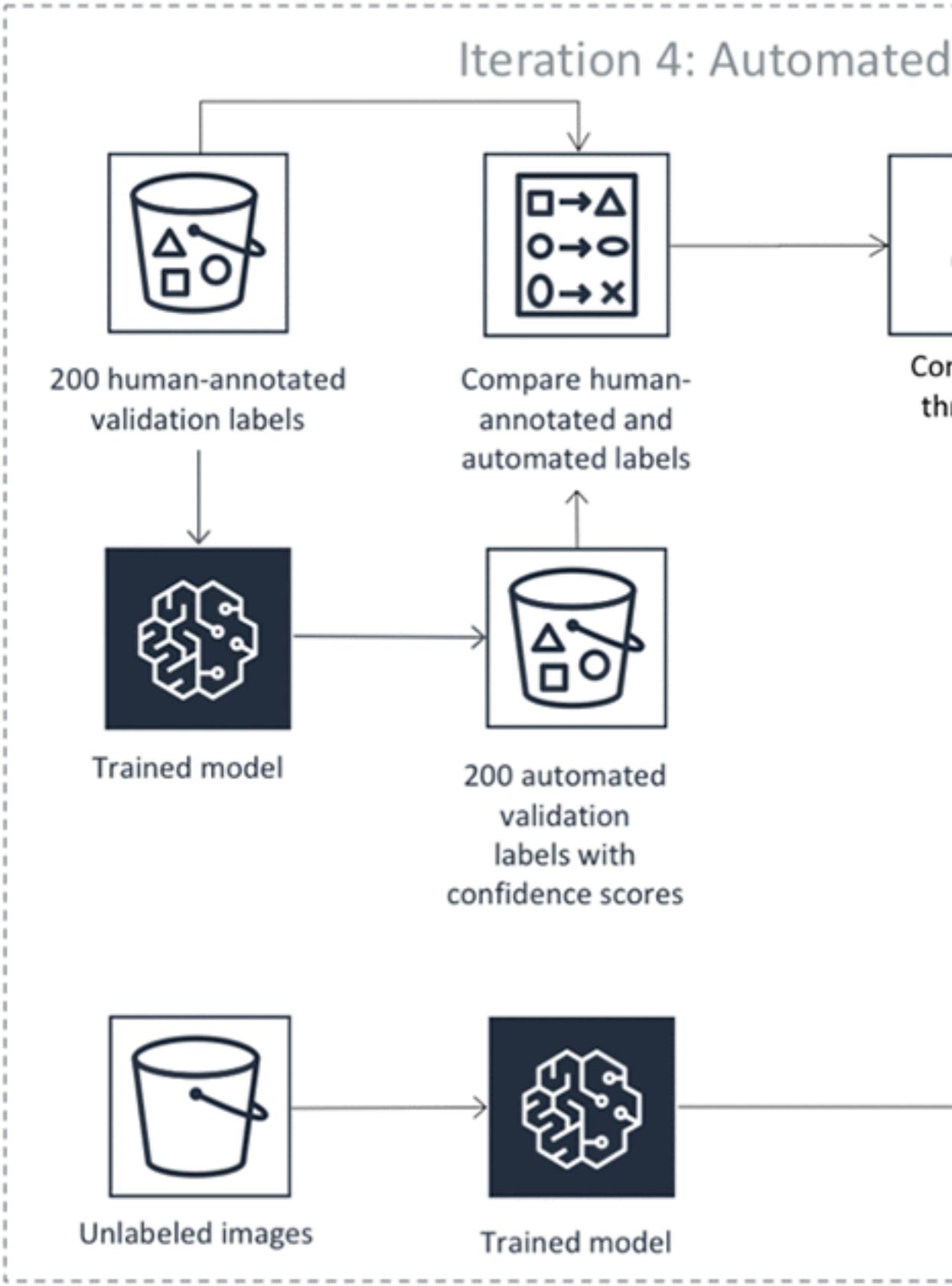
1. 当 Ground Truth 启动自动数据标记作业时，它选择输入数据对象的一个随机样本，并将其发送给工作人员。如果超过 10% 的这些数据对象失败，标注作业将失败。如果标签作业失败，除了查看 Ground Truth 返回的任何错误消息之外，还要检查您的输入数据是否在工作人员 UI 中正确显示、说明是否明确，以及您为工作人员提供了足够的时间来完成任务。
2. 返回标记的数据时，它用于创建训练集和验证集。Ground Truth 使用这些数据集来训练和验证用于自动标记的模型。
3. Ground Truth 运行批处理转换作业，同时使用验证的模型对验证数据进行推理。批量推理为验证数据中的每个对象生成置信度得分和质量指标。
4. 自动标记组件将使用这些质量指标和置信度得分创建信心分数阈值这确保了质量的标签。
5. Ground Truth 对数据集中未标记的数据运行批处理转换作业，同时使用相同的已验证模型进行推理。这会为每个对象生成置信度得分。
6. Ground Truth 自动标记组件确定在步骤 5 中为每个对象生成的置信度得分是否满足在步骤 4 中确定的所需阈值。如果置信度得分满足阈值，则自动标记的预期质量超过要求的准确度水平，并将该对象视为自动标记。
7. 步骤 6 生成具有置信度分数的未标记数据集。Ground Truth 从此数据集中选择置信度低的数据点并将其发送给工作人员。
8. Ground Truth 使用现有的人工标记数据和来自工作人员的这些额外标记数据更新模型。
9. 该过程将重复执行，直到数据集被完全标记或者直到满足其他停止条件为止。例如，如果达到人工注释预算，自动标记将停止。

上述步骤发生在迭代中。选择下表中的每个选项卡，以查看对象检测自动标记作业的每次迭代中发生的流程示例。这些图像中给定步骤中使用的数据对象的数量（例如 200）特定于此示例。如果要标记的对象少于 5,000，则验证集大小为整个数据集的 20%。如果输入数据集中有 5,000 个以上的对象，则验证集大小为整个数据集的 10%。您可以通过更改值来控制每次活动学习迭代收集的人类标签的数量`MaxConcurrentTaskCount`使用 API 操作时`CreateLabelingJob`. 当您使用控制台创建标记作业时，此值设置为 1,000。在 “主动学习流程” 下所示主动学习选项卡，此值设置为 200。

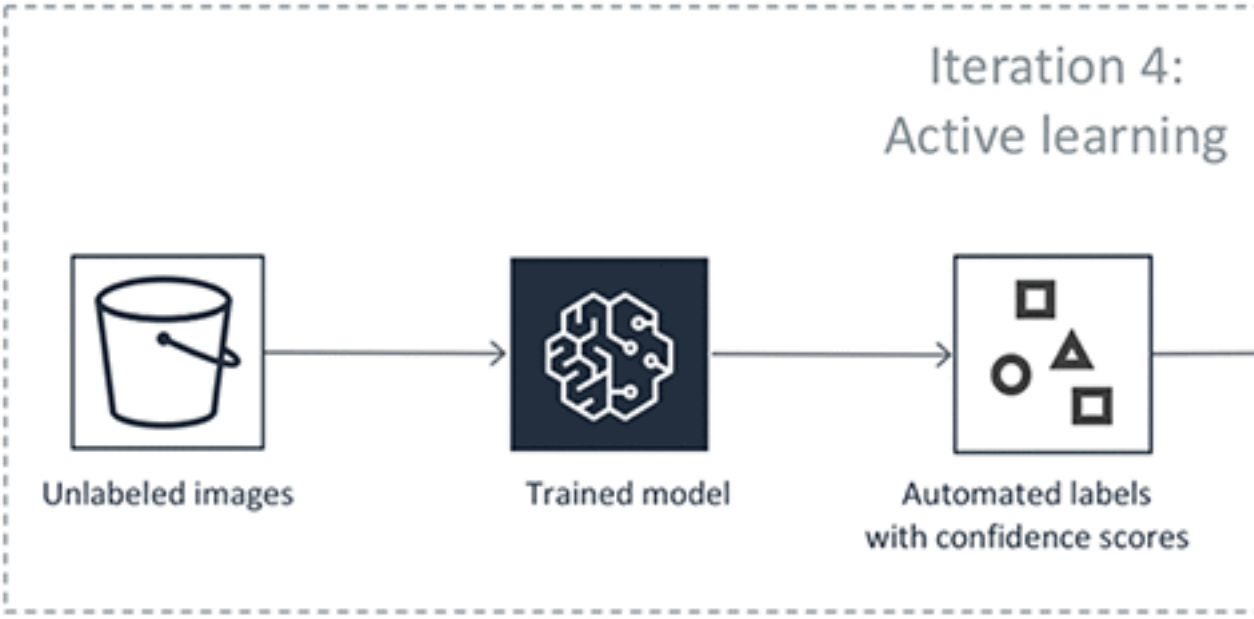
Model Training



Automated Labeling



Active Learning



自动标签的准确性

定义准确性取决于您在自动标签中使用的内置任务类型。对于所有任务类型，这些准确性要求由 Ground Truth 预先确定，不能手动配置。

- 对于图像分类和文本分类，Ground Truth 使用逻辑来查找与至少 95% 标签准确度相对应的标签预测置信度。这意味着 Ground Truth 预计，与人工贴标机为这些示例提供的标签相比，自动标签的准确性至少为 95%。
- 对于边界框，预期的平均值交叉点在联盟 (iOU)的自动标记的图像是 0.6。为了找到 IU 的平均值，Ground Truth 会计算每个班级图像上所有预测和错过的框的平均 iU，然后在各班级中对这些值进行平均值。
- 对于语义分段，自动标记图像的预期平均值为 0.7。为了找到平均值 iOU，Ground Truth 采用图像中所有类（不包括背景）的 IOU 值的平均值。

在 Active Learning 的每次迭代（上面列表中的步骤 3-6）中，可以使用人工注释的验证集找到置信阈值，以便自动标记对象的预期准确度满足某些预定义的准确性要求。

创建自动数据标记作业（控制台）

要在 SageMaker 控制台中创建使用自动标签的标签作业，请按以下步骤操作。

创建自动数据标记作业（控制台）

1. 打开地面的真相标记作业SageMaker 控制台的部分：https://console.amazonaws.cn/sagemaker/groundtruth.
2. 使用创建标记作业（控制台）(p. 353)作为指南，完成 Job overview (作业概览) 和 Task type (任务类型) 部分。请注意，自定义任务类型不支持自动标记。
3. 在 Workers (工作人员) 下，选择您的人力类型。
4. 在同一部分中，选择 Enable automated data labeling (启用自动数据标记)。
5. 使用第 4 步：配置边界框工具 (p. 209)作为指南，在 ***Task Type*** labeling tool (任务类型标记工具) 部分中创建工作人员说明。例如，如果您选择了语义分割作为标签工作类型，此部分被称为语义分割标记工具.

6. 要预览工作人员说明和控制面板，请选择 Preview (预览)。
7. 选择创建。这样，您就创建并启动标记作业和自动标记流程。

你可以看到你的贴标作业出现在标记作业SageMaker 控制台的部分。您的输出数据显示在创建标记作业时指定的 Amazon S3 存储桶中。有关标记作业输出数据的格式和文件结构的更多信息，请参阅 输出数据 (p. 414)。

创建自动数据标记作业 (API)

要使用 SageMaker API 创建自动数据标记作业，请使用`LabelingJobAlgorithmsConfig`的参数`CreateLabelingJob`operation. 要了解如何使用启动标记作业`CreateLabelingJob`操作，请参阅创建标记作业 (API) (p. 356).

在 LabelingJobAlgorithmSpecificationArn 参数中指定您用于自动数据标记的算法的 Amazon 资源名称 (ARN)。从自动标记支持的四种 Ground Truth 内置算法中选择一种：

- 图像分类（单个标签） (p. 218)
- 图像语义分割 (p. 215)
- 对象检测 (Bounding Box (p. 211))
- 文本分类（单标签） (p. 228)

当自动数据标记作业完成后，Ground Truth 返回其用于自动数据标记作业的模型的 ARN。通过在 InitialActiveLearningModelArn 参数中以字符串格式提供 ARN，将此模型用作类似的自动标记作业类型的起始模型。要检索模型的 ARN，请使用类似于以下内容的 Amazon Command Line Interface (Amazon CLI) 命令。

```
# Fetch the mARN of the model trained in the final iteration of the previous labeling
 job.Ground Truth
pretrained_model_arn = sagemaker_client.describe_labeling_job(LabelingJobName=job_name)
['LabelingJobOutput']['FinalActiveLearningModelArn']
```

要加密附加到用于自动标记的 ML 计算实例的存储卷上的数据，请在 `VolumeKmsKeyId` 参数中包括一个 Amazon Key Management Service (Amazon KMS) 密钥。有关的信息AmazonKMS 密钥，请参阅是什么 AmazonKey Management Service？中的AmazonKey Management Service 开发者指南.

对于使用`CreateLabelingJob`操作以创建自动数据标记作业，请参阅object_探测_教程示例中的SageMaker 示例、Ground Truth 标记作业SageMaker 笔记本实例的部分。要了解如何创建和打开笔记本实例，请参阅创建笔记本实例 (p. 149)。要了解如何访问 SageMaker 示例笔记本，请参阅示例笔记本 (p. 158).

自动数据标记所需的 Amazon EC2 实例

下表列出了为训练和批次推理作业运行自动数据标记所需的 Amazon Elastic Compute Cloud (Amazon EC2) 实例。

自动数据标记作业类型	训练实例类型	推理实例类型
图像分类	ml.p3.2xlarge*	ml.c5.xlarge
对象检测（边界框）	ml.p3.2xlarge*	ml.c5.4xlarge
文本分类	ml.c5.2xlarge	ml.m4.xlarge
语义分割	ml.p3.2xlarge*	ml.p3.2xlarge*

* 在亚太地区（孟买）区域 (ap-south-1) 中，改用 ml.p2.8xlarge。

Ground Truth 管理您用于自动数据标记作业的实例。它根据需要创建、配置和终止实例以执行作业。这些实例不会显示在您的 Amazon EC2 实例控制面板中。

使用自己的模型设置主动学习工作流

您可以使用自己的算法创建主动学习工作流程，以便在该工作流程中运行训练和推理以自动标记数据。笔记本电脑 bring_YOUR_own_model_for_sagemaker_labeling_workflow_WIT_active_learning.ipynb 使用 SageMaker 内置算法演示了这一点，BlazingText. 这本笔记本提供了Amazon CloudFormation你可以用来执行此工作流程的堆栈Amazon Step Functions. 您可以在此处找到笔记本和支持文件。GitHub 存储库.

您还可以在 SageMaker 示例存储库中找到此笔记本。请参阅使用示例笔记本了解如何找到 Amazon SageMaker 示例笔记本。

链接标记作业

Amazon SageMaker Ground Truth 可以通过两种方法重用以前作业中的数据集：克隆和链接。

克隆复制以前标记作业的设置，并允许您在设置以运行该作业之前进行其他更改。

链接 不仅使用以前作业的设置，而且还使用结果。这样，您就可以继续执行未完成的作业并将标签或数据对象添加到完成的作业中。链接是一种更复杂的操作。

对于数据处理：

- 克隆将以前作业的 输入 清单（带有可选修改）作为新作业的输入清单。
- 链接将以前作业的 输出 清单作为新作业的输入清单。

当您需要执行以下操作时，链接非常有用：

- 继续执行手动停止的标记作业。
- 在修复问题后继续执行中间作业失败的标记作业。
- 在作业的手动标记部分完成后切换到自动数据标记，反之亦然。
- 将更多数据对象添加到完成的作业中，并从该位置启动作业。
- 将另一个注释添加到完成的作业中。例如，您具有一个为主题标记的短语集合，并希望再次运行该集合，请按主题的隐含受众对其进行分类。

在 Amazon SageMaker Ground Truth 中，您可以通过控制台或 API 配置链接的标记作业。

关键术语：标签属性名称

标签属性名称（API 中的 `LabelAttributeName`）是一个用作键值对的键的字符串，它由工作人员为数据对象分配的标签构成。

以下规则适用于标记属性名称：

- 它不能以 `-metadata` 结束。
- `source` 和 `source-ref` 名称是保留的，不能使用这些名称。
- 对于语义分割标记作业，它必须以 `-ref` 结尾。对于所有其他标记作业，它不能 以 `-ref` 结束。如果您使用控制台创建作业，则 Amazon SageMaker Ground Truth 自动追加。`-ref`除语义分割作业以外的所有标签属性名称。
- 对于链接的标记作业，如果您使用来自原始作业的相同标签属性名称，并将链接的作业配置为使用自动标记，当它在任何时候处于自动标记模式时，Ground Truth 将使用来自原始作业的模型。

在输出清单中，标签属性名称显示为类似于以下内容。

```
"source-ref": "<S3 URI>",
"<label attribute name>": {
  "annotations": [{
    "class_id": 0,
    "width": 99,
    "top": 87,
    "height": 62,
    "left": 175
  }],
  "image_size": [{
    "width": 344,
    "depth": 3,
    "height": 234
  }]
},
"<label attribute name>-metadata": {
  "job-name": "<job name>",
  "class-map": {
    "0": "<label attribute name>"
  },
  "human-annotated": "yes",
  "objects": [{
    "confidence": 0.09
  }],
  "creation-date": "<timestamp>",
  "type": "groundtruth/object-detection"
}
```

如果您在控制台中创建一个作业，并且未明确设置标签属性名称值，Ground Truth 将作业名称作为该作业的标签属性名称。

启动链接作业（控制台）

从现有的作业列表中选择一个停止、失败或完成的标记作业。这会启用 Actions (操作) 菜单。

从 Actions (操作) 菜单中，选择 Chain (链接)。

作业概述面板

在 Job overview (作业概述) 面板中，新的 Job name (作业名称) 是根据您从中链接该作业的作业的标题设置的。您可以更改该名称。

您也可以指定与标记作业名称不同的标签属性名称。

如果您从完成的作业中进行链接，标签属性名称将使用您要配置的新作业的名称。要更改名称，请选中该复选框。

如果您从停止或失败的作业中进行链接，标签属性名称将使用您从中链接的作业的名称。可以轻松查看和编辑该值，因为选中了名称复选框。

属性标签命名注意事项

- 默认使用选择的标签属性名称。将标记没有将数据连接到该标签属性名称的所有数据对象。
- 如果 Using a label attribute name (使用标签属性名称) 在清单中不存在，将导致作业处理数据集中的所有 对象。

在这种情况下，将自动选择 input dataset location (输入数据集位置) 以作为链接的作业的输出清单。输入字段不可用，因此，您无法更改该字段。

将数据对象添加到标记作业

您无法指定备用清单文件。请手动编辑以前作业的输出清单以添加新项目，然后再启动链接的作业。Amazon S3 URI 可以帮助您查找在 Amazon S3 存储桶中存储清单的位置。从中下载清单文件，在您的计算机本地编辑该文件，然后上传新版本以替换该文件。确保在编辑期间不会引入错误。我们建议您使用 JSON linter 检查您的 JSON。很多常用的文本编辑器和 IDE 具有 linter 插件。

启动链接作业 (API)

该过程与使用 `CreateLabelingJob` 设置新的标记作业几乎相同，但有两个主要差别：

- 清单位置：而不是使用之前工作中的原始清单，而是`ManifestS3Uri`中的`DataSource`应该指向的 Amazon S3 URI输出清单来自以前的标记作业。
- 标签属性名称：设置正确的`LabelAttributeName`价值在这里很重要。这是将标记数据作为值的键值对的键部分。示例使用案例包括：
  - 将新的或更具体的标签添加到完成的作业— 设置新的标签属性名称。
  - 标记以前作业中的未标记项目— 使用以前作业中的标签属性名称。

使用部分标记的数据集

如果使用已部分标记的增强清单，您可以获得一些链接好处。请选中 Label attribute name (标签属性名称) 复选框并设置名称，以使它与清单中的名称相匹配。

如果您使用 API，则说明与启动链接的作业的说明相同。但是，请务必将清单上传到 Amazon S3 存储桶并使用清单，而不是使用以前作业中的输出清单。

这些区域有：标签属性名称清单中的值必须符合前面讨论的命名注意事项。

Ground Truth 安全性和权限

可以使用该页面上的主题以了解 Ground Truth 安全功能以及如何配置。Amazon Identity and Access Management允许 IAM 用户或角色创建标记作业的 (IAM) 权限。此外，了解如何创建执行角色. 执行角色是您在创建标记作业时指定的角色。此角色用于启动标记作业。

如果您是新用户并希望快速入门，或者如果您不需要精确的权限，请参阅使用 IAM 托管策略与 Ground Truth (p. 448).

有关 IAM 用户和角色的更多信息，请参阅身份 (用户、组和角色)(在 IAM 用户指南中)。

要了解有关将 IAM 用于 SageMaker 的更多信息，请参阅适用于 Amazon SageMaker 的 Identity and Access Management (p. 2690).

主题

- CORS 许可要求 (p. 446)
- 分配 IAM 权限以使用 Ground Truth (p. 447)
- 输出数据和存储卷加密 (p. 458)
- 人力身份验证和限制 (p. 459)

CORS 许可要求

2020 年早些时候，Chrome 和 Firefox 等广泛使用的浏览器改变了基于图像元数据旋转图像的默认行为，称为EXIF 数据. 以前，浏览器将始终按照存储在磁盘上的方式显示图像，磁盘通常是未旋转的。更改之后，图

像现在根据名为的图像元数据旋转方向值. 这对整个机器学习 (ML) 社区有重要影响。例如，如果注释图像的应用程序不考虑 EXIF 方向，它们可能会以意想不到的方向显示图像，从而导致标签不正确。

从 Chrome 89 开始，Amazon无法再自动阻止图像的旋转，因为 Web 标准组 W3C 决定控制图像轮换的能力违反了网络的同来源政策。因此，为了确保人工在提交创建标签作业的请求时以可预测的方向对输入图像进行注释，您必须向包含输入图像的 Amazon S3 存储桶添加 CORS 标头策略。

> **Important**
>
> 如果您没有向包含输入数据的 Amazon S3 存储桶添加 CORS 配置，则标记这些输入数据对象的任务将失败。

如果您通过 Ground Truth 控制台创建作业，默认情况下 CORS 处于启用状态。如果您的所有输入数据都是不您必须按照以下说明将 CORS 配置添加到包含输入数据的所有 Amazon S3 存储桶中，与输入清单文件位于同一个 Amazon S3 存储桶中。

如果您使用`CreateLabelingJob`要创建 Ground Truth 标签任务的 API，您可以将 CORS 策略添加到 S3 控制台中包含输入数据的 Amazon S3 存储桶。要在 Amazon S3 控制台中包含输入映像的 Amazon S3 存储桶上设置所需的 CORS 标头，请按照中详细的说明进行操作如何通过 CORS 添加跨域资源共享？. 对托管映像的存储桶使用以下 CORS 配置代码。如果您使用 Amazon S3 控制台将策略添加到存储桶中，则必须使用 JSON 格式。

> **Important**
>
> 如果您创建 3D 点云或视频帧标记作业，则必须为 CORS 配置添加其他规则。要了解更多信息，请参阅3D 点云标记作业权限要求 (p. 289)和视频帧 Job 权限要求 (p. 247)。

JSON

```
[{
    "AllowedHeaders": [],
    "AllowedMethods": ["GET"],
    "AllowedOrigins": ["*"],
    "ExposeHeaders": ["Access-Control-Allow-Origin"]
}]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>GET</AllowedMethod>
   <ExposeHeader>Access-Control-Allow-Origin</ExposeHeader>
 </CORSRule>
</CORSConfiguration>
```

分配 IAM 权限以使用 Ground Truth

使用本部分中的主题了解如何使用Amazon Identity and Access Management(IAM) 管理和自定义策略，用于管理对 Ground Truth 和相关资源的访问。

您可以使用本页上的章节来了解以下内容：

- 如何创建向 IAM 用户或角色授予创建标签作业的权限的 IAM 策略。管理员可以使用 IAM 策略来限制对 Amazon 的访问 SageMaker 和其他Amazon特定于 Ground Truth 的服务。
- 如何创建 SageMaker 执行角色. 执行角色是您在创建标记作业时指定的角色。该角色用于启动和管理标签作业。

以下是在该页面上找到的主题的概述：

- 如果您正在开始使用 Ground Truth，或者您的使用案例不需要精确的权限，则建议您使用中所述的 IAM 托管策略使用 IAM 托管策略与 Ground Truth (p. 448).
- 在中了解使用 Ground Truth 控制台所需的权限授予 IAM 使用权限 SageMaker Ground Truth 控制 (p. 448). 本部分包括向 IAM 实体授予创建和修改私有工作团队、订阅供应商工作团队以及创建自定义标签工作流程的权限的策略示例。
- 在创建标记作业时，您必须提供一个执行角色。请使用创建 SageMaker Ground Truth 标签 Job 的执行角色 (p. 452)以了解该角色所需的权限。

使用 IAM 托管策略与 Ground Truth

SageMaker 和 Ground Truth 提供Amazon您可以使用的托管策略来创建标记作业。如果您正在开始使用 Ground Truth 并且您的使用案例不需要精确的权限，则建议您使用以下策略：

- `AmazonSageMakerFullAccess`— 使用此策略向 IAM 用户或角色授予创建标签作业的权限。这是授予 IAM 实体使用的权限的广泛策略 SageMaker 功能以及必要的功能Amazon通过控制台和 API 提供服务。此策略授予 IAM 实体使用 Amazon Cognito 创建标签作业以及创建和管理员工的权限。要了解更多信息，请参阅AmazonSageMakerFullAccess 策略.
- `AmazonSageMakerGroundTruthExecution`— 创建执行角色，您可以附加策略`AmazonSageMakerGroundTruthExecution`转到 IAM 角色。执行角色是您在创建标签作业时指定的角色，该角色用于启动标签作业。此策略允许您创建流式和非流式标签作业，并使用任何任务类型创建标签作业。请注意此托管策略的以下限制。
  - Simple Storage Service（Amazon S3）权限：此策略授予执行角色访问 Amazon S3 存储桶的权限，其名称中包含以下字符串：`GroundTruth`、`Groundtruth`、`groundtruth`、`SageMaker`、`Sagemaker`, 和`sagemaker`或者一个带有对象标签这包括`SageMaker`名字（不区分大小写）。确保输入和输出存储桶名称包含这些字符串，或者向执行角色添加额外权限授予其访问 Amazon S3 存储桶的权限. 您必须向此角色授予权限才能在 Amazon S3 存储桶上执行以下操作：`AbortMultipartUpload`、`GetObject`, 和`PutObject`.
  - 定制工作流：在创建时自定义标记流程，此执行角色仅限于调用Amazon Lambda函数具有以下字符串之一作为函数名称的一部分的函数：`GtRecipe`、`SageMaker`、`Sagemaker`、`sagemaker`，或者`LabelingFunction`. 这适用于注释前和注释后 Lambda 函数。如果您选择使用不含这些字符串的名称，则必须显式提供。`lambda:InvokeFunction`对用于创建标签作业的执行角色的权限。

要了解如何附加Amazon针对 IAM 用户或角色（身份）的托管策略，请参阅添加和删除 IAM 身份权限(在 IAM 用户指南中)。

授予 IAM 使用权限 SageMaker Ground Truth 控制

使用 “Ground Truth” 区域 SageMaker 控制台，您需要授予 IAM 实体访问的权限 SageMaker 和其他 AmazonGround Truth 互作用的服务。访问其他所需的权限Amazon服务取决于您的使用情形：

- 所有使用案例都需要 Amazon S3 权限。这些权限必须授予对包含输入和输出数据的 Amazon S3 存储桶的访问权限。
- Amazon Web Services Marketplace使用供应商工作人员需要许可。
- 私人工作团队设置需要 Amazon Cognito 许可。
- Amazon KMS需要权限才能查看可用Amazon KMS可用于输出数据加密的密钥。
- 需要 IAM 权限才能列出预先存在的执行角色或创建新的执行角色。此外，你必须使用添加`PassRole`允许的权限 SageMaker 以使用选择的执行角色来启动标记作业。

以下部分列出了您可能希望授予 IAM 角色以使用 Ground Truth 的一个或多个功能的策略。

主题

- Ground Truth 控制台权限 (p. 449)
- 自定义标记流程权限 (p. 451)
- 私有人力权限 (p. 452)
- 供应商人力权限 (p. 452)

Ground Truth 控制台权限

向 IAM 用户或角色授 Ground Truth 使用 SageMaker 要创建标记作业，请将以下策略附加到该用户或角色。以下策略将为 IAM 角色授予权限以使用内置任务类型任务类型。如果您要创建自定义标记工作流，请在中添加策略。自定义标记流程权限 (p. 451)对于以下策略。每个Statement以下政策中包含的内容将在此代码块下面进行描述。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SageMakerApis",
            "Effect": "Allow",
            "Action": [
                "sagemaker:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "KmsKeysForCreateForms",
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:ListAliases"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AccessAwsMarketplaceSubscriptions",
            "Effect": "Allow",
            "Action": [
                "aws-marketplace:ViewSubscriptions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManager",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:CreateSecret",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ListAndCreateExecutionRoles",
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "PassRoleForExecutionRoles",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "sagemaker.amazonaws.com"
                }
            }
        },
        {
            "Sid": "GroundTruthConsole",
            "Effect": "Allow",
            "Action": [
                "groundtruthlabeling:*",
                "lambda:InvokeFunction",
                "lambda:ListFunctions",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "s3:GetBucketCors",
                "s3:PutBucketCors",
                "s3:ListAllMyBuckets",
                "cognito-idp:AdminAddUserToGroup",
                "cognito-idp:AdminCreateUser",
                "cognito-idp:AdminDeleteUser",
                "cognito-idp:AdminDisableUser",
                "cognito-idp:AdminEnableUser",
                "cognito-idp:AdminRemoveUserFromGroup",
                "cognito-idp:CreateGroup",
                "cognito-idp:CreateUserPool",
                "cognito-idp:CreateUserPoolClient",
                "cognito-idp:CreateUserPoolDomain",
                "cognito-idp:DescribeUserPool",
                "cognito-idp:DescribeUserPoolClient",
                "cognito-idp:ListGroups",
                "cognito-idp:ListIdentityProviders",
                "cognito-idp:ListUsers",
                "cognito-idp:ListUsersInGroup",
                "cognito-idp:ListUserPoolClients",
                "cognito-idp:ListUserPools",
                "cognito-idp:UpdateUserPool",
                "cognito-idp:UpdateUserPoolClient"
            ],
            "Resource": "*"
        }
    ]
}
```

此策略包括以下语句。您可以通过向`Resource`该声明的清单。

`SageMakerApis`

本声明包括`sagemaker:*`，这将允许用户执行所有操作SageMaker API 操作. 您可以通过限制用户执行不用于创建和监控标签作业的操作来缩小此策略的范围。

`KmsKeysForCreateForms`

如果要授予用户列出并选择的权限，则只需要包含此语句。Amazon KMSGround Truth 控制台中用于输出数据加密的密钥。以上策略向用户授予列出和选择账户中的任何密钥的权限。Amazon KMS. 要限制用户可以列出和选择的密钥，请在其中指定那些关键 ARN`Resource`.

`SecretsManager`

该语句为用户提供描述、列出和创建资源的权限。Amazon Secrets Manager创建标签作业所必需的。

ListAndCreateExecutionRoles

此语句授予用户列表的权限 (ListRoles) 然后创建 (CreateRole) 您的账户中的 IAM 角色。它还授予用户创建的权限 (CreatePolicy) 策略和附加 (AttachRolePolicy) 针对 IAM 实体的策略。在控制台中列出、选择并在必要时创建执行角色是必需的。

如果您已创建执行角色，并希望缩小此语句的范围，以便用户只能在控制台中选择该角色，请指定您希望用户有权在其中查看的 IAM 角色的 ARNResource然后删除操作CreateRole、CreatePolicy, 和AttachRolePolicy.

AccessAwsMarketplaceSubscriptions

需要这些权限才能查看和选择您在创建标签作业时已订阅的供应商工作团队。要授予用户以下操作的权限订阅向供应商工作团队添加声明供应商人力权限 (p. 452)到上面的政策

PassRoleForExecutionRoles

为了授予标签作业创建者预览工作人员 UI 并验证输入数据、标签和说明是否正确显示的权限，这是必需的。此声明授予 IAM 实体权限，以便将用于创建标签作业的 IAM 执行角色传递给 SageMaker 以渲染和预览工作器 UI。要缩小此策略的范围，请在下面添加用于创建标签作业的执行角色的角色 ARNResource.

GroundTruthConsole

- groundtruthlabeling— 这允许用户执行使用 Ground Truth 控制台的某些功能所需的操作。这些包括描述标签作业状态的权限 (DescribeConsoleJob) 中，列出输入清单文件中的所有数据集对象 (ListDatasetObjects) 中，如果选择了数据集采样，则过滤数据集 (RunFilterOrSampleDatasetJob) ，如果使用自动数据标签，则生成输入清单文件 (RunGenerateManifestByCrawlingJob)。这些操作仅在使用 Ground Truth 控制台时可用，不能直接使用 API 调用。
- lambda:InvokeFunction和lambda:ListFunctions— 这些操作授予用户列出和调用用于运行自定义标签工作流程的 Lambda 函数的权限。
- s3:*— 本声明中包含的所有 Amazon S3 权限都用于查看 Amazon S3 存储桶自动化数据设置(ListAllMyBuckets) 中，访问 Amazon S3 中的输入数据 (ListBucket、GetObject)，如果需要，请在 Amazon S3 中检查并创建 CORS 策略 (GetBucketCors和PutBucketCors)，然后将标签作业输出文件写入 S3 (PutObject)。
- cognito-idp— 这些权限用于创建、查看和管理使用 Amazon Cognito 的私人员队伍。要了解这些操作的更多信息，请参阅Amazon Cognito API 参考.

自定义标记流程权限

将以下语句添加到策略中的语句类似于中的语句Ground Truth 控制台权限 (p. 449)要为 IAM 用户提供选择预注释前和注释后 Lambda 函数的权限，同时创建自定义标签工作流.

```
{
    "Sid": "GroundTruthConsoleCustomWorkflow",
    "Effect": "Allow",
    "Action": [
        "lambda:InvokeFunction",
        "lambda:ListFunctions"
    ],
    "Resource": "*"
}
```

要了解如何为 IAM 实体授予权限以创建和测试注释前和注释后 Lambda 函数，请参阅使用带有 Ground Truth 的 Lambda 所需的权限.

私有人力权限

在添加到权限策略时，以下权限授予访问权限以使用 Amazon Cognito 创建和管理私有人力和工作组。使用不需要这些权限即可OIDC IdP 人力.

```
{
    "Effect": "Allow",
    "Action": [
        "cognito-idp:AdminAddUserToGroup",
        "cognito-idp:AdminCreateUser",
        "cognito-idp:AdminDeleteUser",
        "cognito-idp:AdminDisableUser",
        "cognito-idp:AdminEnableUser",
        "cognito-idp:AdminRemoveUserFromGroup",
        "cognito-idp:CreateGroup",
        "cognito-idp:CreateUserPool",
        "cognito-idp:CreateUserPoolClient",
        "cognito-idp:CreateUserPoolDomain",
        "cognito-idp:DescribeUserPool",
        "cognito-idp:DescribeUserPoolClient",
        "cognito-idp:ListGroups",
        "cognito-idp:ListIdentityProviders",
        "cognito-idp:ListUsers",
        "cognito-idp:ListUsersInGroup",
        "cognito-idp:ListUserPoolClients",
        "cognito-idp:ListUserPools",
        "cognito-idp:UpdateUserPool",
        "cognito-idp:UpdateUserPoolClient"
        ],
    "Resource": "*"
}
```

要了解有关使用 Amazon Cognito 创建私有人力的更多信息，请参阅创建和管理 Amazon Cognito 人力 (p. 467).

供应商人力权限

您可以将以下语句添加到中的策略中。授予 IAM 使用权限 SageMaker Ground Truth 控制 (p. 448)授予 IAM 实体订阅权限供应商人力.

```
{
    "Sid": "AccessAwsMarketplaceSubscriptions",
    "Effect": "Allow",
    "Action": [
        "aws-marketplace:Subscribe",
        "aws-marketplace:Unsubscribe",
        "aws-marketplace:ViewSubscriptions"
    ],
    "Resource": "*"
}
```

创建 SageMaker Ground Truth 标签 Job 的执行角色

在配置标记作业时，您需要提供执行角色，这是一个角色 SageMaker 有权假设启动和运行标签作业。

该角色必须为 Ground Truth 授予权限以访问以下内容：

- Amazon S3 可检索输入数据并将输出数据写入到 Amazon S3 存储桶中。您可以提供存储桶 ARN，以便为 IAM 角色授予权限以访问整个存储桶，也可以为该角色授予访问权限以访问存储桶中的特定资源。例如，存储桶的 ARN 可能类似于`arn:aws:s3:::awsexamplebucket1`Amazon S3 存储桶中的资源 ARN

可能类似于arn:aws:s3:::awsexamplebucket1/prefix/file-name.png. 要对 Amazon S3 存储桶中的所有资源应用操作，您可以使用通配符：*. 例如，arn:aws:s3:::awsexamplebucket1/prefix/*. 有关更多信息，请参阅 。Amazon S3 资源在《Amazon Simple Storage Service 用户指南》中。

- CloudWatch 可以记录工作人员指标并标记作业状态。
- Amazon KMS用于数据加密。(可选)
- Amazon Lambda在您创建自定义工作流时处理输入和输出数据。

此外，如果你创建直播标记作业，此角色必须具有访问的权限：

- Amazon SQS 用于创建与以前的 SQS 队列交互管理标记请求.
- Amazon SNS 用于订阅并从您的 Amazon SNS 输入主题中检索消息，并将消息发送到您的 Amazon SNS 输出主题。

所有这些权限都可以通过AmazonSageMakerGroundTruthExecution管理的策略除了：

- 对 Amazon S3 存储桶进行数据和存储卷加密。要了解如何配置这些权限，请参阅使用加密输出数据和存储卷Amazon KMS (p. 457).
- 选择和调用不包括的 Lambda 函数的权限GtRecipe、SageMaker、Sagemaker、sagemaker，或者LabelingFunction在函数名称中。
- 不包含任何一个存储桶的 Amazon S3 存储桶GroundTruth、Groundtruth、groundtruth、SageMaker、Sagemaker, 和sagemaker在前缀或存储桶名称中或对象标签这包括SageMaker名字（不区分大小写）。

如果您需要比中提供的权限更精细的权限AmazonSageMakerGroundTruthExecution，请使用以下策略示例创建适合您的特定使用案例的执行角色。

主题

- 内置任务类型（非流式处理）执行角色要求 (p. 453)
- 内置任务类型（串流）执行角色要求 (p. 454)
- 自定义任务类型的执行角色要求 (p. 456)
- 自动数据标记权限要求 (p. 456)

内置任务类型（非流式处理）执行角色要求

以下策略授予权限，以便为内置任务类型. 该执行策略不包括 Amazon KMS 数据加密或解密的权限。将每个红色的斜体 ARN 替换为您自己的 Amazon S3 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ViewBuckets",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::<input-bucket-name>",
                "arn:aws:s3:::<output-bucket-name>"
            ]
        },
```

```
        {
            "Sid": "S3GetPutObjects",
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::<input-bucket-name>/*",
                "arn:aws:s3:::<output-bucket-name>/*"
            ]
        },
        {
            "Sid": "CloudWatch",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

内置任务类型（串流）执行角色要求

如果您创建流式标记作业，则必须向用于创建标记作业的执行角色添加类似以下内容的策略。要缩小策略的范围，请将*在Resource有特定的Amazon要授予 IAM 角色访问和使用权限的资源。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::<input-bucket-name>/*",
                "arn:aws:s3:::<output-bucket-name>/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "s3:ExistingObjectTag/SageMaker": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
```

```
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<input-bucket-name>",
                "arn:aws:s3:::<output-bucket-name>"
            ]
        },
        {
            "Sid": "CloudWatch",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "StreamingQueue",
            "Effect": "Allow",
            "Action": [
                "sqs:CreateQueue",
                "sqs:DeleteMessage",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage",
                "sqs:SendMessageBatch",
                "sqs:SetQueueAttributes"
            ],
            "Resource": "arn:aws:sqs:*:*:*GroundTruth*"
        },
        {
            "Sid": "StreamingTopicSubscribe",
            "Effect": "Allow",
            "Action": "sns:Subscribe",
            "Resource": [
                "arn:aws:sns:<aws-region>:<aws-account-number>:<input-topic-name>",
                "arn:aws:sns:<aws-region>:<aws-account-number>:<output-topic-name>"
            ],
            "Condition": {
                "StringEquals": {
                    "sns:Protocol": "sqs"
                },
                "StringLike": {
                    "sns:Endpoint": "arn:aws:sns:<aws-region>:<aws-account-
number>:*GroundTruth*"
                }
            }
        },
        {
            "Sid": "StreamingTopic",
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:<aws-region>:<aws-account-number>:<input-topic-name>",
                "arn:aws:sns:<aws-region>:<aws-account-number>:<output-topic-name>"
            ]
        },
        {
            "Sid": "StreamingTopicUnsubscribe",
            "Effect": "Allow",
```

```
            "Action": [
                "sns:Unsubscribe"
            ],
            "Resource": [
                "arn:aws:sns:<aws-region>:<aws-account-number>:<input-topic-name>",
                "arn:aws:sns:<aws-region>:<aws-account-number>:<output-topic-name>"
            ]
        }
    ]
}
```

自定义任务类型的执行角色要求

如果您要创建自定义标记流程，请将以下语句添加到执行角色策略（如中的那些语句）。??? (p. 453)要么内置任务类型（串流）执行角色要求 (p. 454).

此策略授予执行角色的权限`Invoke`您的注释前和注释后 Lambda 函数。

```
{
    "Sid": "LambdaFunctions",
    "Effect": "Allow",
    "Action": [
        "lambda:InvokeFunction"
    ],
    "Resource": [
        "arn:aws:lambda:<region>:<account-id>:function:<pre-annotation-lambda-name>",
        "arn:aws:lambda:<region>:<account-id>:function:<post-annotation-lambda-name>"
    ]
}
```

自动数据标记权限要求

如果您想使用创建标记作业自动标记数据启用，您必须 1) 向附加到执行角色的 IAM 策略中添加一个策略，2) 更新执行角色的信任策略。

以下语句允许将 IAM 执行角色传递给 SageMaker，以便用于运行分别用于主动学习和自动化数据标签的训练和推理作业。将此语句添加到执行角色策略中，如中找到的那样??? (p. 453)要么内置任务类型（串流）执行角色要求 (p. 454). Replace*arn:aws:iam::<account-number>:role/<role-name>*使用执行角色 ARN。您可以在 IAM 控制台下找到您的 IAM 角色 ARN角色.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
    ],
    "Resource": "arn:aws:iam::<account-number>:role/<execution-role-name>",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": [
                "sagemaker.amazonaws.com"
            ]
        }
    }
}
```

以下语句允许 SageMaker 承担执行角色以创建和管理 SageMaker 培训和推理工作。必须将此策略添加到执行角色的信任关系中。要了解如何添加或修改 IAM 角色信任策略，请参阅修改角色(在 IAM 用户指南中)。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": {
        "Effect": "Allow",
        "Principal": {"Service": "sagemaker.amazonaws.com" },
        "Action": "sts:AssumeRole"
    }
}
```

使用加密输出数据和存储卷Amazon KMS

您可以使用Amazon Key Management Service(Amazon KMS) 通过指定一个来加密标记作业的输出数据客户托管密钥当你创建标签作业时。如果您使用 API 操作`CreateLabelingJob`要创建使用自动数据标记的标记作业，您还可以使用客户托管密钥对附加到 ML 计算实例的存储卷进行加密，以运行训练和推理作业。

本节介绍了为启用输出数据加密而必须附加到客户托管密钥的 IAM 策略，以及使用存储卷加密必须附加到客户托管密钥和执行角色的策略。要了解有关这些选项的更多信息，请参阅 输出数据和存储卷加密 (p. 458)。

使用 KMS 加密输出数据

如果你指定Amazon KMS客户托管密钥要加密输出数据，您必须向该密钥添加类似以下内容的 IAM 策略。此策略授予您用于创建标签作业的 IAM 执行角色的权限，以便使用此密钥执行中列出的所有操作`"Action"`. 要了解这些操作的更多信息，请参阅Amazon KMS许可中的Amazon Key Management Service开发人员指南的第一个版本。

要使用此策略，请在中替换 IAM 服务角色 ARN`"Principal"`使用您用于创建标记作业的执行角色的 ARN。在控制台中创建标记作业时，这是您为其指定的角色。IAM 角色在Job 概述部分。使用创建标记作业时`CreateLabelingJob`，这是你指定的 ARN`RoleArn`.

```
{
    "Sid": "AllowUseOfKmsKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/service-role/example-role"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

加密自动数据标记 ML 计算实例存储卷

如果你指定`VolumeKmsKeyId`要加密附加到机器学习计算实例的存储卷，用于自动化数据标签训练和推断，您必须执行以下操作：

- 附加权限使用 KMS 加密输出数据 (p. 457)转到客户托管密钥。
- 将类似以下内容的策略附加到您用于创建标记作业的 IAM 执行角色。这是您为其指定的 IAM 角色`RoleArn`在`CreateLabelingJob`. 要了解有关的更多信息`"kms:CreateGrant"`此策略允许的操作，请参阅`CreateGrant`中的Amazon Key Management ServiceAPI 参考。

```
{
"Version": "2012-10-17",
"Statement":
```

```
 [
   {
    "Effect": "Allow",
    "Action": [
       "kms:CreateGrant"
    ],
    "Resource": "*"
  }
]
}
```

要详细了解 Ground Truth 存储卷加密，请参阅使用 KMS 密钥加密自动数据标记存储卷（仅限 API）(p. 458).

输出数据和存储卷加密

使用 Amazon SageMaker Ground Truth，您可以标记高度敏感的数据、控制数据以及采用安全最佳实践。在标记作业运行时，Ground Truth 会对传输中的数据和静态数据进行加密。此外，你可以使用Amazon Key Management Service(Amazon KMS）使用 Ground Truth 来执行以下操作：

- 使用客户托管密钥加密输出数据。
- 使用Amazon KMS带有自动数据标记作业的客户托管密钥，用于加密附加到用于模型训练和推断的计算实例的存储卷。

可以使用该页面上的主题以了解有关 Ground Truth 安全功能的更多信息。

使用 KMS 密钥加密输出数据

您可以选择提供Amazon KMS在您创建标记作业时客户托管式密钥，Ground Truth 使用该密钥对输出数据进行加密。

如果您没有提供客户管理的密钥，亚马逊 SageMaker 使用默认设置Amazon 托管式密钥对于您的角色账户的 Amazon S3 来加密输出数据。

如果您提供了客户托管式密钥，则必须向中描述的密钥添加所需的权限。使用加密输出数据和存储卷 Amazon KMS (p. 457). 当您使用 API 操作时`CreateLabelingJob`，您可以使用参数指定您的客户托管密钥 ID。`KmsKeyId`. 要了解在使用控制台创建标记作业时如何添加客户托管密钥，请参阅以下过程。

添加Amazon KMS密钥来加密输出数据（控制台）：

1. 完成中的前 7 个步骤创建标记作业（控制台）(p. 353).
2. 在步骤 8 中，选择旁边的箭头其他配置以扩展此部分。
3. 适用于加密密钥，请选择Amazon KMS密钥，您要用于加密输出数据。
4. 完成中的剩余步骤创建标记作业（控制台）(p. 353)创建标记作业。

使用 KMS 密钥加密自动数据标记存储卷（仅限 API）

在使用自动数据标记创建标记作业时，请使用`CreateLabelingJob`API 操作，您可以选择对附加到运行训练和推理作业的 ML 计算实例的存储卷进行加密。要将加密添加到存储卷中，请使用参数`VolumeKmsKeyId`要输入Amazon KMS客户托管密钥。有关该参数的更多信息，请参阅`LabelingJobResourceConfig`。

如果您为指定密钥 ID 或 ARN`VolumeKmsKeyId`，您的 SageMaker 执行角色必须包含调用的权限`kms:CreateGrant`. 要了解如何将该权限添加到执行角色中，请参阅创建 SageMaker Ground Truth 标签 Job 的执行角色 (p. 452)。

Note

如果你指定Amazon KMS在控制台中创建标记作业时，该密钥是仅限用于加密您的输出数据。它不用于加密附加到用于自动化数据标记的 ML 计算实例的存储卷。

人力身份验证和限制

Ground Truth 使您能够使用自己的私有人力以处理标记作业。一个私有人力这是一个抽象概念，它是指为您工作的一群人。每个标记作业是使用工作组（包含人力中的工作人员）创建的。Ground Truth 支持使用 Amazon Cognito 创建私有人力。

Ground Truth 人力映射到 Amazon Cognito 用户池。Ground Truth 工作组映射到 Amazon Cognito 用户组。Amazon Cognito 管理工作人员身份验证。Amazon Cognito 支持开放 ID 连接 (OIDC)，客户可以使用自己的身份提供商 (IdP) 设置 Amazon Cognito 联合。

Ground Truth 每个账户只允许一个人力Amazon区域。每个人力具有专用的 Ground Truth 工作门户登录 URL。

您也可以将工作人员限制为无类域间路由 (CIDR) 块/IP 地址范围。这意味着，注释者必须位于特定的网络才能访问注释站点。您最多可以为一个人力添加 10 个 CIDR 块。要了解更多信息，请参阅 使用 Amazon SageMaker API 管理私人劳动力 (p. 482)。

要了解如何创建私有人力，请参阅创建私有人力（Amazon Cognito）(p. 468)。

限制对人力类型的访问

亚马逊 SageMaker Ground Truth 工作团队分为三个之一人力类型：公有（使用 Amazon Mechanical Turk）、私有和供应商。要使用这些类型之一或工作组 ARN 限制 IAM 用户对特定工作组的访问权限，请使用`sagemaker:WorkteamType`和/或`sagemaker:WorkteamArn`条件键。对于 `sagemaker:WorkteamType` 条件键，请使用字符串条件运算符。对于 `sagemaker:WorkteamArn` 条件键，请使用 Amazon 资源名称 (ARN) 条件运算符。如果用户尝试使用受限制的工作组创建标记作业，SageMaker 返回访问被拒绝错误。

以下策略演示了将 `sagemaker:WorkteamType` 和 `sagemaker:WorkteamArn` 条件键与适当的条件运算符和有效条件值结合使用的不同方法。

以下示例使用 `sagemaker:WorkteamType` 条件键和 `StringEquals` 条件运算符一起限制对公有工作组的访问。它接受以下格式的条件值：*`workforcetype`*`-crowd`，其中 *`workforcetype`* 可以等于 `public`、`private` 或 `vendor`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RestrictWorkteamType",
            "Effect": "Deny",
            "Action": "sagemaker:CreateLabelingJob",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sagemaker:WorkteamType": "public-crowd"
                }
            }
        }
    ]
}
```

以下策略演示如何使用 `sagemaker:WorkteamArn` 条件键限制对公有工作组的访问。第一个策略演示如何将该条件与工作组 ARN 的有效 IAM 正则表达式变量和`ArnLike`条件运算符。第二个策略演示如何将该条件键与 `ArnEquals` 条件运算符和工作组 ARN 一起使用。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RestrictWorkteamType",
            "Effect": "Deny",
            "Action": "sagemaker:CreateLabelingJob",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "sagemaker:WorkteamArn": "arn:aws:sagemaker:*:*:workteam/public-crowd/
*"
                }
            }
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RestrictWorkteamType",
            "Effect": "Deny",
            "Action": "sagemaker:CreateLabelingJob",
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "sagemaker:WorkteamArn": "arn:aws:sagemaker:us-
west-2:394669845002:workteam/public-crowd/default"
                }
            }
        }
    ]
}
```

监控标记作业状态

要监控标记作业的状态，您可以设置一个Amazon CloudWatch Events（CloudWatch Events）规则，当标签作业状态更改为时，Amazon SageMaker Ground Truth 面真相（Ground Truth）将事件发送到 CloudWatch Event`Completed`、`Failed`，或者`Stopped`或者当工作人员接受、拒绝、提交或返回任务时。

创建规则后，您可以添加规则目标对它来说。CloudWatch Event 使用此目标调用另一个目标Amazon处理事件的服务。例如，您可以使用 Amazon Simple Notification Service (Amazon SNS) 主题创建目标，以便在标记作业状态发生更改时向您发送电子邮件通知。

先决条件：

要创建 CloudWatch Events 规则，您将需要一个Amazon Identity and Access Management附加了 events.amazonaws.com 信任策略的 (IAM) 角色。以下是 events.amazonaws.com 信任策略的示例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "events.amazonaws.com"
        ]
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

主题

- 向 CloudWatch Events 发送事件 (p. 461)
- 设置目标以处理事件 (p. 462)
- 标记作业过期 (p. 462)
- 拒绝任务 (p. 462)

向 CloudWatch Events 发送事件

要配置 CloudWatch Events 规则以获取状态更新，请参阅事件，对于你的 Ground Truth 标签工作，请使用 Amazon Command Line Interface(Amazon CLI) `put-rule`命令。可以按照状态更改筛选发送到您的规则的事件。例如，您可以创建一个规则，以仅在标记作业状态变为 `Completed` 时向您发送通知。在使用 `put-rule` 命令时，请指定以下内容来接收标记作业状态：

- `\"source\":[\"aws.sagemaker\"]`
- `\"detail-type\":[\"SageMaker Ground Truth Labeling Job State Change\"]`

要配置 CloudWatch Events 规则以监控所有状态更改，请使用以下命令并替换占位符文本。例如，将替换为*`"GTLabelingJobStateChanges"`*使用唯一的 CloudWatch Event 规则名称以及*`"arn:aws:iam::111122223333:role/MyRoleForThisRule"`*并附加了 events.amazonaws.com 信任策略的 IAM 角色的 Amazon 资源编号 (ARN)。

```
aws events put-rule --name "GTLabelingJobStateChanges"
    --event-pattern "{\"source\":[\"aws.sagemaker\"],\"detail-type\":[\"SageMaker Ground
 Truth Labeling Job State Change\"]}"
    --role-arn "arn:aws:iam::111122223333:role/MyRoleForThisRule"
    --region "region"
```

要按作业状态进行筛选，请使用 `\"detail\":{\"LabelingJobStatus\":[\"`*`Status`*`\"]}}"` 语法。*`Status`* 的有效值为 `Completed`、`Failed` 和 `Stopped`。

以下示例创建 CloudWatch Events 规则，以便在 us-west-2（俄勒冈）区域中的标记作业的状态变为时向您发送通知。`Completed`.

```
aws events put-rule --name "LabelingJobCompleted"
    --event-pattern "{\"source\":[\"aws.sagemaker\"],\"detail-type\":[\"SageMaker Ground
 Truth Labeling Job State Change\"], \"detail\":{\"LabelingJobStatus\":[\"Completed\"]}}"
    --role-arn "arn:aws:iam::111122223333:role/MyRoleForThisRule"
    --region us-west-2
```

以下示例创建 CloudWatch Events 规则，以便在 us-east-1（弗吉尼亚）区域中的标记作业的状态变为时向您发送通知`Completed`要么`Failed`.

```
aws events put-rule --name "LabelingJobCompletedOrFailed"
    --event-pattern "{\"source\":[\"aws.sagemaker\"],\"detail-type\":[\"SageMaker Ground
 Truth Labeling Job State Change\"], \"detail\":{\"LabelingJobStatus\":[\"Completed\",
 \"Failed\"]}}"
    --role-arn "arn:aws:iam::111122223333:role/MyRoleForThisRule"
    --region us-east-1
```

要了解有关的更多信息`put-rule`请求，请参阅CloudWatch Events 中的事件模式中的Amazon CloudWatch Events 用户指南.

设置目标以处理事件

创建规则后，与以下内容类似的事件将发送到 CloudWatch Events。在此示例中，标记作业 `test-labeling-job` 的状态将变为 `Completed`。

```
{
    "version": "0",
    "id": "111e1111-11d1-111f-b111-1111b11dcb11",
    "detail-type": "SageMaker Ground Truth Labeling Job State Change",
    "source": "aws.sagemaker",
    "account": "111122223333",
    "time": "2018-10-06T12:26:13Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:sagemaker:us-east-1:111122223333:labeling-job/test-labeling-job"
    ],
    "detail": {
        "LabelingJobStatus": "Completed"
    }
}
```

要处理事件，您需要设置目标。例如，如果您希望在标记作业状态发生更改时收到电子邮件，请使用中的过程设置 Amazon SNS 通知中的Amazon CloudWatch 用户指南以设置 Amazon SNS 主题并订阅电子邮件。创建主题后，可以使用该主题创建目标。

向 CloudWatch Events 规则添加目标

1. 打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/home
2. 在导航窗格中，选择 Rules (规则)。
3. 选择要将目标添加到的规则。
4. 选择 Actions，然后选择 Edit。
5. 在 Targets (目标) 下，选择 Add Target (添加目标)，然后选择要在检测到标记作业状态更改事件时使用的 Amazon 服务。
6. 配置您的目标。有关说明，请参阅中有关配置目标的主题。Amazon该服务的文档.
7. 选择 Configure details (配置详细信息)。
8. 对于 Name (名称)，输入一个名称并（可选）在 Description (描述) 中提供有关规则用途的详细信息。
9. 请确保选中 State (状态) 旁边的复选框，以便您的规则以 Enabled (已启用) 状态列出。
10. 选择 Update rule (更新规则)。

标记作业过期

如果您的标记作业在 30 天后未完成，则会过期。如果标记作业过期，您可以链接作业以创建一个新的标记作业，该作业只会向工作人员发送未标记的数据。有关更多信息以及如何使用链接创建标记作业，请参阅 链接标记作业 (p. 444)。

拒绝任务

工作人员能够拒绝任务。

如果说明不清楚、输入数据不正确显示或遇到任务的其他问题，工作人员会拒绝任务。如果每个数据集对象的工作人员数量 (`NumberOfHumanWorkersPerDataObject`) 拒绝任务，数据对象被标记为已过期且不会发送给其他工作人员。

创建和管理人力

此功能在中国区域中不可用。

人力是您选择对数据集进行标记的一组工作人员。您可以选择 Amazon Mechanical Turk 人力、供应商管理的人力，或者也可以创建自己的私有人力来标记或查看您的数据集。无论您选择哪种人力，Amazon SageMaker 都负责向工作人员发送任务。

当你使用私人劳动力时，你还可以创建工作团队，您在您的人力中为特定人员分配的一组工作人员工作—Amazon SageMaker Ground Truth标记任务或Amazon Augmented AI人工审查任务。您可以有多个工作团队，并可以将一个或多个工作团队分配给每个作业。

您可以使用 Amazon Cognito 或自己的私人 OpenID Connect (OIDC) 身份提供商 (IdP) 来管理私人员队伍和工作团队。有关以这种方式管理您的人力所需权限的更多信息，请参阅使用亚马逊所需的权限 SageMaker TGround Truth 控制台 (p. 2699)。

主题

- 使用 Amazon Mechanical Turk 人力 (p. 463)
- 管理供应商人力 (p. 466)
- 使用私有人力 (p. 467)

使用 Amazon Mechanical Turk 人力

Amazon Mechanical Turk（Mechanical Turk）人力可为您提供最多的工作人员Amazon SageMaker Ground Truth标记作业和Amazon Augmented AI人工审查任务。Amazon Mechanical Turk 人力是世界范围内的资源。工作人员每周 7 天、每天 24 小时提供服务。当您使用 Amazon Mechanical Turk 人力时，通常会为您的人工审核任务和标记作业获得最快的周转时间。

任何 Amazon Mechanical Turk 人力账单都将作为您的 Ground Truth 或亚马逊 Augmented AI 账单的一部分处理。您不需要创建单独的 Mechanical Turk 账户即可使用 Amazon Mechanical Turk 人力。

> **Important**
>
> 您不应与此人力共享机密信息、个人信息或受保护的健康信息。当您结合使用亚马逊 A2I 时，不应使用 Amazon Mechanical Turk 员Amazon适用于包含受保护健康信息的工作负载，例如 Amazon Textract 和 Amazon Rekognition。

当您创建 Ground Truth 标签作业或 Amazon A2I 人工审核工作流程（流程定义）时，您可以选择 Mechanical Turk 作为员工。您可以使用 SageMaker 控制台和 API 创建标记作业和人工审核工作流。

当您使用 API 操作创建贴标作业或人工审核工作流程时，您将以下 ARN 用于 Amazon Mechanical Turk 员工`WorkteamArn`. Replace*`region`*使用Amazon您用来创建标签作业或人工循环的区域。例如，如果您在美国西部（俄勒冈）创建标记作业，请替换*`region`*和`us-west-2`.

- `arn:aws:sagemaker:`*`region`*`:394669845002:workteam/public-crowd/default`

Ground Truth 和 Amazon A2I要求当您使用 Mechanical Turk 时，您的输入数据没有个人身份信息 (PII)。如果您使用 Mechanical Turk 工作人员并且没有指定输入数据不含 PII，那么您的 Ground Truth 标签作业和 Augmented AI 任务将失败。当您创建 Ground Truth 标签作业以及使用内置集成或创建 Amazon A2I 人类循环时，您可以指定输入数据不含 PII`StartHumanLoop`operation.

使用以下几节可了解如何在这些服务中使用 Mechanical Turk。

主题

- 使用 Mechanical Turk 与 Ground Truth (p. 464)
- 对使用 Amazon A2I 使用 Mechanical Turk (p. 465)
- 什么时候不支持 Mechanical Turk？ (p. 466)

使用 Mechanical Turk 与 Ground Truth

当您使用控制台创建标记作业时，您可以使用 Mechanical Turk 和 GGround Truth TurkCreateLabelingJoboperation.

当您创建标记作业时，我们建议您根据作业的复杂性和您需要的质量，调整对每个数据对象进行注释的工作人员数量。Amazon SageMaker Ground Truth 使用注释整合来提高标签的质量。对于较为复杂的标记作业，工作人员数量较多会给标签质量带来帮助，但对于比较简单的作业则没多大作用。有关更多信息，请参阅 合并注释 (p. 437)。请注意，Amazon A2I 人工审核工作流程不支持注释整合。

要在您创建标记作业（控制台）时使用 Mechanical Turk：

1. 使用以下命令可以使用 SageMaker 控制台的 Ground Truth 区域创建标记作业：创建标记作业（控制台） (p. 353).
2. 当你选择工作程类型中的人力部分，选择Amazon Mechanical Turk.
3. 使用指定工作人员完成任务所需的总时间任务超时.
4. 指定任务可供工作人员使用的总时间任务过期. 这就是工作人员在任务失败之前必须拿起多长时间。
5. 选择每项任务的价格使用下拉列表。这是工作人员完成单项任务所获得的金额。
6. （可选）如果适用，请选择数据集不包含成人内容。 如果包含成人内容，则 SageMaker 可能会限制可以查看您的任务的 Mechanical Turk 工作人员。
7. 您必须阅读并确认以下声明，方法是选中复选框才能使用 Mechanical Turk 工作人员。如果您的输入数据包含机密信息、个人信息或受保护的健康信息，您必须选择其他人力。

   您了解并同意，Mechanical Turk 人力由位于世界各地的独立承包商组成，您不应与此人力共享机密信息、个人信息或受保护的健康信息。
8. （可选）选中旁边的复选框启用自动化数据标签如果您想启用自动数据标记功能。要了解有关此功能的更多信息，请参自动数据标记 (p. 438).
9. 您可以指定每个数据集对象的工作人员数下其他配置. 例如，如果在此字段中输入 3，则每个数据对象将由 3 个工作人员标记。

当你通过选择创建标签作业时Create，您的标签任务将发送给 Mechanical Turk 员。

要在您创建标记作业 (API) 时使用 Mechanical Turk：

1. 使用以下命令创建标记作业：CreateLabelingJob操作：创建标记作业 (API) (p. 356).
2. 对使用以下命令WorkteamArn. Replace*region*使用Amazon您用来创建标签作业的区域。

   ```
   arn:aws:sagemaker:region:394669845002:workteam/public-crowd/default
   ```
3. 使用TaskTimeLimitInSeconds以指定工作人员完成任务所需的总时间。
4. 使用TaskAvailabilityLifetimeInSeconds以指定任务可供工作人员使用的总时间。这就是工作人员在任务失败之前必须拿起多长时间。
5. 使用NumberOfHumanWorkersPerDataObject以指定每个数据集对象的工作人员数量。
6. 使用PublicWorkforceTaskPrice来设置每个任务的价格。这是工作人员完成单项任务所获得的金额。
7. 使用DataAttributes以指定您的输入数据不包含机密信息、个人信息或受保护的健康信息。

Ground Truth要求如果您使用 Mechanical Turk 工作人员，则输入数据不含个人身份信息 (PII)。如果您使用 Mechanical Turk 并且没有指定输入数据不含 PII，请使用`FreeOfPersonallyIdentifiableInformation`标记，你的标签作业将失败。

使用`FreeOfAdultContent`标记以声明您的输入数据不含成人内容。 如果包含成人内容，则 SageMaker 可能会限制可以查看您的任务的 Mechanical Turk 工作人员。

您可以在 GitHub 上找到的以下笔记本中看到如何使用此 API 的示例：Ground Truth Jupyter 笔记本示例. 您可以在 SageMaker 下访问这些笔记本示例笔记本 (p. 158)在笔记本实例.

对使用 Amazon A2I 使用 Mechanical Turk

在创建人工审核工作流程时，您可以指定要将 Mechanical Turk 与 Amazon A2I 结合使用，也称为流定义、在控制台中或使用`CreateFlowDefinition`API 操作。当您使用此人工审核工作流程配置人工循环时，必须指定输入数据不含 PII。

要在您创建人工审核工作流程（控制台）时使用 Mechanical Turk：

1. 使用以下命令在 SageMaker 控制台的 Augmented AI 部分创建人工审核工作流程：创建人工审核工作流程（控制台）(p. 2623).
2. 当你选择工作程类型中的人力部分，选择Amazon Mechanical Turk.
3. 选择每项任务的价格使用下拉列表。这是工作人员完成单项任务所获得的金额。
4. （可选）您可以指定每个数据集对象的工作人员数下其他配置. 例如，如果在此字段中输入 3，则每个数据对象将由 3 个工作人员标记。
5. （可选）指定工作人员完成任务所需的总时间任务超时.
6. （可选）指定任务可供工作人员使用的总时间任务过期. 这就是工作人员在任务失败之前必须拿起多长时间。
7. 创建人工审核工作流程后，您可以通过在参数中提供人工循环来配置人工循环，方法是在参数中提供其亚马逊资源名称 (ARN)`FlowDefinitionArn`. 您可以使用内置任务类型的 API 操作之一或 Amazon A2I 运行时 API 操作来配置人工循环，`StartHumanLoop`. 要了解更多信息，请参阅 创建和启动人工循环 (p. 2640)。

   配置人类循环时，必须使用`FreeOfPersonallyIdentifiableInformation`中的内容分类器`DataAttributes`. 如果您使用 Mechanical Turk 并且没有指定输入数据不含 PII，那么您的人工审核任务将失败。

   使用`FreeOfAdultContent`标记以声明您的输入数据不含成人内容。 如果包含成人内容，则 SageMaker 可能会限制可以查看您的任务的 Mechanical Turk 工作人员。

要在您创建人工审核工作流程 (API) 时使用 Mechanical Turk：

1. 使用以下命令创建人工审核工作流程`CreateFlowDefinition`操作：创建人工审核工作流程 (API) (p. 2624).
2. 对使用以下命令`WorkteamArn`. Replace*region*使用Amazon您用来创建标签作业的区域。

   ```
   arn:aws:sagemaker:region:394669845002:workteam/public-crowd/default
   ```
3. 使用`TaskTimeLimitInSeconds`以指定工作人员完成任务所需的总时间。
4. 使用`TaskAvailabilityLifetimeInSeconds`以指定任务可供工作人员使用的总时间。这就是工作人员在任务失败之前必须拿起多长时间。
5. 使用`TaskCount`以指定每个数据集对象的工作人员数量。例如，如果为此参数指定 3，则每个数据对象将由 3 个工作器标记。
6. 使用`PublicWorkforceTaskPrice`来设置每个任务的价格。这是工作人员完成单项任务所获得的金额。

7. 创建人工审核工作流程后，您可以通过在参数中提供人工循环来配置人工循环，方法是在参数中提供其亚马逊资源名称 (ARN)`FlowDefinitionArn`. 您可以使用内置任务类型的 API 操作之一或 Amazon A2I 运行时 API 操作来配置人工循环，`StartHumanLoop`. 要了解更多信息，请参阅 创建和启动人工循环 (p. 2640)。

   配置人类循环时，必须使用`FreeOfPersonallyIdentifiableInformation`中的内容分类器`DataAttributes`. 如果您使用 Mechanical Turk 并且没有指定输入数据不含 PII，那么您的人工审核任务将失败。

   使用`FreeOfAdultContent`标记以声明您的输入数据不含成人内容。 如果包含成人内容，则 SageMaker 可能会限制可以查看您的任务的 Mechanical Turk 工作人员。

您可以在 GitHub 上找到的以下笔记本中看到如何使用此 API 的示例：Amazon A2I Jupyter 笔记本电脑示例.

什么时候不支持 Mechanical Turk？

在以下情况下，该员工不受支持。在每种情况下，都必须使用私人的要么供应商人力。

- Ground Truth 视频帧标记作业和 3D 点云标记作业不支持此人力。
- 如果您的输入数据包含个人身份信息 (PII)，则无法使用此人力。
- Mechanical Turk 在某些Amazon特殊区域。如果适用，请参阅您的特殊区域的文档以了解更多信息。

管理供应商人力

您可以使用供应商管理的人力，以使用 Amazon SageMaker Ground Truth（地面真相）和 Amazon Augmented AI 能（Amazon A2I）标记您的数据。供应商在提供数据标记服务以进行机器学习方面具有丰富的经验。必须通过 Amazon SageMaker 控制台单独创建和管理这两种服务的供应商人力。

供应商通过AmazonMarketplace。您可以在供应商服务的详细信息页面上找到服务的详细信息，如工作人员数和他们工作的小时数。您可以使用这些详细信息估计标记作业所需的成本和您预期作业可能需要的时间。一旦选择供应商，即表示您可以使用AmazonMarketplace。

订阅是您与供应商之间的协议。协议中对价格、时间表或退款政策等协议项目作出详细的规定。如果您对标记作业有任何问题，请直接与供应商联系。

您可以订阅任意数量的供应商，以满足您的数据注释需求。当您创建标记作业或人工审核工作流程时，您可以指定将作业发送到特定供应商。

> Important
>
> 在您将敏感数据发送给供应商之前，请检查供应商详细信息页面上的安全和合规举措，并查看作为订阅协议一部分的最终用户许可协议 (EULA)。您有责任确保供应商符合您对个人或机密信息的合规性要求。请勿与此人力共享受保护的健康信息。

您必须使用控制台来订阅供应商人力。订阅后，您便可以使用 `ListSubscribedWorkteams` 操作列出您订阅的供应商。

订阅供应商人力

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 从选择 SageMaker 控制台中的适当页面。
   - 对于 Ground Truth 标签工作，请选择对人力进行标记，选择供应商，然后选择查找数据标记服务.
   - 对于 Amazon A2I 人工审核工作流程，请选择人工审查员工队伍，选择供应商，然后选择查找人工审核服务.
3. 控制台将打开 Amazon Web Services Marketplace，并且：

- 对于，选择数据标记服务类别
- 对于，选择亚马逊 A2I 人工审核服务类别

此处显示了可用于此服务的供应商服务的列表。

4. 选择供应商。Amazon Web Services Marketplace 显示有关数据标记服务或人工审核服务的详细信息。使用此信息可以确定供应商是否满足您的任务要求。
5. 如果供应商满足您的要求，请选择 Continue to subscribe (继续订阅)。
6. 查看订阅的详细信息。如果您同意接受条款，选择订阅以完成服务的订阅。

使用私有人力

私有人力是您 选择的一组工作人员。他们可能是贵公司的员工，也可能是您所在行业的一组主题专家。例如，如果任务是标记医疗图像，您可以创建一个由对相关图像具有相关知识的人员组成的私有人力。

每个Amazon每个账户均可访问每个区域中的一个私有人力，并且拥有者能够创建多个私人的 工作团队在那个人力中。可以使用单个私有工作团队完成一个标记作业/人员审核任务或一个作业。您可以将每个工作团队分配给一个单独的作业，也可以使用单个团队完成多个作业。一个工作人员可以参加多个工作团队。

您的私人劳动力可以使用Amazon Cognito或您自己的私有 OpenID Connect (OIDC) 身份提供商 (IdP)。

如果您是的新用户Amazon SageMaker Ground Truth要么Amazon Augmented AI并且不要求使用自己的 IdP 管理员工，建议您使用 Amazon Cognito 来创建和管理私人员工队伍。

创建人力后，除创建和管理工作团队之外，您还可以执行以下操作：

- 跟踪工作人员性能
- 创建和管理 Amazon SNS 主题在标签任务可用时通知工作人员
- 使用 IP 地址管理私有人力对任务的访问

> **Note**
>
> 您的私有人力在 Ground Truth 和 Amazon A2I 之间共享。要创建和管理 Augmented AI 使用的私有工作团队，请使用 SageMaker 控制台的 Ground Truth 部分。

主题

- 创建和管理 Amazon Cognito 人力 (p. 467)
- 创建和管理 OIDC IdP 人力 (p. 474)
- 使用 Amazon SageMaker API 管理私人劳动力 (p. 482)
- 跟踪工作人员性能 (p. 483)
- 为工作团队创建和管理 Amazon SNS 主题 (p. 484)

创建和管理 Amazon Cognito 人力

如果您想使用 Amazon SageMaker 控制台创建员工，或者不想承担管理员工凭证和身份验证的开销，请使用 Amazon Cognito 创建和管理私人员工队伍。当您使用 Amazon Cognito 创建私有人力时，它会为您的私有人力提供身份验证、授权和用户管理。

主题

- 创建私有人力（Amazon Cognito）(p. 468)
- 管理私有人力（Amazon Cognito）(p. 470)

创建私有人力（Amazon Cognito）

当您使用 Amazon Cognito 时，您可以通过以下方式之一创建私有人力：

- 当您创建标记作业时，创建新的人力。要了解如何操作，请参阅在创建标记 Job 业时创建 Amazon Cognito 人力 (p. 468)。
- 在创建标签工作之前，创建新的员工队伍。要了解如何操作，请参阅使用 “标记人力” 页面创建 Amazon Cognito 人力 (p. 469)。
- 在 Amazon Cognito 控制台中创建用户池后，导入现有员工队伍。要了解如何操作，请参阅创建私有人力（Amazon Cognito 控制台）(p. 469)。

创建私有人力后，该人力及其关联的所有工作团队和工作人员都可用于所有标记 Ground Truth 业任务和 Amazon Augmented AI 人工审核工作流程任务。

如果您是 Amazon SageMaker 的新手并且想要测试 Ground Truth 或 Amazon A2I，我们建议您使用控制台创建一个由组织人员组成的私人工作团队。在创建标签或人工审核工作流程（流程定义）时，请使用此工作团队来测试工作人员 UI 和作业工作流程。

主题

- 创建私有人力（Amazon SageMaker 控制台）(p. 468)
- 创建私有人力（Amazon Cognito 控制台）(p. 469)

创建私有人力（Amazon SageMaker 控制台）

您可以通过以下两种方式之一在 Amazon SageMaker 控制台中创建私有人力：

- 在中创建标记作业时标记作业Amazon SageMaker Ground Truth 部分的页面。
- 使用标记人力Amazon SageMaker Ground Truth 部分的页面。如果您在为 Amazon A2I 人工审核工作流创建私有人力，请使用此方法。

这两种方法还会创建一个包含人力所有成员的默认工作团队。 此私有人力可用于 Ground Truth 和 Amazon 增强人工智能作业。

当您使用控制台创建私人劳动力时，SageMaker 会使用 Amazon Cognito 作为员工的身份提供商。如果您想使用自己的 OpenID Connect (OIDC) 身份提供商 (IdP) 来创建和管理私人劳动力，则必须使用 SageMaker API 操作创建员工`CreateWorkforce`. 要了解更多信息，请参阅 创建私有人力（OIDC IdP）(p. 474)。

在创建标记 Job 业时创建 Amazon Cognito 人力

如果您在创建标记作业时尚未创建私有人力，而您选择使用私有人力，系统会提示您创建一个工作团队。这将使用 Amazon Cognito 创建私有人力。

在创建标记作业时创建人力（控制台）

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中，选择 Labeling jobs (标记作业) 并填写所有必填字段。有关如何启动标记作业的说明，请参阅开始使用 (p. 206)。选择 Next（下一步）。
3. 为人力类型选择 Private (私有)。
4. 在人力部分中，输入：
   a. Team name (团队名称)。
   b. 最多 100 名人力成员的电子邮件地址。电子邮件地址区分大小写。您的工作人员必须使用与最初输入地址时使用的相同大小写进行登录。可以在创建作业后添加其他人力成员。
   c. 组织的名称。SageMaker 使用此选项来自定义发送给工作人员的电子邮件。

d. 工作人员报告与任务相关的问题时所使用的联系电子邮件地址。

当您创建标记作业时，将向每个工作人员发送一封电子邮件以邀请他们加入人力。创建人力后，您可以使用 SageMaker 控制台或 Amazon Cognito 控制台添加、删除和禁用工作人员。

使用 “标记人力” 页面创建 Amazon Cognito 人力

要使用 Amazon Cognito 创建和管理私有人力，您可以使用标记人力页. 在按照以下说明操作时，您可以选择通过输入从 Amazon Cognito 用户池导入预先存在的人力来创建私有人力。要导入人力，请参阅创建私有人力（Amazon Cognito 控制台）(p. 469)。

使用工作人员电子邮件创建私有人力

1. 打开位于 https://console.aws.amazon.com/sagemaker/ 的 Amazon SageMaker 控制台。
2. 在导航窗格中，选择 Labeling workforces (标记人力)。
3. 选择 Private (私有)，然后选择 Create private team (创建私有团队)。
4. 选择 Invite new workers by email (通过电子邮件邀请新工作人员)。
5. 在电子邮件地址框中粘贴或键入包含最多 50 个电子邮件的列表，以逗号分隔。
6. 输入组织名称和联系人电子邮件。
7. （可选）选择团队订阅到的 SNS 主题，这样在有新的 Ground Truth 标记作业可用时，其工作人员将收到通知。Amazon SNS 通知受 Ground Truth 支持，而且不受 Augmented AI 的支持。如果您将工作人员订阅为接收 SNS 通知，他们仅接收有关 Ground Truth 标记作业的通知。而不会收到有关 Augmented AI 任务的通知。
8. 单击 Create private team (创建私有团队) 按钮。

在导入私有人力后，刷新页面。在存储库的私人劳动力总结页面上，您可以看到有关人力的 Amazon Cognito 用户池的信息、人力的工作团队的列表，以及您的私有人力的所有成员的列表。

Note

如果您删除所有私有工作团队，则必须重复此过程才能在该区域内使用是有人力。

创建私有人力（Amazon Cognito 控制台）

Amazon Cognito 用于定义和管理您的私有人力和工作团队。它是一项服务，可用于为工作人员创建身份并通过身份提供商对这些身份进行验证。 私有人力对应于单个人力Amazon Cognito 用户池. 私人工作团队对应Amazon Cognito 用户组在该用户池内。

Amazon Cognito 支持的示例身份提供商：

- 社交登录提供商，例如 Facebook 和 Google
- OpenID Connect (OIDC) 提供商
- 安全断言标记语言 (SAML) 提供商，如 Active Directory
- Amazon Cognito 内置身份提供商

有关更多信息，请参阅 。什么是 Amazon Cognito？.

要使用 Amazon Cognito 创建私有人力，您现在必须拥有一个包含至少一个用户组的 Amazon Cognito 用户池。请参阅教程：创建用户池以了解如何创建用户池。请参阅将组添加到用户池以了解如何将用户组添加到池。

创建用户池后，请按照以下步骤操作，通过将该用户池导入 Amazon SageMaker 来创建私有人力。

通过导入 Amazon Cognito 用户池来创建私有人力

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.

2. 在导航窗格中，选择 Labeling workforces (标记人力)。
3. 选择 Private (私有)。
4. 选择 Create private team (创建私有团队)。这将创建一个私有人力和一个工作团队。
5. 选择从现有 Amazon Cognito 用户组导入工作人员.
6. 选择已创建的用户池。用户池需要域和现有用户组。如果您收到缺少域名的错误，请在域名上的选项应用集成您的组的 Amazon Cognito 控制台的页面。
7. 选择一个应用程序客户端。我们建议使用 SageMaker 生成的客户端。
8. 从池中选择一个用户组以导入其成员。
9. （可选）选择团队订阅到的 Amazon Simple Notification Service (Amazon SNS) 主题，这样在有新的标记作业可用时，其工作人员将收到通知。Amazon SNS 通 Ground Truth 支持，但不受支持。如果您将工作人员订阅为接收 SNS 通知，他们仅接收有关 Ground Truth 标记作业的通知。他们不会收到有关 Augmented AI 任务的通知。
10. 选择 Create private team (创建私有团队)。

    Important

    在使用 Amazon Cognito 用户池创建人力后，如果未先从 SageMaker 控制台中删除与该池关联的所有工作团队，则不应删除此人力。

在导入私有人力后，请刷新页面以查看 Private workforce summary (私有人力摘要) 页面。在此页面上，您可以看到有关人力的 Amazon Cognito 用户池的信息、人力的工作团队的列表，以及您的私有人力的所有成员的列表。此人力现在可分别在 Amazon SageMaker Ground Truth 中用于人工审核任务和数据标记作业。

管理私有人力（Amazon Cognito）

使用 Amazon Cognito 创建私有员工后，您可以使用 Amazon SageMaker 控制台和 API 操作创建和管理工作团队。

您可以使用SageMaker 控制台要么Amazon Cognito 控制台.

- 添加和删除工作团队。
- 将工作人员添加到您的人力和一个或多个工作团队。
- 从您的人力和一个或多个工作团队中禁用或删除工作人员。如果您使用 Amazon Cognito 控制台将工作人员添加到人力，则必须使用同一控制台从人力中删除工作人员。

您可以使用 SageMaker API 限制特定 IP 地址的工作人员访问任务。有关更多信息，请参阅使用 Amazon SageMaker API 管理私人劳动力 (p. 482)。

主题

- 管理员工队伍（Amazon SageMaker 控制台）(p. 470)
- 管理私有人力（Amazon Cognito 控制台）(p. 472)

管理员工队伍（Amazon SageMaker 控制台）

您可以使用 Amazon SageMaker 控制台创建和管理构成私有人力的工作团队和个人工。

使用工作团队将私有人力的成员分配给标记或人工审核工作. 当您使用 SageMaker 控制台创建员工时，会有一个名为每个人在私人劳动力这使您能够将整个员工分配给工作。由于导入 Amazon Cognito 用户池可能包含您不想包含在工作团队中的成员，因此不会为 Amazon Cognito 用户池创建类似的工作团队。

您可选择两种方法来创建新的工作团队：

- 您可以在 SageMaker 控制台中创建工作团队，并将人力中的成员添加到团队。

- 您可以使用 Amazon Cognito 控制台创建用户组，然后通过导入用户组创建工作团队。您可以将多个用户组导入每个工作团队。您可以在 Amazon Cognito 控制台中更新用户组来管理工作团队的成员。参阅 管理私有人力（Amazon Cognito 控制台）(p. 472) 了解更多信息。

使用 SageMaker 控制台创建工作团队

您可以使用 SageMaker 控制台创建新的 Amazon Cognito 用户组或导入现有用户组，在标记人力页. 有关在 Amazon Cognito 控制台中创建用户组的更多信息，请参阅管理私有人力（Amazon Cognito 控制台）(p. 472).

使用 SageMaker 控制台创建工作团队

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 从左侧菜单中，选择 Labeling workforces (标记人力)。
3. 在 Private (私有) 下，选择 Create private team (创建私有团队)。
4. 在 Team details (团队详细信息) 下，输入 Team name (团队名称)。该名称在您的账户和 Amazon 区域中都必须是唯一的。
5. 在 Add workers (添加工作人员) 下，选择使用用户组向团队添加工作人员的方法。
   - 如果你选择了通过将工作人员添加到新的 Amazon Cognito 用户组创建团队，选择要添加到团队的工作人员。
   - 如果你选择了通过导入现有 Amazon Cognito 用户组创建团队，选择作为新团队的一部分的用户组。
6. 如果您选择一个 SNS topic (SNS 主题)，则添加到团队的所有工作人员将订阅该 Amazon SNS 主题，并在团队具有新的工作项时收到通知。从现有的与 Ground Truth 关的 Amazon SNS 主题列表中进行选择，或选择创建新主题以打开主题创建对话框。

   Add Augmented AI 不支持 Amazon SNS 通知。如果您将工作人员订阅为接收 SNS 通知，他们将仅接收有关 GGround Truth 标记作业的通知。而不会收到有关 Augmented AI 任务的通知。

如果工作团队具有新的 Ground Truth 标记作业以及作业即将到期，该团队中订阅某个主题的工作人员将会收到通知。

有关使用 Amazon SNS 主题的更多信息，请阅读为工作团队创建和管理 Amazon SNS 主题 (p. 484)。

订阅

在创建工作团队后，您可以访问 Amazon Cognito 控制台以查看有关该团队的更多信息，并更改或设置其成员订阅的 Amazon SNS 主题。 如果在为团队订阅主题之前添加了任何团队成员，则需要手动为这些成员订阅该主题。有关创建和管理 Amazon SNS 主题的更多信息，请参阅为工作团队创建和管理 Amazon SNS 主题。

添加或删除工作人员

一个工作团队是人力中您可以向其分配作业的一组工作人员。您可以将一个工作人员添加到多个工作团队。在将一个工作人员添加到工作团队后，可以禁用或删除该工作人员。

将工作人员添加到人力

通过将工作人员添加到人力，您可以将该工作人员添加到人力中的任何工作团队。

使用私有人力摘要页面添加工作人员

1. 打开位于 https://console.aws.amazon.com/sagemaker/ 的 Amazon SageMaker 控制台。
2. 选择标记人力以导航到私有人力摘要页面。
3. 选择 Private (私有)。
4. 选择邀请新工作人员.

5. 在电子邮件地址框中粘贴或键入电子邮件列表，以逗号分隔。此列表中最多可包含 50 个电子邮件地址。

将工作人员添加到工作团队

必须先将工作人员添加到人力，然后才能将其添加到工作团队。要将工作人员添加到工作团队，请首先使用上述步骤导航到 Private workforce summary (私有人力摘要) 页面。

从私有人力摘要页面将工作人员添加到工作团队

1. 在私有团队部分中，选择要向其中添加工作人员的团队。
2. 选择 Workers (工作人员) 选项卡。
3. 选择向团队添加工作人员然后选择要添加的工作人员旁边的框。
4. 单击向团队添加工作人员.

禁用工作人员并将其从人力中删除

禁用工作人员将使其不再接收作业。此操作不会从人力中删除工作人员，也不会从与工作人员关联的任何工作团队中删除工作人员。要禁用工作人员或从工作团队中删除工作人员，请首先使用上述步骤导航到 “私有人力摘要” 页面。

使用 “私有人力摘要” 页面停用工作人员

1. 在 Workers (工作人员) 部分中，选择要禁用的工作人员。
2. 选择 Disable（禁用 Amazon Macie)。

如果需要，您可以在禁用工作人员后再将其 Enable (启用)。

如果已将工作人员添加到控制台中，您可以在 SageMaker 控制台中直接从私有人力中删除工作人员。如果您在 Amazon Cognito 控制台中添加工作人员（用户），请参阅管理私有人力（Amazon Cognito 控制台）(p. 472)以了解如何在 Amazon Cognito 控制台中删除工作人员。

使用 “私有人力摘要” 页面删除工作人员

1. 在 Workers (工作人员) 部分中，选择要删除的工作人员。
2. 如果尚未禁用工作人员，请选择 Disable (禁用)。
3. 选择工作人员，然后选择 Delete (删除)。

管理私有人力（Amazon Cognito 控制台）

私有人力对应于单个人力Amazon Cognito 用户池. 私人工作团队对应Amazon Cognito 用户组在该用户池中。对应于工作人员Amazon Cognito 用户在这些团体中。

在创建人力后，您可以通过 Amazon Cognito 控制台添加工作团队和各个工作人员。在 Amazon Cognito 控制台中，您可以从私有人力中删除工作人员，或从各个团队中删除工作人员。

> **Important**
>
> 无法从 Amazon Cognito 控制台中删除工作团队。删除与 Amazon SageMaker 工作团队关联的 Amazon Cognito 用户组将导致出现错误。要删除工作团队，请使用 SageMaker 控制台。

创建工作团队（Amazon Cognito 控制台）

您可以通过将 Amazon Cognito 用户组添加到与私有人力关联的用户池来创建新的工作团队以完成作业。要将 Amazon Cognito 用户组添加到现有工作人员池，请参阅向用户池添加组.

使用现有 Amazon Cognito 用户组创工作团队

1. 在打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中，选择 Workforces (人力)。
3. 对于 Private teams (私有团队)，选择 Create private team (创建私有团队)。
4. 在 Team details (团队详细信息) 下，给该团队命名。该名称在您的账户和 Amazon 区域中都必须是唯一的。
5. 适用于添加工作人员，选择导入现有 Amazon Cognito 用户组，然后选择作为新团队的一部分的一个或多个用户组。
6. 如果您选择SNS 主题，添加到团队的所有工作人员将订阅 Amazon Simple Notification Service (Amazon SNS) 主题，并在团队具有新的工作项时收到通知。从与 SageMaker Ground Truth 或 Amazon Augmented AI 有关的现有 SNS 主题列表中进行选择，或选择创建新主题创建一个。

   Note

   Amazon SNS 通知受 Ground Truth 的支持，但不受 Augmented AI 的支持。如果您将工作人员订阅为接收 SNS 通知，他们仅接收有关 Ground Truth 标记作业的通知。他们不会收到有关 Augmented AI 任务的通知。

订阅

在创建工作团队后，您可以使用 Amazon Cognito 控制台查看有关该团队的更多信息，并更改或设置其成员订阅的 SNS 主题。 如果在为团队订阅主题之前添加了任何团队成员，则需要手动为这些成员订阅该主题。有关更多信息，请参阅 为工作团队创建和管理 Amazon SNS 主题 (p. 484)。

添加和删除工作人员（Amazon Cognito 控制台）

在使用 Amazon Cognito 控制台将工作人员添加到工作团队时，必须先将用户添加到与人力关联的用户池，然后再将该用户添加到用户组。可通过多种方式将用户添加到用户池。有关更多信息，请参阅注册并确认用户账户。

将工作人员添加到工作团队

在将一个用户添加到一个池中后，此用户可与该池内的用户组关联。在将一个用户添加到一个用户组后，此用户将成为使用该用户组创建的任何工作团队的工作人员。

将用户添加到用户组

1. 打开 Amazon Cognito 控制台：https://console.aws.amazon.com/cognito/.
2. 选择 Manage User Pools (管理用户池)。
3. 选择与您的 SageMaker 人力关联的用户池。
4. 在 General Settings (常规设置) 下，选择 Users and Groups (用户和组)，并执行下列操作之一：
   - 选择 Groups (组)，再选择要将用户添加到的组，然后选择 Add users (添加用户)。选择要添加的用户，方式是选择要添加的用户，方式是选择用户名右侧的加号图标。
   - 选择 Users (用户)，再选择要添加到用户组的用户，然后选择 Add to group (添加到组)。从下拉菜单中，选择组，然后选择 Add to group (添加到组)。

禁用工作人员并将其从工作团队中删除

禁用工作人员将使其不再接收作业。此操作不会从人力中删除工作人员，也不会从与工作人员关联的任何工作团队中删除工作人员。要在 Amazon Cognito 中删除工作团队中的一个用户，请从与工作团队关联的用户组中删除该用户。

要停用工作人员（Amazon Cognito 控制台）

1. 打开 Amazon Cognito 控制台：https://console.aws.amazon.com/cognito/.

2. 选择 Manage User Pools (管理用户池)。
3. 选择与您的 SageMaker 人力关联的用户池。
4. 在 General Settings (常规设置) 下，选择 Users and Groups (用户和组)。
5. 选择要禁用的用户。
6. 选择禁用用户.

可以通过选择 Enable User (启用用户) 来启用已禁用的用户。

从用户组中删除用户（Amazon Cognito 控制台）

1. 打开 Amazon Cognito 控制台：https://console.aws.amazon.com/cognito/.
2. 选择 Manage User Pools (管理用户池)。
3. 选择与您的 SageMaker 人力关联的用户池。
4. 在 General Settings (常规设置) 下，选择 Users and Groups (用户和组)。
5. 适用于用户选项卡上，选择X要从中删除用户的组右侧的图标。

创建和管理 OIDC IdP 人力

当您想要使用自己的 OIDC IdP 管理和身份验证员工时，使用 OpenID Connect (OIDC) 身份提供商 (IdP) 创建私人劳动力。个人员凭证和其他数据将保密。Ground Truth 和 Amazon A2I 只能查看您通过向这些服务发送的索赔提供的员工信息。要使用 OIDC IdP 创建员工队伍，您的 IdP 必须支持组因为 Ground Truth 和 Amazon A2I 将 IdP 中的一个或多个组映射到工作团队。要了解更多信息，请参阅 向 Ground Truth 和亚马逊 A2I 发送必填和可选索赔 (p. 474)。

如果您是 Ground Truth 或 Amazon A2I 的新用户，则可以通过创建私人工作团队并将自己添加为工作人员来测试工作人员 UI 和工作流程。创建标签作业或人工审核工作流程时，请使用此工作团队。首先，使用中的说明创建一个私人 OIDC IdP 员工创建私有人力（OIDC IdP）(p. 474). 下一步：请参阅管理私有人力（OIDC IdP）(p. 480)以了解如何创建工作团队。

主题

- 创建私有人力（OIDC IdP）(p. 474)
- 管理私有人力（OIDC IdP）(p. 480)

创建私有人力（OIDC IdP）

当您想使用自己的身份提供商对员工进行身份验证和管理时，使用 OpenID Connect (OIDC) 身份提供商 (IdP) 创建私人劳动力。使用此页面了解如何配置 IdP 以与亚马逊沟通 SageMaker Ground Truth（Ground Truth）或亚马逊 Augmented AI（Amazon A2I），并学习如何使用自己的 IdP 创建员工队伍。

要使用 OIDC IdP 创建员工队伍，您的 IdP 必须支持组因为 Ground Truth 和 Amazon A2I 使用您指定的一个或多个组来创建工作团队。您可以使用工作团队为标签作业和人工审核任务指定工作人员。因为团体不是标准声明，您的 IdP 对于一组用户（工作人员）可能有不同的命名约定。因此，您必须使用自定义声明确定工作人员所属的一个或多个用户组`sagemaker:groups`这是从你的 IdP 发送到 Ground Truth 或亚马逊 A2I。要了解更多信息，请参阅 向 Ground Truth 和亚马逊 A2I 发送必填和可选索赔 (p. 474)。

您可 IdP 使用 SageMaker API 操作`CreateWorkforce`. 创建私有人力后，该人力及其关联的所有工作团队和工作人员都可用于所有 Ground Truth 标记作业任务和 Amazon A2I 人工审核工作流程任务。要了解更多信息，请参阅 创建 OIDC IdP 人力 (p. 476)。

向 Ground Truth 和亚马逊 A2I 发送必填和可选索赔

当你使用自己的 IdP、Ground Truth 和亚马逊 A2I 使用`Issuer`、`ClientId`, 和`ClientSecret`通过从您的`AuthorizationEndpoint`.

Ground Truth 和亚马逊 A2I 将使用此代码从 IdP 那里获取自定义索赔`TokenEndpoint`要么`UserInfoEndpoint`. 您可以配置`TokenEndpoint`返回 JSON Web 令牌 (JWT) 或`UserInfoEndpoint`返回 JSON 对象。JWT 或 JSON 对象必须包含您指定的必需和可选声明。一个要求是一个键值对，其中包含有关工作人员的信息或有关 OIDC 服务的元数据。下表列出了必须包含的声明，这些声明可以选择包含在 IdP 返回的 JWT 或 JSON 对象中。

Note

您可以使用下表中的某些参数使用`:`或者`-`. 例如，您可以使用指定工作线程所属组`sagemaker:groups`要么`sagemaker-groups`在申请中。

名称	必填	接受的格式和值	说明	示例
`sagemaker:groups` 或者 `sagemaker-groups`	是	数据类型： 如果工作人员属于单个组，请使用字符串识别该组。 如果工作线程属于多个组，请使用最多包含 10 个字符串的列表。 允许使用的字符： 正则表达式：[\ p {L}\ p {M}\ p {N}\ p {L}\ p {P} + 配额： 每个工作人员 10 组 每组名称 63 个字符	将工作线程分配给一个或多个组。组用于将工作人员映射到工作团队中。	属于单个组的工作人员示例：`"work_team1"` 属于多个组的工作人员的示例：`["work_team1", "work_team2"]`
`sagemaker:sub` 或者 `sagemaker-sub`	是	数据类型： 字符串	这对于在 Ground Truth 平台内跟踪员工身份进行审计以及识别该员工处理的任务是必需的。 对于 ADFS：客户必须使用主安全标识符 (SID)。	`"111011101-123456789-368705`
`sagemaker:client_id` 或者 `sagemaker-client_id`	是	数据类型： 字符串 允许使用的字符： 正则表达式：[\ w+-] + QUOTE： 128 个字符	客户端 ID。必须为此客户端 ID 发放所有令牌。	`"00b600bb-1f00-05d0-bd00-00be00fbd0e0"`
`sagemaker:name` 或者 `sagemaker-name`	是	数据类型： 字符串	要在工作人员门户中显示的工作人员姓名。	`"Jane Doe"`

名称	必填	接受的格式和值	说明	示例
email	否	数据类型： 字符串	工作线程电子邮件。Ground Truth 使用此电子邮件通知员工他们已被邀请从事贴标任务。如果您为该工作人员所在的工作团队设置 Amazon SNS 主题，Ground Truth 还将使用此电子邮件在标签任务可用时通知您的员工。	"example-email@domain.com"
email_verified	否	数据类型： Bool 接受的值： True, False	指示用户电子邮件是否已验证。	True

以下是 JSON 对象语法的示例UserInfoEndpoint可以返回。

```
{
    "sub":"122",
    "exp":"10000",
    "sagemaker-groups":["group1","group2"]
    "sagemaker-name":"name",
    "sagemaker-sub":"122",
    "sagemaker-client_id":"123456"
}
```

Ground Truth 或亚马逊 A2I 比较中列出的组sagemaker:groups要么sagemaker-groups验证您的工作人员是否属于标签作业或人工审核任务中指定的工作组。验证工作团队后，标签或人工审核任务将发送给该工作人员。

创建 OIDC IdP 人力

您可以使用创建人力 SageMaker API 操作CreateWorkforce以及相关的特定于语言的 SDK。指定WorkforceName以及参数中有关 OIDC IDP 的信息OidcConfig. 建议您使用占位符重定向 URI 配置 OIDC，然后在创建员工后使用工作人员门户网址更新 URI。要了解更多信息，请参阅 配置 OIDC IdP (p. 477)。

以下显示了请求的示例。请参阅CreateWorkforce以了解更多有关此请求中每个参数的详情。

```
CreateWorkforceRequest: {
    #required fields
    WorkforceName: "example-oidc-workforce",
    OidcConfig: {
        ClientId: "clientId",
        ClientSecret: "secret",
        Issuer: "https://example-oidc-idp.com/adfs",
        AuthorizationEndpoint: "https://example-oidc-idp.com/adfs/oauth2/authorize",
        TokenEndpoint: "https://example-oidc-idp.com/adfs/oauth2/token",
        UserInfoEndpoint: "https://example-oidc-idp.com/adfs/oauth2/userInfo",
        LogoutEndpoint: "https://example-oidc-idp.com/adfs/oauth2/log-out",
        JwksUri: "https://example-oidc-idp.com/adfs/discovery/keys"
    },
```

```
    SourceIpConfig: {
        Cidrs: ["string", "string"]
    }
}
```

配置 OIDC IdP

如何配置 OIDC IdP 取决于您使用的 IdP 和业务需求。

配置 IdP 时，必须指定回调或重定向 URI。在 Ground Truth 或 Amazon A2I 对工作人员进行身份验证后，此 URI 将将工作人员重定向到工作人员门户，员工可以在其中访问标签或人工审核任务。要创建员工门户网址，您需要使用`CreateWorkforce`API 操作。具体来说，您必须使用所需的自定义 sagemaker 声明配置 OIDC IdP（有关更多详细信息，请参阅下一节）。因此，建议您使用占位符重定向 URI 配置 OIDC，然后在创建员工后更新 URI。请参阅创建 OIDC IdP 人力 (p. 476)了解如何使用此 API 创建员工队伍。

您可以在 SageMaker Ground Truth 控制台，或者使用 SageMaker API 操作，`DescribeWorkforce`. 工作人员门户网址位于`SubDomain`响应中的参数。

> **Important**
>
> 确保将员工子域添加到 OIDC IdP 允许列表中。当你将子域添加到允许列表时，它必须以`/oauth2/idpresponse`.

要在创建私人员工队伍后查看员工门户网址 (控制台)：

1. 打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中，选择 Labeling workforces (标记人力)。
3. 选择 Private (私有) 选项卡。
4. In私人劳动力总结你会看到为门户登录 URL 添加标记. 这是你的工作人员门户网址。

要在创建私人员工队伍 (API) 后查看员工门户网址：

当您使用创建私有人力时使用`CreateWorkforce`，您可以指定`WorkforceName`. 用这个名字来打电话`DescribeWorkforce`. 下表包含使用Amazon CLI和Amazon SDK for Python (Boto3).

SDK for Python (Boto3)

```
response = client.describe_workforce(WorkforceName='string')
print(f'The workforce subdomain is: {response['SubDomain']}')
```

Amazon CLI

```
$ C:\>  describe-workforce --workforce-name 'string'
```

验证您的 OIDC IdP 员工身份验证响应

创建 OIDC IdP 人力后，您可以使用以下步骤使用 cURL 验证其身份验证工作流程。此过程假定你有权访问终端，并且你已安装了 cURL。

要验证 OIDC IdP 授权响应：

1. 使用如下配置的 URI 获取授权码：

```
{AUTHORIZE ENDPOINT}?client_id={CLIENT ID}&redirect_uri={REDIRECT
 URI}&scope={SCOPE}&response_type=code
```

   a. Replace*{AUTHORIZE ENDPOINT}*使用 OIDC IdP 的授权终端节点。

b. Replace*{CLIENT ID}*使用 OAuth 客户端的客户端 ID。

c. Replace*{REDIRECT URI}*使用工作人员门户网址。如果尚未存在，则必须添加/oauth2/idpresponse到 URL 的末尾。

d. 如果你有自定义范围，请用它来替换*{SCOPE}*. 如果您没有自定义作用域，请替换*{SCOPE}*和openid.

以下是进行上述修改后 URI 的示例：

```
https://example.com/authorize?
client_id=f490a907-9bf1-4471-97aa-6bfd159f81ac&redirect_uri=https%3A%2F%2F
%2Fexample.labeling.sagemaker.aws
%2Foauth2%2Fidpresponse&response_type=code&scope=openid
```

2. 将步骤 1 中修改的 URI 复制并粘贴到浏览器中，然后按键盘上的 Enter 键。
3. 使用 IdP 进行身份验证。
4. 在 URI 中复制身份验证码查询参数。这个参数存在code=. 以下是响应具体形式的示例。在此示例中，复制code=MCNYDB...以及之后的一切。

```
https://example.labeling.sagemaker.aws/oauth2/idpresponse?code=MCNYDB....
```

5. 在进行下面列出的必要修改后，打开终端并输入以下命令：

```
curl --request POST \
  --url '{TOKEN ENDPOINT}' \
  --header 'content-type: application/x-www-form-urlencoded' \
  --data grant_type=authorization_code \
  --data 'client_id={CLIENT ID}' \
  --data client_secret={CLIENT SECRET} \
  --data code={CODE} \
  --data 'redirect_uri={REDIRECT URI}'
```

a. Replace*{TOKEN ENDPOINT}*使用 OIDC IdP 的令牌终端节点。

b. Replace*{CLIENT ID}*使用 OAuth 客户端的客户端 ID。

c. Replace*{CLIENT SECRET}*使用 OAuth 客户端提供的客户机密。

d. Replace*{CODE}*使用您在步骤 4 中复制的身份验证码查询参数。

e. Replace*{REDIRECT URI}*使用工作人员门户网址。

以下是进行上述修改后 cURL 请求的示例：

```
curl --request POST \
  --url 'https://example.com/token' \
  --header 'content-type: application/x-www-form-urlencoded' \
  --data grant_type=authorization_code \
  --data 'client_id=f490a907-9bf1-4471-97aa-6bfd159f81ac' \
  --data client_secret=client-secret \
  --data code=MCNYDB... \
  --data 'redirect_uri=https://example.labeling.sagemaker.aws/oauth2/idpresponse'
```

6. 此步取决于access_token你的 IdP 返回，一个纯文本访问令牌或 JWT 访问令牌。

- 如果您的 IdP 不支持 JWT 访问令牌，access_token可能是纯文本（例如，UUID）。您看到的响应可能类似于以下内容。在此情况下，请转到步骤 7。

```
{
  "access_token":"179c144b-fccb-4d96-a28f-eea060f39c13",
```

```
  "token_type":"Bearer",
  "expires_in":3600,
  "refresh_token":"ef43e52e-9b4f-410c-8d4c-d5c5ee57631a",
  "scope":"openid"
}
```

- 如果您的 IdP 支持 JWT 访问令牌，则步骤 5 应该生成 JWT 格式的访问令牌。例如，响应可能类似于以下内容：

```
{
    "access_token":"eyJh...JV_adQssw5c",
    "refresh_token":"i6mapTIAVSp2oJkgUnCACKKfZxt_H5MBLiqcybBBd04",
    "refresh_token_expires_in":6327,
    "scope":"openid",
    "id_token":"eyJ0eXAiOiJK9...-rDaQzUHl6cQQWNiDpWOl_lxXjQEvQ"
}
```

复制 JWT 并对其进行解码。你可以使用 python 脚本或第三方网站来解码它。例如，您可以访问该网站https://jwt.io/然后将 JWT 粘贴到编码框来解码它。

确保解码响应包含以下内容：

- 这些区域有：必需 SageMaker 表格中的索赔向 Ground Truth 和亚马逊 A2I 发送必填和可选索赔 (p. 474). 如果没有，则必须重新配置 OIDC IdP 以包含这些声明。
- 这些区域有：发布者您在设置 IdP 人力时指定。

7. 在终端中，在进行下面列出的必要修改后输入以下命令：

```
curl -X POST -H 'Authorization: Bearer {ACCESS TOKEN}' -d '' -k -v {USERINFO ENDPOINT}
```

a. Replace*{USERINFO ENDPOINT}*使用 OIDC IdP 的用户信息终端节点。
b. Replace*{ACCESS TOKEN}*在步骤 7 中收到的响应中使用访问令牌。这是的条目`"access_token"`参数。

以下是进行上述修改后 cURL 请求的示例：

```
 curl -X POST -H 'Authorization: Bearer eyJ0eX...' -d '' -k -v https://example.com/
userinfo
```

8. 对上述过程中最后一步的响应可能类似于以下代码块。

如果`access_token`在步骤 6 中返回的是纯文本，您必须验证此响应是否包含所需信息。在此情况下，响应必须包含必需 SageMaker 表格中的索赔向 Ground Truth 和亚马逊 A2I 发送必填和可选索赔 (p. 474). 例如，`sagemaker-groups` 和 `sagamaker-name`。

```
{
    "sub":"122",
    "exp":"10000",
    "sagemaker-groups":["group1","group2"]
    "sagemaker-name":"name",
    "sagemaker-sub":"122",
    "sagemaker-client_id":"123456"
}
```

后续步骤

使用 IdP 创建私人员队伍并验证了 IdP 身份验证响应后，就可以使用 IdP 组创建工作团队。要了解更多信息，请参阅 管理私有人力（ OIDC IdP ） (p. 480)。

您可以将员工对任务的访问限制为特定 IP 地址，也可以使用 SageMaker API。要了解更多信息，请参阅“使用 Amazon SageMaker API 管理私人劳动力 (p. 482)”。

管理私有人力（OIDC IdP）

使用 OpenID Connect (OIDC) 身份提供商 (IdP) 创建私人劳动力后，您可以使用 IdP 管理员工。例如，您可以通过 IdP 直接添加、删除工作线程和分组工作线程。

要将员工添加到 Amazon SageMaker Ground Truth（Ground Truth）标签作业或亚马逊 Augmented AI (Amazon A2I) 人工审核任务中，您可以使用 1-10 个 IdP 组创建工作团队，然后将该工作组分配给该工作或任务。通过在创建标签作业（Ground Truth）或人工审核工作流程（Amazon A2I）时指定该工作组，可以将工作团队分配给工作或任务。

您只能为每个贴标作业或人工审核工作流程分配一个团队。您可以使用同一个团队创建多个标签作业或人工审查任务。您还可以创建多个工作团队来处理不同的标签作业或人工审核任务。

先决条件

要使用 OIDC IdP 组创建和管理私人工作团队，首先必须使用 SageMaker API 操作创建员工`CreateWorkforce`. 要了解更多信息，请参阅 创建私有人力（OIDC IdP）(p. 474)。

添加工作团队

您可以使用 SageMaker 控制台创建一个私人工作团队，使用您的 OIDC IdP 员工标记人力下面的页面Ground Truth. 如果您正在创建 Ground Truth 标签作业，则还可以在创建标签作业时创建一个私人工作团队。

> Note
>
> 您可以在 SageMaker 控制台的 Ground Truth 区域为 Amazon A2I 创建和管理工作团队。

您还可以使用 SageMaker API 和关联的特定于语言的 SDK 来创建私有工作团队。

通过以下步骤了解如何使用 SageMaker 控制台和 API 创建私有工作团队。

在标签员工页面上创建私有工作团队（控制台）

1. 转到 SageMaker 控制台的 Ground Truth 区域：https://console.aws.amazon.com/sagemaker/groundtruth.
2. Select标记人力.
3. Select私密.
4. 在私有团队部分，选择创建私有团队.
5. 在详细团队部分中，输入团队名称.
6. 在添加人力部分中，输入单个用户组的名称。IdP 中与此组关联的所有工作人员都将添加到此工作团队中。
7. 要添加多个用户组，请选择添加新的用户组然后输入要添加到此工作团队的用户组的名称。每行输入一个用户组。
8. （可选）对于 Ground Truth 标签作业，如果您为智威汤逊中的员工提供电子邮件，则如果您选择 SNS 主题，则在新的标签任务可用时，Ground Truth 会通知员工。
9. Select创建私有团队.

要在创建 Ground Truth 标记作业时创建私有工作团队（控制台）

1. 转到 SageMaker 控制台的 Ground Truth 区域：https://console.aws.amazon.com/sagemaker/groundtruth.
2. Select标记作业.

3. 使用中的说明操作创建标记作业（控制台）(p. 353)创建标记作业。当你到达时停止工件第二页上的部分。
4. Select私密对于你的工作人员类型。
5. 输入团队名称.
6. 在添加人力在部分中，在下面输入单个用户组的名称用户组. IdP 中与此组关联的所有工作人员都将添加到此工作团队中。

   Important

   您为其指定的组名用户组必须与 OIDC IdP 中指定的组名称匹配。
7. 要添加多个用户组，请选择添加新的用户组然后输入要添加到此工作团队的用户组的名称。每行输入一个用户组。
8. 要创建标记作业，请完成所有剩余步骤。

您创建的私有团队将用于此标签作业，并在标记人力SageMaker 控制台的部分。

要使用 SageMaker API 创建私有工作团队

您可以使用 SageMaker API 操作创建私有工作团队`CreateWorkteam`.

使用此操作时，请在`OidcMemberDefinition`参数`Groups`.

Important

您为其指定的组名`Groups`必须与 OIDC IdP 中指定的组名称匹配。

例如，如果您的用户组名称为`group1`、`group2`, 和`group3`在 OIDC IdP 中，配置`OidcMemberDefinition`如下所示：

```
"OidcMemberDefinition": {
   "Groups": ["group1", "group2", "group3"]
 }
```

此外，您必须使用`WorkteamName`参数。

在工作团队中添加或删除 IdP 组

创建工作团队后，您可以使用 SageMaker API 来管理该工作团队。使用`UpdateWorkteam`操作来更新该工作组中包含的 IdP 用户组。

- 使用`WorkteamName`参数以标识要更新的工作团队。
- 使用此操作时，请在`OidcMemberDefinition`参数`Groups`. 如果用户组与工作团队关联并且您确实不将其包括在此列表中，则该用户组不再与此工作团队关联。

删除工作团队

您可以使用 SageMaker 控制台和 SageMaker API 删除工作团队。

在 SageMaker 控制台中删除私有工作团队

1. 转到 SageMaker 控制台的 Ground Truth 区域：https://console.aws.amazon.com/sagemaker/groundtruth.
2. Select标记人力.
3. Select私密.
4. 在私有团队部分中，选择要删除的工作团队。
5. 选择删除。

删除私有工作团队 (API)

您可以使用 SageMaker API 操作删除私有工作团队`DeleteWorkteam`.

管理个人工

当您使用自己的 OIDC IdP 创建员工时，您不能使用 Ground Truth 或 Amazon A2I 来管理个人员。

- 要将工作人员添加到工作团队，请将该工作人员添加到与该工作团队关联的组中。
- 要从工作组中删除工作人员，请从与该工作组关联的所有用户组中删除该工作人员。

更新、删除和描述你的员工

您可以使用 SageMaker API 更新、删除和描述 OIDC IdP 员工。您可以使用以下 API 操作列表，您可以用于管理人力。有关其他详细信息，包括如何找到员工名称，请参阅使用 Amazon SageMaker API 管理私人劳动力 (p. 482).

- `UpdateWorkforce`— 您可能希望更新使用自己的 OIDC IdP 创建的员工队伍，以指定不同的授权终端节点、令牌终端节点或颁发者。你可以更新中找到的任何参数`OidcConfig`使用此操作。

  只有在没有与员工关联的工作团队时，才能更新 OIDC IdP 配置。要了解如何删除工作团队，请参阅删除工作团队 (p. 481).
- `DeleteWorkforce`— 使用此操作删除您的私人劳动力。如果您有与员工关联的任何工作团队，则必须先删除这些工作团队，然后才能删除工作队伍。有关更多信息，请参阅 删除工作团队 (p. 481)。
- `DescribeWorkforce`— 使用此操作可以列出私人员工队伍信息，包括员工名称、亚马逊资源名称 (ARN) 以及允许的 IP 地址范围 (CIDR)（如果适用）。

使用 Amazon SageMaker API 管理私人劳动力

您可以使用 Amazon SageMaker API 操作管理、更新和删除私有人力。对于此页面上链接的每个 API 操作，您可以在另请参阅API 文档的部分。

查找人力

一些 SageMaker 员工相关的 API 操作需要您的员工姓名作为输入。Amazon Cognito 可 IdP 在Amazon区域使用`ListWorkforces`其中的 API 操作Amazon区域。

如果您使用自己的 OIDC IdP 创建了员工队伍，则可以在 SageMaker 控制台的 Ground Truth 区域中找到员工名称。

在 SageMaker 控制台中查找您的员工名称

1. 转到 SageMaker 控制台的 Ground Truth 区域：https://console.aws.amazon.com/sagemaker/groundtruth.
2. Select为人力贴标.
3. Select私密.
4. 在私人劳动力总结部分中，找到你的员工 ARN。您的员工名称位于此 ARN 的末尾。例如，如果 ARN 为`arn:aws:sagemaker:us-east-2:111122223333:workforce/example-workforce`，员工名称是`example-workforce`.

将工作人员对任务的访问限制为允许的 IP 地址

默认情况下，不限制人力访问特定 IP 地址。您可以使用`UpdateWorkforce`操作要求工作人员使用特定范围的 IP 地址（CIDR) 来访问任务。如果指定了一个或多个 CIDR，则试图使用指定范围之外的任何 IP 地址访

问任务的工作人员将被拒绝，并将在工作人员门户上收到 HTTP 204 无内容错误消息。您可以使用指定最多 10 个 CIDR 值。`UpdateWorkforce`.

将人力限制为一个或多个 CIDR 后，`UpdateWorkforce`列出所有允许的 cIDR。您也可以使用`DescribeWorkforce`操作来查看工作人员的所有允许的 cIDR。

更新 OIDC 身份提供商人力配置

您可能想要更新使用自己的 OIDC IdP 创建的员工队伍，以指定不同的授权终端节点、令牌终端节点或颁发者。你可以更新中找到的任何参数`OidcConfig`使用`UpdateWorkforce`operation.

> Important
>
> 只有在没有与员工关联的工作团队时，才能更新 OIDC IdP 配置。您可以使用删除私有工作组`DeleteWorkteam`operation.

删除私有人力

每个人力只能有一个私有人力Amazon区域。您可能需要将私有人力删除Amazon区域何时：

- 您希望使用新的 Amazon Cognito 用户池创建人力。
- 您已经使用 Amazon Cognito 创建了私人劳动力，并且想要使用自己的 OpenID Connect (OIDC) 身份提供商 (IdP) 创建员工队伍。

要删除私人员工队伍，请使用`DeleteWorkforce`API 操作。如果您有与员工关联的任何工作团队，则必须先删除这些工作团队，然后才能删除员工。您可以使用删除私有工作组`DeleteWorkteam`operation.

跟踪工作人员性能

Amazon SageMaker Ground Truth 将工作人员事件记录到 Amazon CloudWatch，例如当工作人员启动或提交任务时。使用 Amazon CloudWatch 指标来衡量和跟踪团队或单个工作人员的吞吐量。

> Important
>
> 工作人员事件跟踪不适用于 Amazon Advanced Augmented AI 人工审核工作流程。

启用追踪

在新工作团队的设置过程中，将创建 Amazon CloudWatch 工作人员事件记录的 Amazon CloudWatch 事件的权限。由于此功能是在 2019 年 8 月添加的，因此之前创建的工作团队可能没有正确的权限。如果您的所有工作团队都是在 2019 年 8 月之前创建的，请创建一个新的工作团队。它不需要任何成员，可以在创建后删除，但通过创建它，您可以建立权限并将权限应用于您的所有工作团队，无论这些团队何时创建都是如此。

检查日志

启用跟踪后，将记录工作人员的活动。打开 Amazon CloudWatch 控制台，然后选择。日志在导航窗格中。您应看到一个名为 /aws/sagemaker/groundtruth/WorkerActivity 的日志组。

每个完成的任务将由一个日志条目表示，其中包含有关工作人员、他们的团队、作业、接受任务的时间以及任务提交时间的信息。

Example 日志条目

```
{
  "worker_id": "cd449a289e129409",
  "cognito_user_pool_id": "us-east-2_IpicJXXXX",
  "cognito_sub_id": "d6947aeb-0650-447a-ab5d-894db61017fd",
  "task_accepted_time": "Wed Aug 14 16:00:59 UTC 2019",
  "task_submitted_time": "Wed Aug 14 16:01:04 UTC 2019",
```

```
  "task_returned_time": "",
  "task_declined_time": "",
  "workteam_arn": "arn:aws:sagemaker:us-east-2:############:workteam/private-crowd/Sample-
labeling-team",
  "labeling_job_arn": "arn:aws:sagemaker:us-east-2:############:labeling-job/metrics-demo",
  "work_requester_account_id": "############",
  "job_reference_code": "############",
  "job_type": "Private",
  "event_type": "TasksSubmitted",
  "event_timestamp": "1565798464"
}
```

每个事件中有用的数据点是 `cognito_sub_id`。您可以将其与单个工作人员匹配。

1. 打开位于 https://console.aws.amazon.com/sagemaker/ 的 Amazon SageMaker 控制台。
2. 在Ground Truth部分，选择人力.
3. 选择 Private (私有)。
4. 选择 Private teams (私有团队) 部分中的团队名称。
5. 在 Team summary (团队摘要) 部分中，选择 Amazon Cognito user group (Amazon Cognito 用户组)下标识的用户组。这会将您带到 Amazon Cognito 控制台中的该组。
6. Group (组) 页面列出了组中的用户。选择 Username (用户名) 列中任何用户的链接以查看有关该用户的更多信息，包括唯一的 sub (子) ID。

要获取有关团队的所有成员的信息，请使用ListUsers行动 (例子) 在 Amazon Cognito API 中。

使用日志指标

如果您不想编写自己的脚本来处理和可视化原始日志信息，则 Amazon CloudWatch 指标会为您提供有关工作人员活动的见解。

查看指标

1. 访问 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Metrics（指标）。
3. 选择`AWS/SageMaker/Workteam`命名空间，然后探索指标 (p. 2810). 例如，选择工作团队和人力使用指标，可以计算特定标记作业的每个提交任务的平均时间。

有关更多信息，请参阅使用 Amazon CloudWatch 指标。

为工作团队创建和管理 Amazon SNS 主题

当您想执行以下操作时，请使用本主题中的过程：

- 创建您希望现有工作团队订阅的主题。
- 在创建工作团队之前创建主题。
- 使用 API 调用创建或修改工作团队并指定主题 Amazon 资源名称 (ARN)。

如果您使用控制台创建工作团队，控制台会提供一个选项用于为团队创建新主题，这样您就无需执行这些步骤。

> Important
>
> 亚马逊 A2I 不支持 Amazon SNS 功能。如果您为工作团队订阅 Amazon SNS 主题，则工作人员只会收到有关 Ground Truth 标签作业的通知。员工不会收到有关新的 Amazon A2I 人工审核任务的通知。

创建 Amazon SNS 主题

为工作组通知创建 Amazon SNS 主题的步骤与中的步骤类似开始使用中的Amazon SNS 开发人员指南，除了一个重要的补充之外，您必须添加访问策略以便亚马逊SageMaker可以代表您向该主题发布消息。

在创建主题时添加策略

1. 通过以下网址打开 Amazon SNS 控制台：https://console.aws.amazon.com/sns/v3/home。
2. 在 Create topic (创建主题) 中，输入主题的名称，然后选择 Next steps (后续步骤)。
3. 在 Access policy (访问策略) 中，选择 Advanced (高级)。
4. 在 JSON editor (JSON 编辑器) 中，找到显示主题 ARN 的 `Resource` 属性。
5. 复制 `Resource` ARN 值。
6. 在最后一个右方括号 (`]`) 之前，添加以下策略。

```
    , {
        "Sid": "AwsSagemaker_SnsAccessPolicy",
        "Effect": "Allow",
        "Principal": {
            "Service": "sagemaker.amazonaws.com"
        },
        "Action": "sns:Publish",
        "Resource": "arn:partition:sns:region:111122223333:MyTopic", # ARN of the topic
 you copied in the previous step
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": "arn:partition:sagemaker:region:111122223333:workteam/
*" # Workteam ARN
            },
            "StringEquals": {
                "aws:SourceAccount": "111122223333" # SNS topic account
            }
        }
    }
```

7. 创建 主题。

创建主题后，它将显示在 Topics (主题) 摘要屏幕中。有关创建主题的更多信息，请参阅创建主题中的Amazon SNS 开发人员指南.

管理工作人员订阅

如果您在创建了工作团队之后，将工作团队订阅到某个主题，则在创建工作团队时添加到该团队的各个工作团队成员不会自动订阅主题。有关为工作人员的电子邮件地址订阅主题的信息，请参阅为终端节点订阅 Amazon SNS 主题中的Amazon SNS 开发人员指南.

工作人员自动订阅您主题的唯一情况是，当您在创建工作团队时创建或导入 Amazon Cognito 用户组。和您可以在创建该工作组时设置主题订阅。有关使用 Amazon Cognito 创建和管理工作团队的更多信息，请参阅创建工作团队（ Amazon Cognito 控制台 ） (p. 472).

Crowd HTML 元素参考

此功能在中国区域中不可用。

Crowd HTML Elements 是 Web 组件，是将 HTML 标记、CSS 和 JavaScript 功能抽象为一个或一组 HTML 标记的网络标准。向客户 Amazon SageMaker 供在 HTML 中设计其自己的自定义任务模板的能力。

作为起点，您可以使用通过以下 GitHub 存储库之一中的 Crowd HTML 元素构建的模板：

- 适用于 Amazon SageMaker Ground Truth 的示例任务 UI
- 适用于 Amazon Augmented AI (A2I) 的超过 60 个示例任务 UI

这些存储库包括专为音频、图像、文本、视频和其他类型的数据标记和注释任务而设计的模板。

有关如何在 Amazon SageMaker Ground Truth Growd 中实施自定义模板的更多信息，请参阅创建自定义标记工作流 (p. 325). 要了解有关 Amazon Awd AI 中自定义模板的更多信息，请参阅创建自定义 Worker 任务模板 (p. 2648).

Crowd SageMaker rowd HTML 元素

下面列出的一些 Crowd HTML 元素可让构建自定义模板的过程变得更轻松，并为工作人员提供熟悉的 UI。Ground Truth、Augmented AI 和 Mechanical Turk 支持这些元素。

主题

- crowd-alert (p. 487)
- crowd-badge (p. 487)
- crowd-button (p. 488)
- crowd-bounding-box (p. 489)
- crowd-card (p. 493)
- crowd-checkbox (p. 493)
- crowd-classifier (p. 495)
- 多类别分类 (p. 497)
- crowd-entity-annotation (p. 498)
- crowd-fab (p. 500)
- crowd-form (p. 501)
- crowd-icon-button (p. 502)
- crowd-image-classifier (p. 503)
- crowd-image-classifier-multi-select (p. 506)
- crowd-input (p. 507)
- crowd-instance-segmentation (p. 509)
- crowd-instructions (p. 512)
- crowd-keypoint (p. 513)
- crowd-line (p. 516)
- crowd-modal (p. 518)
- crowd-polygon (p. 519)
- crowd-line (p. 524)
- crowd-radio-button (p. 527)
- crowd-radio-group (p. 528)
- crowd-semantic-segmentation (p. 529)
- crowd-slider (p. 532)
- crowd-tab (p. 533)
- crowd-tabs (p. 533)

- crowd-text-area (p. 534)
- crowd-toast (p. 536)
- crowd-toggle-button (p. 536)

crowd-alert

提醒工作人员注意当前情况的消息。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-alert>` 元素的 Liquid 模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

dismissible

一个布尔值开关，如果存在，允许工作人员关闭消息。

type

一个字符串，它指定要显示的消息的类型。可能的值为“info”（默认值）、“success”、“error”和“warning”。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-badge

一个图标，悬浮在它要附加到的另一个元素的右上角。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-badge>` 元素的模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

for

一个字符串，它指定徽章附加到的元素的 ID。

icon

一个字符串，它指定徽章中要显示的图标。该字符串必须是开源 iron-icons 集中的图标的名称（预加载）或者是指向自定义图标的 URL。

此属性覆盖 label 属性。

以下是可用于在 <crowd-badge> HTML 元素中添加 iron-icon 的语法的示例。请将 *icon-name* 替换为要从该图标集中使用的图标的名称。

```
<crowd-badge icon="icon-name" for="short-instructions"/>
```

标签

要在徽章中显示的文本。建议使用三个或更少字符，因为文本太大将溢出徽章区域。可以通过设置 icon 属性显示图标而非文本。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-button

表示某项操作的样式按钮。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 <crowd-button> 元素的模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

disabled

一个布尔值开关，如果存在，将按钮显示为已禁用并防止点击。

form-action

一个开关，如果设置为“submit”，则提交其父 crowd-form (p. 501) 元素；如果设置为“reset”，则重置其父 <crowd-form> 元素。

href

指向在线资源的 URL。如果您需要样式为按钮的链接，则使用此属性。

icon

一个字符串，它指定按钮的文本旁边要显示的图标。该字符串必须是开源 iron-icons 集中的图标的名称（预加载）。例如，要插入搜索 iron-icon，请使用以下命令：

```
<crowd-button>
    <iron-icon icon="search"/>
</crowd-button>
```

该图标定位在文本的左侧或右侧，具体由 icon-align 属性指定。

要使用自定义图标，请参阅 icon-url。

icon-align

图标相对于按钮文本的左或右位置。默认为“左侧”。

icon-url

指向图标的自定义图像的 URL。可以使用自定义图像取代由 icon 属性指定的标准图标。

正在加载

一个布尔值开关，如果存在，将按钮显示为正处于加载状态。此属性优先于 disabled 属性（如果这两个属性都存在）。

target

当您使用 `href` 属性让按钮充当指向特定 URL 的超链接时，`target` 属性可选指向链接的 URL 应加载的框架或窗口。

variant

按钮的一般样式。对于主要按钮使用“primary”，对于辅助按钮使用“normal”，对于第三级按钮使用“link”，或者使用“icon”只显示无文本的图标。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-bounding-box

用于在图像上绘制矩形并为图像中每个矩形所包围的部分分配一个标签的小部件。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 <crowd-bounding-box> 元素的 Liquid 模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。有关更多示例，请参阅该 GitHub 存储库。

属性

此元素支持以下属性。

header

要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

initial-value

JSON 对象的数组，其中每个对象在组件加载时都设置一个边界框。数组中的每个 JSON 对象包含以下属性。可以调整通过 initial-value 属性设置的边界框，并通过工作人员答案输出中的 initialValueModified 布尔值跟踪工作人员答案是否已调整。

- 高度height 框的高度（以像素为单位）。
- 标签— 作为标记任务的一部分分配给框的文本。此文本必须与在 <crowd-bounding-box> 元素的 labels 属性中定义的其中一个标签相匹配。
- 左边left 框左上角与图像左侧的距离，以像素为单位。
- 顶部top 框左上角与图像顶部的距离，以像素为单位。
- widthwidth 框的宽度（以像素为单位）。

您可以使用 Liquid 模板语言从自定义模板中以前作业的清单文件中提取边界框初始值：

```
initial-value="[
  {% for box in task.input.manifestLine.label-attribute-name-from-prior-job.annotations
 %}
    {% capture class_id %}{{ box.class_id }}{% endcapture %}
    {% assign label = task.input.manifestLine.label-attribute-name-from-prior-job-
metadata.class-map[class_id] %}
    {
      label: {{label | to_json}},
      left: {{box.left}},
      top: {{box.top}},
      width: {{box.width}},
      height: {{box.height}},
    },
  {% endfor %}
 ]"
```

labels

一个 JSON 格式的字符串数组，其中每个字符串都是一个标签，工作人员可以将其分配给图像中由矩形包围的部分。限制：10 个标签。

名称

此小部件的名称。它用作小部件输入（以输出格式表示）的密钥。

src

在其上绘制边界框的图像的 URL。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：full-instructions (p. 491)、short-instructions (p. 491)

区域

此元素需要以下区域。

full-instructions

有关如何绘制边界框的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

输出

此元素支持以下输出。

boundingBoxes

JSON 对象的数组，其中每个对象指定已由工作人员创建的一个边界框。数组中的每个 JSON 对象包含以下属性。

- 高度height 框的高度（以像素为单位）。
- 标签— 作为标记任务的一部分分配给框的文本。此文本必须与在 <crowd-bounding-box> 元素的 labels 属性中定义的其中一个标签相匹配。
- 左边left 框左上角与图像左侧的距离，以像素为单位。
- 顶部top 框左上角与图像顶部的距离，以像素为单位。
- widthwidth 框的宽度（以像素为单位）。

inputImageProperties

一个 JSON 对象，它指定正在由工作人员注释的图像的维度。此对象包含以下属性。

- 高度height 图像的高度（以像素为单位）。
- widthwidth 图像的宽度（以像素为单位）。

Example ：示例元素输出

以下是此元素的常见使用情况的输出示例。

单一标签、单一框/多个标签、单一框

```
[
  {
    "annotatedResult": {
      "boundingBoxes": [
        {
          "height": 401,
          "label": "Dog",
          "left": 243,
          "top": 117,
          "width": 187
        }
      ],
```

```
      "inputImageProperties": {
        "height": 533,
        "width": 800
      }
    }
  }
]
```

单一标签、多个框

```
[
  {
    "annotatedResult": {
      "boundingBoxes": [
        {
          "height": 401,
          "label": "Dog",
          "left": 243,
          "top": 117,
          "width": 187
        },
        {
          "height": 283,
          "label": "Dog",
          "left": 684,
          "top": 120,
          "width": 116
        }
      ],
      "inputImageProperties": {
        "height": 533,
        "width": 800
      }
    }
  }
]
```

多个标签、多个框

```
[
  {
    "annotatedResult": {
      "boundingBoxes": [
        {
          "height": 395,
          "label": "Dog",
          "left": 241,
          "top": 125,
          "width": 158
        },
        {
          "height": 298,
          "label": "Cat",
          "left": 699,
          "top": 116,
          "width": 101
        }
      ],
      "inputImageProperties": {
        "height": 533,
        "width": 800
      }
    }
```

```
  }
]
```

您可以有多个标签，但只有所使用的标签会显示在输出中。

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-card

一个框，其中包含用于显示消息的已提升外观。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是为使用 `<crowd-card>` 元素的情感分析任务设计的模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

heading

在框顶部显示的文本。

image

指向要显示在框内的图像的 URL。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-checkbox

可以选中或取消选中的一个 UI 组件，让用户可以从一组选项中选择多个选项。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-checkbox>` 元素的 Liquid 模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

checked

一个布尔值开关，如果存在，将复选框显示为选中状态。

以下是用于默认选中复选框的语法的示例。

```
  <crowd-checkbox name="checkedBox" value="checked" checked>This box is checked</crowd-
checkbox>
```

disabled

一个布尔值开关，如果存在，将复选框显示为已禁用并防止选中复选框。

以下是用于禁用复选框的语法的示例。

```
  <crowd-checkbox name="disabledCheckBox" value="Disabled" disabled>Cannot be selected</
crowd-checkbox>
```

名称

一个字符串，用于标识工作人员提交的答案。此值将与 JSON 对象中指定答案的密钥匹配。

必需

一个布尔值开关，如果存在，要求工作人员提供输入。

以下是用于要求选中复选框的语法的示例。

```
  <crowd-checkbox name="work_verified" required>Instructions were clear</crowd-checkbox>
```

值

一个字符串，用作输出中复选框状态的名称。如果未指定，则默认为“on”。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

输出

提供 JSON 对象。`name` 字符串是对象名称，而 `value` 字符串是基于复选框状态的布尔值的属性名称；如果选中，则为 true，否则为 false。

Example ：示例元素输出

对于多个框使用相同的 **`name`** 值。

```
<!-- INPUT -->
<div><crowd-checkbox name="image_attributes" value="blurry"> Blurry </crowd-checkbox></div>
```

```
<div><crowd-checkbox name="image_attributes" value="dim"> Too Dim </crowd-checkbox></div>
<div><crowd-checkbox name="image_attributes" value="exposed"> Too Bright </crowd-
checkbox></div>
```

```
//Output with "blurry" and "dim" checked
[
  {
    "image_attributes": {
      "blurry": true,
      "dim": true,
      "exposed": false
    }
  }
]
```

请注意，所有三个颜色值都是单个对象的属性。

对于每个框使用不同的 **name** 值。

```
<!-- INPUT -->
<div><crowd-checkbox name="Stop" value="Red"> Red </crowd-checkbox></div>
<div><crowd-checkbox name="Slow" value="Yellow"> Yellow </crowd-checkbox></div>
<div><crowd-checkbox name="Go" value="Green"> Green </crowd-checkbox></div>
```

```
//Output with "Red" checked
[
  {
    "Go": {
      "Green": false
    },
    "Slow": {
      "Yellow": false
    },
    "Stop": {
      "Red": true
    }
  }
]
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-classifier

用于对非图像内容（如音频、视频或文本）进行分类的小部件。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用以下内容构建的 HTML 工作人员任务模板的示例。`crowd-classifier`. 此示例使用液体模板语言自动化：

- 在`categories`参数
- 正在分类的对象`classification-target`参数。

复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

categories

一个 JSON 格式的字符串数组，其中每个字符串都是工作人员可以分配给文本的一个类别。您应包含 “其他” 作为一个类别，否则工作人员可能无法提供答案。

header

要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

名称

此小部件的名称。它用作小部件输入（以输出格式表示）的密钥。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：目标分类 (p. 496)、full-instructions (p. 496)、short-instructions (p. 496)

区域

此元素支持以下区域。

目标分类

工作人员要分类的内容。这可以是纯文本或 HTML。如何使用 HTML 的示例包括但不限于嵌入视频或音频播放器、嵌入 PDF 或对两个或更多图像进行比较。

full-instructions

有关如何执行文本分类的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

输出

此元素的输出是使用指定的`name`value 作为属性名称，以及来自`categories`作为财产的价值。

Example ：示例元素输出

下面是此元素中的输出示例。

```
[
  {
    "<name>": {
      "label": "<value>"
    }
  }
]
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

多类别分类

用于将各种形式的内容（如音频、视频或文本）分类到一个或多个类别的小部件。要分类的内容被称为宾语.

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用此元素构建的 HTML 工作人员任务模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

`crowd-classifier-multi-select` 元素支持以下属性。每个属性都接受一个或多个字符串值。

categories

必填项。一个 JSON 格式的字符串数组，其中每个字符串都是工作人员可以分配给对象的一个类别。

header

必填项。要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

名称

必填项。此小部件的名称。在表单输出中，名称用作小部件输入的键。

exclusion-category

可选。JSON 格式的字符串，格式如下：`"{ text: 'default-value' }"`. 此属性设置默认值，在任何标签均不适用于工作人员 UI 中显示的对象时，工作人员可以选择该默认值。

元素层次结构

此元素具有以下父元素和子元素：

- 父元素：crowd-form (p. 501)
- 子元素：目标分类 (p. 496)、full-instructions (p. 496)、short-instructions (p. 496)

区域

此元素使用以下区域。

分类

工作人员要分类的内容。内容可以是纯文本，也可以是您在模板中使用 HTML 指定的对象。例如，您可以使用 HTML 元素包含视频或音频播放器、嵌入 PDF 文件或包含两个或更多图像的比较。

full-instructions

有关如何分类的一般说明。

short-instructions

特定于任务的重要说明。这些说明会突出显示。

输出

此元素的输出是一个使用指定的namevalue 作为属性名称，以及来自的字符串categories作为财产的价值。

Example ：示例元素输出

下面是此元素中的输出示例。

```
[
  {
    "<name>": {
        labels: ["label_a", "label_b"]
    }
  }
]
```

另请参阅

有关更多信息，请参阅下列内容：

- 文本分类（多标签）(p. 230)
- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-entity-annotation

用于标记较长文本中的单词、短语或字符串的小部件。工作人员选择一个标签，然后突出显示该标签适用的文本。

> 重要提示：包含小部件
>
> 请勿将 <crowd-entity-annotation> 元素和 <crowd-form> 元素一起使用。它包含自己的表单提交逻辑和 Submit (提交) 按钮。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 <crowd-entity-annotation> 元素的模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

header

要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

initial-value

一个 JSON 格式的对象数组，每个对象定义一个在初始化时应用到文本的注释。对象包含一个 label 值（与 labels 属性中的值匹配）、一个整数 startOffset 值（用于所标记范围的起始 unicode 偏移）以及一个整数 endOffset 值（用于结束 unicode 偏移）。

Example

```
[
  {
    label: 'person',
    startOffset: 0,
    endOffset: 16
  },
  ...
]
```

labels

JSON 格式的对象数组，每个对象包含下列内容：

- **label**（必需）：用于标识实体的名称。
- **fullDisplayName**（可选）：用于任务小部件中的标签列表。如果未指定，则默认为标签值。
- **shortDisplayName**（可选）：3 到 4 个字母的缩写，用于显示上述选定实体。如果未指定，则默认为标签值。

  强烈建议使用 **shortDisplayName**。

  根据选择所显示的值可能会重叠，在工作区中管理标记的实体时会造成困难。强烈建议为每个标签提供 3 到 4 个字符的 shortDisplayName，以防止重叠并确保工作区对于您的工作人员易于管理。

Example

```
[
  {
    label: 'person',
    shortDisplayName: 'per',
    fullDisplayName: 'person'
  }
]
```

名称

在 DOM 中用作小部件的名称。它还用作标签属性名称，采用输入和输出清单的形式。

text

要标注的文本。默认情况下，模板化系统转义引号和 HTML 字符串。如果您的代码已转义或者部分转义，请参阅变量筛选条件 (p. 329)以了解控制转义的更多方法。

元素层次结构

此元素具有以下父元素和子元素。

- 子元素：full-instructions (p. 499)、short-instructions (p. 500)

区域

此元素支持以下区域。

full-instructions

有关如何使用小部件的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

输出

此元素支持以下输出。

实体

指定注释的开始、结束和标签的 JSON 对象。此对象包含以下属性。

- 标签— 分配的标签。
- startOffset— 所选文本的开始的 Unicode 偏移。
- 末端偏移— 选择之后第一个字符的 Unicode 偏移。

Example ：示例元素输出

下面是此元素中的输出示例。

```
{
  "myAnnotatedResult": {
    "entities": [
      {
        "endOffset": 54,
        "label": "person",
        "startOffset": 47
      },
      {
        "endOffset": 97,
        "label": "event",
        "startOffset": 93
      },
      {
        "endOffset": 219,
        "label": "date",
        "startOffset": 212
      },
      {
        "endOffset": 271,
        "label": "location",
        "startOffset": 260
      }
    ]
  }
}
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-fab

一个浮动按钮，图像位于其中心。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是为使用 <crowd-fab> 元素的图像分类设计的 Liquid 模板的示例。该模板使用 JavaScript 以使工作人员能够报告工作人员 UI 问题。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

disabled

一个布尔值开关，如果存在，将浮动按钮显示为已禁用并防止点击。

icon

一个字符串，它指定要显示在按钮中心的图标。该字符串必须是开源 iron-icons 集中的图标的名称（预加载）或者是指向自定义图标的 URL。

以下是可用于在 <crowd-fab> HTML 元素中添加 iron-icon 的语法的示例。请将 *icon-name* 替换为要从该图标集中使用的图标的名称。

```
<crowd-fab "id="button1" icon="icon-name" title="Issue"/>
```

标签

一个字符串，包含可以使用的单个字符，而不是图标。表情符号或多个字符可能会导致按钮改而显示省略号。

标题

一个字符串，当鼠标悬停在按钮上时显示工具提示。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-form

所有自定义任务的表单包装程序。设置和实施重要的操作，以便正确地提交表单数据。

如果 <crowd-form> 元素内未包含类型为“submit”的 crowd-button (p. 488)，它将自动附加在 <crowd-form> 元素内。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 <crowd-form> 元素的图像分类模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：无
- 子元素：任一项UI 模板 (p. 485)元素

元素事件

crowd-form 元素扩展标准 HTML form 元素并继承其事件，例如 onclick 和 onsubmit。

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-icon-button

图像位于其中心的按钮。当用户触摸该按钮时，会从按钮中心发出波纹效果。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是为使用 <crowd-icon-button> 元素的图像分类设计的 Liquid 模板的示例。该模板使用 JavaScript 以使工作人员能够报告工作人员 UI 问题。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

disabled

一个布尔值开关，如果存在，将按钮显示为已禁用并防止点击。

icon

一个字符串，它指定要显示在按钮中心的图标。该字符串必须是开源 iron-icons 集中的图标的名称（预加载）或者是指向自定义图标的 URL。

以下是可用于在 <crowd-icon-button> HTML 元素中添加 iron-icon 的语法的示例。请将 *icon-name* 替换为要从该图标集中使用的图标的名称。

```
<crowd-icon-button id="button1" icon="icon-name" title="Issue"/>
```

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-image-classifier

用于分类图像的小部件。使用以下支持的图像格式之一：APNG、BMP、GIF、图标、JPEG、PNG、SVG。图片没有大小限制。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-image-classifier>` 元素的图像分类模板的示例。复制以下代码，并将其保存到具有扩展名的文件中`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素需要以下属性。

categories

一个 JSON 格式的字符串数组，其中每个字符串都是工作人员可以分配给图像的一个类别。您应包含“其他”作为一个类别，以便工作人员可以提供答案。您最多可指定 10 个类别。

header

要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

名称

此小部件的名称。它用作小部件输入（以输出格式表示）的密钥。

叠加

要叠加在源图像上的信息。这用于界限框、语义分段和实例分段任务的验证工作流。

它是一个 JSON 对象，其中包含一个以 camelCase 中的任务类型名称为键的对象。该键的值是包含上一任务中的标签和其他必要信息的对象。

示例为`crowd-image-classifier`下面是具有用于验证边界框任务的属性的元素：

```
<crowd-image-classifier
    name="boundingBoxClassification"
    header="Rate the quality of the annotations based on the background section
       in the instructions on the left hand side."
    src="https://i.imgur.com/CIPKVJo.jpg"
    categories="['good', 'bad', 'okay']"
    overlay='{
        "boundingBox": {
            labels: ["bird", "cat"],
            value: [
                {
                  height: 284,
                  label: "bird",
                  left: 230,
                  top: 974,
                  width: 223
                },
                {
                  height: 69,
                  label: "bird",
                  left: 79,
                  top: 889,
```

```
                  width: 247
                }
            ]
        },
    }'
> ... </crowd-image-classifier>
```

语义分段验证任务将使用overlay值如下所示：

```
<crowd-image-classifier
  name='crowd-image-classifier'
  categories='["good", "bad"]'
  src='URL of image to be classified'
  header='Please classify'
  overlay='{
    "semanticSegmentation": {
      "labels": ["Cat", "Dog", "Bird", "Cow"],
      "labelMappings": {
        "Bird": {
          "color": "#ff7f0e"
        },
        "Cat": {
          "color": "#2ca02c"
        },
        "Cow": {
          "color": "#d62728"
        },
        "Dog": {
          "color": "#2acf59"
        }
      },
      "src": "URL of overlay image",
    }
  }'
> ... </crowd-image-classifier>
```

实例分段任务将使用overlay值如下所示：

```
<crowd-image-classifier
  name='crowd-image-classifier'
  categories='["good", "bad"]'
  src='URL of image to be classified'
  header='Please classify instances of each category'
  overlay='{
    "instanceSegmentation": {
       "labels": ["Cat", "Dog", "Bird", "Cow"],
       "instances": [
        {
         "color": "#2ca02c",
         "label": "Cat"
        },
        {
         "color": "#1f77b4",
         "label": "Cat"
        },
        {
         "color": "#d62728",
         "label": "Dog"
        }
       ],
       "src": "URL of overlay image",
    }
  }'
```

```
> ... </crowd-image-classifier>
```

src

要分类的图像的 URL。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：full-instructions (p. 505)、short-instructions (p. 505)、工作人员评论 (p. 505)

区域

此元素使用以下区域。

full-instructions

关于工作人员如何对映像进行分类的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

工作人员评论

当您需要工作人员解释他们做出选择的原因时，请在验证工作流中使用此选项。使用开头标签和结束标签之间的文本，为工作人员提供有关注释中应包含哪些信息的说明。

它使用以下属性：

header

用于留下评论的行动呼吁的短语。用作添加注释的模态窗口的标题文本。

可选。默认为“添加注释”。

link-text

此文本显示在小组件中的类别下方。单击时，它会打开一个模式窗口，工作人员可以在其中添加注释。

可选。默认为“添加注释”。

占位符

注释文本区域中的示例文本，当工作人员开始键入时被覆盖。如果工作人员将字段留空，则不会显示在输出中。

可选。默认值待留空。

输出

此元素的输出是一个字符串，用于指定在 <crowd-image-classifier> 元素的 categories 属性中定义的其中一个值。

Example ：示例元素输出

下面是此元素中的输出示例。

```
[
  {
    "<name>": {
      "label": "<value>"
      "workerComment": "Comment - if no comment is provided, this field will not be
 present"
    }
  }
]
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-image-classifier-multi-select

一个小部件，用于将图像分类到一个或多个类别中。使用以下支持的图像格式之一：APNG、BMP、GIF、图标、JPEG、PNG、SVG。图片没有大小限制。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用此 crowd 元素构建的 HTML 工作人员任务模板的示例。复制以下代码，并将其保存到具有扩展名的文件中`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

`crowd-image-classifier-multi-select` 元素支持以下属性。每个属性都接受一个或多个字符串值。

categories

必填项。一个 JSON 格式的字符串数组，其中每个字符串都是工作人员可以分配给图像的一个类别。工作人员必须至少选择一个类别，且可以选择所有类别。

header

必填项。要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

名称

必填项。此小部件的名称。在表单输出中，名称用作小部件输入的键。

src

必填项。要分类的图像的 URL。

exclusion-category

可选。JSON 格式的字符串，格式如下："`{ text: 'default-value' }`". 此属性设置默认值，在任何标签均不适用于工作人员 UI 中显示的图像时，工作人员可以选择该默认值。

元素层次结构

此元素具有以下父元素和子元素：

- 父元素：crowd-form (p. 501)
- 子元素：full-instructions (p. 505)、short-instructions (p. 505)、工作人员评论 (p. 505)

区域

此元素使用以下区域。

full-instructions

关于工作人员如何对映像进行分类的一般说明。

short-instructions

特定于任务的重要说明。这些说明会突出显示。

输出

此元素的输出是一个字符串，指定在 `<crowd-image-classifier-multi-select>` 元素的 `categories` 属性中定义的一个或多个值。

Example ：示例元素输出

下面是此元素中的输出示例。

```
[
  {
    "<name>": {
        labels: ["label_a", "label_b"]
    }
  }
]
```

另请参阅

有关更多信息，请参阅下列内容：

- 图像分类（多标签）(p. 221)
- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-input

接受输入数据的框。

不能自结束

与 HTML 标准中的 `input` 元素不同，此元素不能通过在右方括号之前放入斜杠（例如，`<crowd-input ... />`）自结束。它必须后跟一个 `</crowd-input>` 以结束此元素。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-input>` 元素的 Liquid 模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

allowed-pattern

一个正则表达式，与 auto-validate 属性结合使用以忽略工作人员键入时的非匹配字符。

auto-focus

如果此值设置为 true，则浏览器在加载后将焦点放在输入区域内。这样，工作人员就可以直接键入，而无需先将其选中。

auto-validate

一个布尔值开关，如果存在，将开启输入验证。可以通过 error-message 和 allowed-pattern 属性修改验证器的行为。

disabled

一个布尔值开关，如果存在，将输入区域显示为已禁用。

error-message

如果验证失败，显示在输入字段下方左侧的文本。

标签

在文本字段内显示的字符串。

当工作人员开始在文本字段中键入或已设置 value 属性时，此文本会缩小并升至文本字段上方。

max-length

输入将接受的最大字符数。超出此限制的输入将被忽略。

min-length

字段中输入的最小长度

名称

设置要用于 DOM 中的输入和表单输出的名称。

占位符

一个字符串值，用作占位符文本（直至工作人员开始将数据输到输入中时才显示），它不用作默认值。

必需

一个布尔值开关，如果存在，要求工作人员提供输入。

type

选取一个字符串以设置输入的 HTML5 `input-type` 行为。示例包括 `file` 和 `date`。

值

一个预设值，如果工作人员未提供输入，则成为默认值。预设值显示在文本字段中。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

输出

提供 `name` 字符串作为属性名称，并将在字段中输入的文本作为其值。

Example ：示例 JSON 输出

多个元素的值在相同的对象中输出，其 name 属性值作为其属性名称。无输入的元素不会显示在输出中。例如，我们使用三个输入：

```
<crowd-input name="tag1" label="Word/phrase 1"></crowd-input>
<crowd-input name="tag2" label="Word/phrase 2"></crowd-input>
<crowd-input name="tag3" label="Word/phrase 3"></crowd-input>
```

如果只有两个具有输入，输出为：

```
[
  {
    "tag1": "blue",
    "tag2": "red"
  }
]
```

这意味着，任何构建用于解析这些结果的代码都应该能够对答案中每个输入是否存在作出应对。

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-instance-segmentation

用于标识一个图像中特定对象的各个实例并为每个标记的实例创建彩色叠加层的小部件。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用<crowd-instance-segmentation>. 复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

使用类似于以下内容的模板，允许员工添加自己的类别（标签）。

```
<script src="https://assets.crowd.aws/crowd-html-elements.js"></script>
<crowd-form>
  <crowd-instance-segmentation
    id="annotator"
    name="myTexts"
    src="{{ task.input.taskObject | grant_read_access }}"
    header="Click Instructions to add new labels."
    labels="['placeholder']"
  >
    <short-instructions>
      <h3>Add a label to describe each type of object in this image.</h3>
      <h3>Cover each instance of each object with a segmentation mask.</h3>
      <br>
      <h3>
        Add new label
      </h3>
      <crowd-input name="_customLabel" id="customLabel"></crowd-input>
      <crowd-button id="addLabel">Add</crowd-button>

      <br><br><br>
      <h3>
      Manage labels
```

```
      </h3>
      <div id="labelsSection"></div>
    </short-instructions>

    <full-instructions>
      Describe your task in more detail here.
    </full-instructions>
  </crowd-instance-segmentation>
</crowd-form>

<script>
  document.addEventListener('all-crowd-elements-ready', function(event) {
    document.querySelector('crowd-instance-segmentation').labels = [];
  });

  function populateLabelsSection() {
    labelsSection.innerHTML = '';
    annotator.labels.forEach(function(label) {
      const labelContainer = document.createElement('div');
      labelContainer.innerHTML = label + ' <a href="javascript:void(0)">(Delete)</a>';
      labelContainer.querySelector('a').onclick = function() {
        annotator.labels = annotator.labels.filter(function(l) {
          return l !== label;
        });
        populateLabelsSection();
      };
      labelsSection.appendChild(labelContainer);
    });
  }

  addLabel.onclick = function() {
    annotator.labels = annotator.labels.concat([customLabel.value]);
    customLabel.value = null;

    populateLabelsSection();
  };
</script>
```

属性

此元素支持以下属性。

header

要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

labels

一个 JSON 格式的字符串数组，其中每个字符串都是工作人员可以分配给图像中一个对象实例的标签。工作人员可以通过在工具中的标签下选择“添加实例”来为每个相关实例生成不同的叠加颜色。

名称

此小部件的名称。它用作标记数据（以输出格式表示）的键值。

src

要标记的图像的 URL。

initial-value

一个 JSON 对象，其中包含先前实例分割作业的颜色映射以及到该先前作业输出的覆盖图像的链接。当您希望工作人员验证先前标记作业的结果并在必要时进行调整时，请包括此内容。

属性将显示如下：

```
initial-value="{
  "instances": [
    {
      "color": "#2ca02c",
      "label": "Cat"
    },
    {
      "color": "#1f77b4",
      "label": "Cat"
    },
    {
      "color": "#d62728",
      "label": "Dog"
    }
  ],
  "src": {{ "S3 file URL for image" | grant_read_access }}
}"
```

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：full-instructions (p. 511)、short-instructions (p. 511)

区域

此元素支持以下区域。

full-instructions

有关如何执行图像分割的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

输出

此元素支持以下输出。

labeledImage

一个 JSON 对象，其中包含标签的 Base64 编码的 PNG。

实例

一个 JSON 数组，其中包含具有实例标签和颜色的对象。

- 颜色— 标签的 RGB 颜色的十六进制值。`labeledImage`PNG。
- 标签— 赋予使用该颜色的叠加层的标签。此值可能重复，因为标签的不同实例都用其唯一颜色标识。

inputImageProperties

一个 JSON 对象，它指定正在由工作人员注释的图像的维度。此对象包含以下属性。

- 高度— 图像的高度（以像素为单位）。
- width— width 图像的宽度（以像素为单位）。

Example ：示例元素输出

下面是此元素中的输出示例。

```
[
  {
    "annotatedResult": {
      "inputImageProperties": {
        "height": 533,
        "width": 800
      },
      "instances": [
        {
          "color": "#1f77b4",
          "label": "<Label 1>":
        },
        {
          "color": "#2ca02c",
          "label": "<Label 1>":
        },
        {
          "color": "#ff7f0e",
          "label": "<Label 3>":
        },
      ],
      "labeledImage": {
        "pngImageData": "<Base-64 Encoded Data>"
      }
    }
  }
]
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-instructions

一个元素，该元素在工作人员单击链接或按钮时，在以下三个选项卡页上显示说明：Summary (摘要)、Detailed Instructions (详细说明) 和 Examples (示例)。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-instructions>` 元素的 Liquid 模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

link-text

要显示的用于打开说明的文本。默认值为 Click for instructions (单击以获取说明)。

link-type

一个字符串，它指定说明的触发器类型。可能的值为“link”（默认值）和“button”。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

区域

此元素支持以下区域。

detailed-instructions

提供任务特定说明的内容。此内容显示在“Detailed Instructions”(详细说明) 选项卡页面上。

negative-example

提供任务完成不足示例的内容。此内容显示在“Examples”(示例) 选项卡页面上。此元素内可能会提供多个示例。

positive-example

提供任务完成得当的示例的内容。此内容显示在“Examples”(示例) 选项卡页面上。

short-summary

对要完成的任务作总结的简短语句。此内容显示在“Summary”(摘要) 选项卡页面上。此元素内可能会提供多个示例。

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-keypoint

生成在图像上选择和注释关键点的工具。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-keypoint>` 元素的 Liquid 模板的示例。复制以下代码，并将其保存到具有扩展名的文件中`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

header

要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

initial-value

一个采用 JSON 格式的要点数组，这些要点在启动应用于映像。例如：

```
initial-value="[
  {
    'label': 'Left Eye',
    'x': 1022,
    'y': 429
  },
  {
    'label': 'Beak',
    'x': 941,
    'y': 403
  }
]
```

> **Note**
>
> 请注意，此属性中使用的标签值必须在 `labels` 属性中具有匹配值，否则该点不会渲染。

labels

JSON 格式的字符串数组，用作关键点注释标签。

名称

一个字符串，用于标识工作人员提交的答案。此值将与 JSON 对象中指定答案的密钥匹配。

src

要注释的图像的源 URI。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：full-instructions (p. 514)、short-instructions (p. 514)

区域

此元素需要以下区域。

full-instructions

有关如何注释图像的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

输出

此元素支持以下输出。

inputImageProperties

一个 JSON 对象，它指定正在由工作人员注释的图像的维度。此对象包含以下属性。

- 高度— height 图像的高度（以像素为单位）。

- width— width 图像的宽度（以像素为单位）。

keypoints

一个 JSON 对象数组，包含关键点的坐标和标签。每个对象包含以下属性。

- 标签— label 分配给关键点的标签。
- x— x 关键点在图像上的 X 坐标，以像素为单位。
- y— y 关键点在图像上的 Y 坐标，以像素为单位。

> **Note**
>
> X 和 Y 坐标以图像左上角的 0,0 为原点。

Example ：示例元素输出

下面是使用此元素的输出示例。

```
[
  {
    "crowdKeypoint": {
      "inputImageProperties": {
        "height": 1314,
        "width": 962
      },
      "keypoints": [
        {
          "label": "dog",
          "x": 155,
          "y": 275
        },
        {
          "label": "cat",
          "x": 341,
          "y": 447
        },
        {
          "label": "cat",
          "x": 491,
          "y": 513
        },
        {
          "label": "dog",
          "x": 714,
          "y": 578
        },
        {
          "label": "cat",
          "x": 712,
          "y": 763
        },
        {
          "label": "cat",
          "x": 397,
          "y": 814
        }
      ]
    }
  }
]
```

您可能有多个可用标签，但只有被使用的标签会显示在输出中。

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-line

用于在图像上绘制线条的小部件。每条线都与一个标签相关联，输出数据将报告每条线的起点和终点。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 <crowd-line> 元素的 Liquid 模板的示例。复制以下代码，并将其保存到具有扩展名的文件中.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。有关更多示例，请参阅该 GitHub 存储库。

属性

此元素支持以下属性。

header

可选。要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

initial-value

可选。JSON 对象的数组，其中每个对象在组件加载时都设置一条线。数组中的每个 JSON 对象包含以下属性：

- 标签— 作为标记任务的一部分分配给行的文本。此文本必须与中定义的其中一个标签相匹配。标签的属性<crowd-line>元素。
- 顶点—x和y像素相对于图像左上角的起点和终点相对于线的起点和终点。

```
initial-value="{
    lines: [
    {
        label: 'sideline', // label of this line annotation
        vertices:[         // an array of vertices which decide the position of the line
        {
            x: 84,
            y: 110
        },
        {
            x: 60,
            y: 100
        }
        ]
    },
    {
        label: 'yardline',
        vertices:[
        {
            x: 651,
            y: 498
        },
        {
```

```
            x: 862,
            y: 869
        }
        ]
    }
   ]
}"
```

通过initial-value属性可以调整。工作人员的答案是否被调整通过initialValueModifiedworker 应答输出中的布尔值。

labels

必填项。一个 JSON 格式的字符串数组，其中每个字符串都是工作人员可以分配给行的一个标签。

限制:10 个标签

标签颜色

可选。字符串数组。每个字符串都是标签的十六进制（十六进制）代码。

名称

必填项。此小部件的名称。它用作小部件输入（以输出格式表示）的密钥。

src

必填项。要在其上绘制线的图像的 URL。

区域

此元素需要以下区域。

full-instructions

有关如何绘制线的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：short-instructions (p. 517)、full-instructions (p. 517)

输出

inputImageProperties

一个 JSON 对象，它指定正在由工作人员注释的图像的维度。此对象包含以下属性。

- 高度— 图像的高度（以像素为单位）。
- width— 图像的宽度（以像素为单位）。

lines

JSON 数组，其中包含具有线标签和顶点的对象。

- 标签— 给一行的标签。
- 顶点 —x和y像素相对于图像左上角的起点和终点相对于线的起点和终点。

Example ：示例元素输出

下面是此元素中的输出示例。

```
{
    "crowdLine": { //This is the name you set for the crowd-line
      "inputImageProperties": {
        "height": 1254,
        "width": 2048
      },
      "lines": [
        {
          "label": "yardline",
          "vertices": [
            {
              "x": 58,
              "y": 295
            },
            {
              "x": 1342,
              "y": 398
            }
          ]
        },
        {
          "label": "sideline",
          "vertices": [
            {
              "x": 472,
              "y": 910
            },
            {
              "x": 1480,
              "y": 600
            }
          ]
        }
      ]
    }
  }
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-modal

打开时在显示屏上弹出的一个小窗口。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是可与 <crowd-modal> 元素一起使用的语法示例。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

link-text

要显示的用于打开模态的文本。默认值为“Click to open modal”(单击以打开模态)。

link-type

一个字符串，它指定模态的触发器类型。可能的值为“link”（默认值）和“button”。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-polygon

用于在图像上绘制多边形并将标签分配给每个多边形中包含的图像部分的小部件。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-polygon>` 元素的 Liquid 模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

header

要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

labels

JSON 格式的字符串数组，其中每个字符串都是工作人员可以分配给多边形包含的图像部分的一个标签。

名称

此小部件的名称。它用作小部件输入（以输出格式表示）的密钥。

src

要在其上绘制多边形的图像的 URL。

initial-value

JSON 对象的数组，其中每个对象在组件加载时都定义一个要绘制的多边形。数组中的每个 JSON 对象包含以下属性。

- 标签— 标记任务的一部分分配给多边形的文本。此文本必须与在 <crowd-polygon> 元素的 labels 属性中定义的其中一个标签相匹配。
- 顶点— JSON 对象的数组。每个对象都包含多边形中某个点的 x 和 y 坐标值。

Example

`initial-value` 属性可能类似这样。

```
initial-value =
  '[
     {
     "label": "dog",
     "vertices":
       [
         {
            "x": 570,
            "y": 239
         },
        ...
         {
            "x": 759,
            "y": 281
         }
       ]
     }
  ]'
```

由于此 JSON 数组将位于 HTML 元素中，所以必须用单引号或双引号括起来。上面的示例使用单引号在 JSON 本身中封装 JSON 和双引号。如果您必须在 JSON 中混合使用单引号和双引号，请将其替换为 HTML 实体代码（`"` 表示双引号，`'` 表示单引号）以安全转义它们。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：full-instructions (p. 520)、short-instructions (p. 520)

区域

以下区域是必需的。

full-instructions

有关如何绘制多边形的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

输出

此元素支持以下输出。

polygons

JSON 对象的数组，其中每个对象描述已由工作人员创建的一个多边形。数组中的每个 JSON 对象包含以下属性。

- 标签— 标记任务的一部分分配给多边形的文本。
- 顶点— JSON 对象的数组。每个对象都包含多边形中某个点的 x 和 y 坐标值。图像左上角的坐标为 0,0。

inputImageProperties

一个 JSON 对象，它指定正在由工作人员注释的图像的维度。此对象包含以下属性。

- 高度— 图像的高度（以像素为单位）。
- width— 图像的宽度（以像素为单位）。

Example ：示例元素输出

以下是此元素的常见使用情况的输出示例。

单个标签，单个多边形

```
{
    "annotatedResult":
    {
      "inputImageProperties": {
        "height": 853,
        "width": 1280
      },
      "polygons":
      [
        {
          "label": "dog",
          "vertices":
          [
            {
              "x": 570,
              "y": 239
            },
            {
              "x": 603,
              "y": 513
            },
            {
              "x": 823,
              "y": 645
            },
            {
              "x": 901,
              "y": 417
            },
            {
              "x": 759,
              "y": 281
            }
          ]
        }
      ]
    }
  }
]
```

单个标签，多个多边形

```
[
  {
```

```
"annotatedResult": {
  "inputImageProperties": {
    "height": 853,
    "width": 1280
  },
  "polygons": [
    {
      "label": "dog",
      "vertices": [
        {
          "x": 570,
          "y": 239
        },
        {
          "x": 603,
          "y": 513
        },
        {
          "x": 823,
          "y": 645
        },
        {
          "x": 901,
          "y": 417
        },
        {
          "x": 759,
          "y": 281
        }
      ]
    },
    {
      "label": "dog",
      "vertices": [
        {
          "x": 870,
          "y": 278
        },
        {
          "x": 908,
          "y": 446
        },
        {
          "x": 1009,
          "y": 602
        },
        {
          "x": 1116,
          "y": 519
        },
        {
          "x": 1174,
          "y": 498
        },
        {
          "x": 1227,
          "y": 479
        },
        {
          "x": 1179,
          "y": 405
        },
        {
          "x": 1179,
          "y": 337
        }
```

```
          ]
        }
      ]
    }
  }
]
```

多个标签，多个多边形

```
[
  {
    "annotatedResult": {
      "inputImageProperties": {
        "height": 853,
        "width": 1280
      },
      "polygons": [
        {
          "label": "dog",
          "vertices": [
            {
              "x": 570,
              "y": 239
            },
            {
              "x": 603,
              "y": 513
            },
            {
              "x": 823,
              "y": 645
            },
            {
              "x": 901,
              "y": 417
            },
            {
              "x": 759,
              "y": 281
            }
          ]
        },
        {
          "label": "cat",
          "vertices": [
            {
              "x": 870,
              "y": 278
            },
            {
              "x": 908,
              "y": 446
            },
            {
              "x": 1009,
              "y": 602
            },
            {
              "x": 1116,
              "y": 519
            },
            {
              "x": 1174,
              "y": 498
            },
```

```
            {
              "x": 1227,
              "y": 479
            },
            {
              "x": 1179,
              "y": 405
            },
            {
              "x": 1179,
              "y": 337
            }
          ]
        }
      ]
    }
  }
]
```

您可以有多个标签，但只有所使用的标签会显示在输出中。

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-line

用于在图像上绘制折线或线条的小部件。每条折线都与一个标签关联，可以包括两个或多个顶点。折线可以与自身相交，其起点和终点可以放置在图像上的任意位置。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-polyline>` 元素的 Liquid 模板的示例。复制以下代码并将其保存到具有扩展名的文件中`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。有关更多示例，请参阅该 GitHub 存储库。

属性

此元素支持以下属性。

header

可选。要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

initial-value

可选。JSON 对象的数组，其中每个对象在组件加载时都设置一个折线。数组中的每个 JSON 对象包含以下属性：

- 标签— 作为标记任务的一部分分配给折线的文本。此文本必须与中定义的其中一个标签相匹配。标签的属性`<crowd-polyline>`元素。
- 顶点—`x`和`y`折线顶点相对于图像左上角的像素坐标。

```
initial-value= "{
```

```
    polylines: [
    {
        label: 'sideline', // label of this line annotation
        vertices:[         // an array of vertices which decide the position of the line
        {
            x: 84,
            y: 110
        },
        {
            x: 60,
            y: 100
        }
        ]
    },
    {
        label: 'yardline',
        vertices:[
        {
            x: 651,
            y: 498
        },
        {
            x: 862,
            y: 869
        },
        {
            x: 1000,
            y: 869
        }
        ]
    }
   ]
}"
```

通过initial-value属性可以调整。工作人员的答案是否被调整通过initialValueModifiedworker 应答输出中的布尔值。

labels

必填项。一个 JSON 格式的字符串数组，其中每个字符串都是工作人员可以分配给该行的一个标签。

限制:10 个标签

标签颜色

可选。字符串数组。每个字符串都是标签的十六进制（十六进制）代码。

名称

必填项。此小部件的名称。它用作小部件输入（以输出格式表示）的密钥。

src

必填项。要在其上绘制折线的图像的 URL。

区域

此元素需要以下区域。

full-instructions

有关如何绘制折线的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：short-instructions (p. 526)、full-instructions (p. 525)

输出

inputImageProperties

一个 JSON 对象，它指定正在由工作人员注释的图像的维度。此对象包含以下属性。

- 高度— 图像的高度（以像素为单位）。
- width— 图像的宽度（以像素为单位）。

多边形

JSON 数组，其中包含具有多边形的标签和顶点的对象。

- 标签— 给一行的标签。
- 顶点 —x和y折线顶点相对于图像左上角的像素坐标。

Example ：示例元素输出

下面是此元素中的输出示例。

```
{
   "crowdPolyline": { //This is the name you set for the crowd-polyline
     "inputImageProperties": {
       "height": 1254,
       "width": 2048
     },
     "polylines": [
       {
         "label": "sideline",
         "vertices": [
           {
             "x": 651,
             "y": 498
           },
           {
             "x": 862,
             "y": 869
           },
           {
             "x": 1449,
             "y": 611
           }
         ]
       },
       {
         "label": "yardline",
         "vertices": [
```

```
            {
              "x": 1148,
              "y": 322
            },
            {
              "x": 1705,
              "y": 474
            },
            ,
            {
              "x": 1755,
              "y": 474
            }
          ]
        }
      ]
    }
  }
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-radio-button

可以选中或取消选中的按钮。当单选按钮位于单选按钮组内时，任何时候只能选中组内的一个单选按钮。下面的示例介绍了如何在 `crowd-radio-group` 元素内配置 `crowd-radio-button` 元素。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是可与 `<crowd-radio-button>` 元素一起使用的语法示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

在此 GitHub 示例中的自定义工作人员任务模板中，可以看到上一个实例：实体识别标记的作业自定义模板。

人群 HTML 元素单选按钮不支持 HTML 标签，`required`. 要需要选择单选按钮，请使用`<input type="radio">`元素来创建单选按钮并添加`required`标签。这些区域有：`name`所有人的属性`<input>`属于同一组单选按钮的元素必须相同。例如，以下模板要求用户在`animal-type`在提交之前组。

属性

此元素支持以下属性。

checked

一个布尔值开关，如果存在，将单选按钮显示为选中状态。

disabled

一个布尔值开关，如果存在，将按钮显示为已禁用并防止选中按钮。

名称

一个字符串，用于标识工作人员提交的答案。此值将与 JSON 对象中指定答案的密钥匹配。

Note

如果您在 crowd-radio-group (p. 528) 元素之外使用按钮，但具有相同的 `name` 字符串和不同的 `value` 字符串，则输出中的 `name` 对象将对于每个 `value` 字符串包含一个布尔值。为确保一个组中只有一个按钮被选中，请将其作为 crowd-radio-group (p. 528) 元素的子元素并使用不同的名称值。

值

元素的布尔值的属性名称。如果未指定，它将“on”用作默认值，例如 `{ "<name>": { "<value>": <true or false> } }`。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-radio-group (p. 528)
- 子元素：无

输出

输出具有以下模式的对象：`{ "<name>": { "<value>": <true or false> } }`. 如果你在外面使用按钮crowd-radio-group (p. 528)元素，但是具有相同的`name`字符串和不同`value`字符串，name 对象将包含每个对象的布尔值`value`字符串。为确保按钮组中只有一个按钮处于选中状态，需要将其设置为 crowd-radio-group (p. 528) 元素的子元素并使用不同的名称值。

Example 此元素的示例输出

```
[
  {
    "btn1": {
      "yes": true
    },
    "btn2": {
      "no": false
    }
  }
]
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-radio-group

一组单选按钮。只能选中组内的一个单选按钮。选择一个单选按钮将清除同一组内先前选择的任何单选按钮。有关使用 `crowd-radio-group` 元素的自定义 UI 模板的示例，请参阅此实体识别标记作业自定义模板。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是可与 `<crowd-radio-group>` 元素一起使用的语法示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素不支持任何特殊属性。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：crowd-radio-button (p. 527)

输出

输出表示其中 crowd-radio-button (p. 527) 元素的对象数组。

Example 元素输出示例

```
[
  {
    "btn1": {
      "yes": true
    },
    "btn2": {
      "no": false
    }
  }
]
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-semantic-segmentation

一个用于分割图像和向每个图像分段分配标签的小部件。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 <crowd-semantic-segmentation> 元素的 Liquid 模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

header

要在图像上方显示的文本。通常是一个针对工作人员的问题或简单说明。

initial-value

一个 JSON 对象，其中包含先前语义分割作业的颜色映射以及到该先前作业输出的覆盖图像的链接。当您希望工作人员验证先前标记作业的结果并在必要时进行调整时，请包括此内容。

属性将显示如下：

```
  initial-value='{
    "labelMappings": {
        "Bird": {
          "color": "#ff7f0e"
        },
        "Cat": {
          "color": "#2ca02c"
        },
        "Cow": {
          "color": "#d62728"
        },
        "Dog": {
          "color": "#1f77b4"
        }
      },
    "src": {{ "S3 file URL for image" | grant_read_access }}
  }'
```

当使用 Ground Truth内置的任务类型和注释整合（其中有多个工作人员标记为单个图像），标签映射包含在单个工作人员输出记录中，但整体结果表示为internal-color-map在综合结果中。

您可以使用 Liquid 模板语言在自定义模板中将 internal-color-map 转换为 label-mappings：

```
initial-value="{
  'src' : '{{ task.input.manifestLine.label-attribute-name-from-prior-job|
 grant_read_access }}',
  'labelMappings': {
     {% for box in task.input.manifestLine.label-attribute-name-from-prior-job-
metadata.internal-color-map %}
       {% if box[1]['class-name'] != 'BACKGROUND' %}
         {{ box[1]['class-name'] | to_json }}: {
           'color': {{ box[1]['hex-color'] | to_json }}
         },
       {% endif %}
     {% endfor %}
   }
}"
```

labels

一个 JSON 格式的字符串数组，其中每个字符串都是工作人员可以分配给图像分段的一个标签。

名称

此小部件的名称。它用作小部件输入（以输出格式表示）的密钥。

src

要分割的图像的 URL。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：full-instructions (p. 531)、short-instructions (p. 531)

区域

此元素支持以下区域。

full-instructions

有关如何执行图像分割的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

输出

此元素支持以下输出。

labeledImage

一个 JSON 对象，其中包含标签的 Base64 编码的 PNG。

labelMappings

一个 JSON 对象，其中包含用分隔标签命名的对象。

- 颜色— 标签的 RGB 颜色的十六进制值`labeledImage`PNG。

initialValueModified

表示初始值是否已被修改的布尔值。仅当输出来自调整任务时，才会包含此内容。

inputImageProperties

一个 JSON 对象，它指定正在由工作人员注释的图像的维度。此对象包含以下属性。

- 高度— 图像的高度（以像素为单位）。
- width— 图像的宽度（以像素为单位）。

Example ：示例元素输出

下面是此元素中的输出示例。

```
[
  {
    "annotatedResult": {
      "inputImageProperties": {
        "height": 533,
        "width": 800
      },
      "labelMappings": {
        "<Label 2>": {
          "color": "#ff7f0e"
        },
        "<label 3>": {
          "color": "#2ca02c"
        },
        "<label 1>": {
          "color": "#1f77b4"
        }
      },
      "labeledImage": {
        "pngImageData": "<Base-64 Encoded Data>"
      }
    }
  }
]
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-slider

一个带有滑动旋钮的栏，让工作人员可以通过移动旋钮从值范围中选择一个值。对于反映强度级别的设置（如音量、亮度或颜色饱和度），使用滑块是一个不错的选择。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-slider>` 元素的调查模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

disabled

一个布尔值开关，如果存在，将滑块显示为已禁用。

editable

一个布尔值开关，如果存在，将显示一个向上/向下按钮，可以选择此按钮来选择值。

通过向上/向下按钮选择值是通过移动滑块上的旋钮来选择值的替代方法。滑块上的旋钮将与向上/向下按钮选择同步移动。

max

一个数值，它指定滑块上的最大值。

min

一个数值，它指定滑块上的最小值。

名称

一个字符串，用于标识工作人员提交的答案。此值将与 JSON 对象中指定答案的密钥匹配。

pin

一个布尔值开关，如果存在，将随旋钮移动而显示旋钮上方的当前值。

必需

一个布尔值开关，如果存在，要求工作人员提供输入。

secondary-progress

当与 `crowd-slider-secondary-color` CSS 属性一起使用时，进度条的颜色将随由 `secondary-progress` 表示的点相应改变。例如，如果用于表示流视频上的进度，`value` 将表示查看器位于视频时间线中的位置。`secondary-progress` 值将表示时间线上视频已缓冲到的点。

step

一个数值，它指定滑块上可选值之间的差值。

值

一个预设值，如果工作人员未提供输入，则成为默认值。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-tab

样式看起来与具有以下信息的选项卡相似的组件。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 `<crowd-tab>` 元素的示例模板。复制以下代码，并将其保存到具有扩展名的文件中。`.html`. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

header

显示在选项卡上的文本。通常是一些简短的描述性名称，表示选项卡下包含的信息。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-tabs (p. 533)
- 子元素：无

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-tabs

选项卡式信息的容器。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 <crowd-tabs> 元素的示例模板。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素没有属性。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：crowd-tab (p. 533)

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-text-area

用于文本输入的字段。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是一个 Liquid 模板的示例，该模板旨在转录使用 <crowd-text-area> 元素的音频剪辑。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

allowed-pattern

一个正则表达式，与 auto-validate 属性结合使用以忽略工作人员键入时的非匹配字符。

auto-focus

一个布尔值开关，如果存在，加载时将游标放在此元素中，以便用户可以立即开始键入，而不必先在元素内单击。

auto-validate

一个布尔值开关，如果存在，将开启输入验证。可以通过 error-message 和 allowed-pattern 属性修改验证器的行为。

char-counter

一个布尔值开关，如果存在，将一个小的文本字段放在元素的右下角，同时显示元素内的字符数。

disabled

一个布尔值开关，如果存在，将输入区域显示为已禁用。

error-message

如果验证失败，显示在输入字段下方左侧的文本。

标签

在文本字段内显示的字符串。

当工作人员开始在文本字段中键入或已设置 value 属性时，此文本会缩小并升至文本字段上方。

max-length

一个整数，指定元素允许的最大字符数。键入或粘贴的超出最大值的字符将被忽略。

max-rows

一个整数，指定 crowd-text-area 内允许的文本行的最大数量。正常情况下，元素将扩展以容纳新行。如果设置了此属性，则在行数超过此属性后，内容将向上滚出视图外，并出现一个滚动条控件。

name

一个字符串，用于表示输出中元素的数据。

占位符

以占位符文本形式呈现给用户的字符串。它在用户在输入区域中放入内容后消失。

rows

一个整数，指定文本行中元素的高度。

值

一个预设值，如果工作人员未提供输入，则成为默认值。预设值显示在文本字段中。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

输出

此元素将 `name` 作为属性名称输出，将元素的文本内容作为值输出。文本中的回车表示为 `\n`。

Example 此元素的示例输出

```
[
  {
    "textInput1": "This is the text; the text that\nmakes the crowd go wild."
  }
]
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-toast

显示屏上临时出现的小通知。只显示一个 crowd-toast。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下是使用 <crowd-toast> 元素的 Liquid 模板的示例。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

duration

一个数字，它指定在屏幕上显示通知的持续时间（以毫秒为单位）。

text

通知中要显示的文本。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

crowd-toggle-button

充当打开/关闭开关的按钮，用于切换状态。

请参阅中使用此 Crowd HTML 元素的 HTML 模板的交互式示例CodePen.

以下示例说明了可用于使用 <crowd-toggle-button> HTML 元素的不同方法。复制以下代码，并将其保存到具有扩展名的文件中。.html. 在任何浏览器中打开该文件，以进行预览并与该模板进行交互。

属性

此元素支持以下属性。

checked

一个布尔值开关，如果存在，显示按钮切换到“ON”位置。

disabled

一个布尔值开关，如果存在，将按钮显示为已禁用并防止切换。

invalid

当处在关闭位置时，使用此属性的按钮将显示为警报颜色。标准为红色，但可以在 CSS 中进行更改。切换为打开后，此按钮将与处于打开位置的其他按钮显示为相同的颜色。

名称

一个字符串，用于标识工作人员提交的答案。此值与 JSON 对象中指定答案的密钥匹配。

必需

一个布尔值开关，如果存在，要求工作人员提供输入。

值

输出中用作元素布尔状态的属性名称的值。如果未提供，则默认为“on”。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素：crowd-form (p. 501)
- 子元素：无

输出

此元素输出 `name` 作为对象的名称，其中包含 `value` 作为属性名称，并将元素的状态作为属性的布尔值。如果未指定此元素的值，则属性名称默认为“on”。

Example 此元素的示例输出

```
[
  {
    "theToggler": {
      "on": true
    }
  }
]
```

另请参阅

有关更多信息，请参阅下列内容。

- 使用 Amazon SageMaker 标记数据的 Ground Truth (p. 205)
- Crowd HTML 元素参考 (p. 485)

Augmented AI Crowd HTML 元素

以下 Crowd HTML 元素仅可用于亚马逊 Augmented AI 人工工作流程任务。

主题

- crowd-textract-documentation (p. 538)

- crowd-rekognition-detect-moderation-labels (p. 540)

crowd-textract-documentation

一个小部件，用于启用对 Amazon Textract 文档分析结果的人工审核。

属性

此元素支持以下属性。

header

这是将显示为标题的文本。

src

这是指向要由工作人员分析的图像的链接。

initialValue

这将为工作人员 UI 中提供的属性设置初始值。

以下是 `initialValue` 输入的示例：

```
[
            {
                "blockType": "KEY_VALUE_SET",
                "confidence": 38.43309020996094,
                "geometry": {
                    "boundingBox": {
                        "width": 0.32613086700439453,
                        "weight": 0.0942094624042511,
                        "left": 0.4833833575248718,
                        "top": 0.5227988958358765
                    },
                    "polygon": [
                        {"x": 0.123, "y": 0.345}, ...
                    ]
                }
                "id": "8c97b240-0969-4678-834a-646c95da9cf4",
                "relationships": [
                    {
                        "type": "CHILD",
                        "ids": [
                            "7ee7b7da-ee1b-428d-a567-55a3e3affa56",
                            "4d6da730-ba43-467c-a9a5-c6137ba0c472"
                        ]
                    },
                    {
                        "type": "VALUE",
                        "ids": [
                            "6ee7b7da-ee1b-428d-a567-55a3e3affa54"
                        ]
                    }
                ],
                "entityTypes": [
                    "KEY"
                ],
                "text": "Foo bar"
            },
]
```

blockTypes

这将决定工作人员可以执行的分析的类型。当前仅支持 `KEY_VALUE_SET`。

keys

这将指定工作人员可以添加的新键和关联的文本值。`keys` 的输入值可以包括以下元素：

- `importantFormKey` 接受字符串，并且用于指定单个键。
- `importantFormKeyAliases` 可用来指定别名，这些别名是提供的键的可接受替代项。使用此元素可标识键的替代拼写或表示形式。此参数接受包含一个或多个字符串的列表。

以下是 `keys` 的输入示例。

```
[
      {
        importantFormKey: 'Address',
        importantFormKeyAliases: [
          'address',
          'Addr.',
          'Add.',
        ]
      },
      {
        importantFormKey: 'Last name',
        importantFormKeyAliases: ['Surname']
      }
]
```

no-key-edit

这将防止工作人员编辑通过 `initialValue` 传递的注释的键。这将防止工作人员编辑已在文档中检测到的键。该项为必填项。

no-geometry-edit

这将防止工作人员编辑通过 `initialValue` 传递的注释的多边形。例如，这将防止工作人员编辑给定键周围的边界框。该项为必填项。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素 — crowd-form
- 子元素 —full-instructions (p. 539)、short-instructions (p. 539)

区域

此元素支持以下区域。您可以在这些区域中使用自定义 HTML 和 CSS 代码来设置针对工作人员的说明的格式。例如，使用 `short-instructions` 部分可提供有关如何完成任务的正确示例和错误示例。

full-instructions

有关如何使用小部件的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

使用 crowd 元素的工作人员模板示例

使用此 crowd 元素的工作人员模板示例如下所示。

输出

下面是此元素中的输出示例。您可以在 Amazon Textract 中查找此输出的详细说明AnalyzeDocumentAPI 文档。

```
{
  "AWS/Textract/AnalyzeDocument/Forms/V1": {
    blocks: [
      {
        "blockType": "KEY_VALUE_SET",
        "id": "8c97b240-0969-4678-834a-646c95da9cf4",
        "relationships": [
          {
            "type": "CHILD",
            "ids": ["7ee7b7da-ee1b-428d-a567-55a3e3affa56", "4d6da730-ba43-467c-a9a5-
c6137ba0c472"]
          },
          {
            "type": "VALUE",
            "ids": ["6ee7b7da-ee1b-428d-a567-55a3e3affa54"]
          }
        ],
        "entityTypes": ["KEY"],
        "text": "Foo bar baz"
      }
    ]
  }
}
```

crowd-rekognition-detect-moderation-labels

一个小部件，用于启用对 Amazon Rekognition 图像审核结果的人工审核。

属性

此元素支持以下属性。

header

这是将显示为标题的文本。

src

这是指向要由工作人员分析的图像的链接。

categories

这支持 `categories` 作为字符串数组或 对象数组，其中每个对象都具有一个 `name` 字段。

如果类别作为对象提供，则以下情况适用：

- 显示的类别是 `name` 字段的值。
- 返回的答复包含任何选定类别的完整 对象。

如果类别作为字符串提供，则以下情况适用：

- 返回的答复是选择的所有字符串的数组。

exclusion-category

通过设置此属性，您可以在 UI 中的类别下创建一个按钮。

- 当用户选择该按钮时，将取消选择并禁用所有类别。
- 再次选择该按钮会重新启用类别，以便用户可以选择它们。
- 如果您在选择该按钮后提交，则会返回一个空数组。

元素层次结构

此元素具有以下父元素和子元素。

- 父元素 — crowd-form
- 子元素 —full-instructions (p. 541)、short-instructions (p. 541)

Amazon 地区

此元素支持以下 Amazon 区域。您可以在这些区域中使用自定义 HTML 和 CSS 代码来设置针对工作人员的说明的格式。例如，使用 `short-instructions` 部分可提供有关如何完成任务的正确示例和错误示例。

full-instructions

有关如何使用小部件的一般说明。

short-instructions

在醒目位置显示的重要的任务特定说明。

具有 crowd 元素的示例工作人员模板

使用 crowd 元素的工作人员模板示例如下所示。

输出

下面是此元素中的输出示例。有关此输出的详细信息，请参阅 Amazon RekognitionDetectModerationLabelsAPI 文档。

```
{
  "AWS/Rekognition/DetectModerationLabels/Image/V3": {
    "ModerationLabels": [
        { name: 'Gore', parentName: 'Violence' },
        { name: 'Corpses', parentName: 'Violence' },
    ]
  }
}
```

准备和分析数据集

使用亚马逊导入、准备、转换、可视化和分析数据 SageMaker 数据牧马人。您可以将 Data Wrangler 集成到机器学习工作流程中，以简化和简化数据预处理和功能工程，只需几乎没有编码即可。您还可以添加自己的 Python 脚本和转换以自定义数据准备工作流。

从 Amazon S3 导入数据，Amazon Athena，并使用 Data Wrangler 创建复杂的机器学习数据准备工作流程，其中包括内置和自定义的数据转换和分析，包括功能目标泄露和快速建模。

定义数据准备工作流程之后，或数据流，你可以将它集成 SageMaker 正在处理， SageMaker 管道 SageMaker 功能商店，简化了处理、共享和存储 ML 训练数据的任务。您还可以将数据流导出到 python 脚本并创建自定义机器学习数据准备管道。

有关更多信息，请参阅使用亚马逊准备 ML 数据 SageMaker DATA Wrangler (p. 556)。

主题

- 检测训练前数据偏差 (p. 542)
- 使用亚马逊准备 ML 数据 SageMaker DATA Wrangler (p. 556)
- 使用 Studio 笔记本大规模准备数据 (p. 691)

检测训练前数据偏差

法律、政策和计算机科学等学科都对算法偏见、歧视、公平性和相关主题进行了研究。如果计算机系统歧视某些个人或个人群体，可能会被视为有偏见。支持这些应用程序的机器学习模型从数据中学习，这些数据可能反映差异或其他固有的偏见。例如，培训数据可能不足以反映各种人口群体，或者可能包含有偏见的标签。根据表现出这些偏见的数据集训练的机器学习模型最终可能会学习它们，然后在预测中重现甚至加剧这些偏见。机器学习领域提供了一个机会，通过检测偏见并在机器学习生命周期的每个阶段进行测量，来解决偏见。你可以使用亚马逊 SageMaker 澄清以确定用于训练模型的数据是否对任何偏见进行编码

可以在训练之前和训练后测量偏差，并在将模型部署到端点进行推理之后，根据基线进行监控。训练前偏差指标旨在在用于训练模型之前检测和衡量原始数据中的偏差。所使用的指标与模型无关，因为它们不依赖于任何模型输出。但是，有不同的公平概念需要采取不同的偏见措施。亚马逊 SageMaker 澄清提供了偏见指标来量化各种公平性标准。

有关偏置指标的其他信息，请参阅金融 Machine Learning 的公平性措施.

亚马逊 SageMaker 澄清偏见和公平性条款

SageMaker 澄清使用以下术语来讨论偏见和公平性。

功能

表格数据栏中包含的个人可衡量的属性或观察到的现象的特征。

Label

是训练机器学习模型的目标功能。称作观察标签要么观察结果.

预测的标签

模型所预测的标签。也称作预测结果.

示例

由要素值和标签值描述的观测实体，包含在表格数据的行中。

数据集

样本的集合。

偏差

不同群体（如年龄或收入等级）的训练数据或模型预测行为的不平衡。偏见可能是用于训练模型的数据或算法造成的。例如，如果机器学习模型主要根据中年人的数据进行培训，那么在进行涉及年轻人和老年人的预测时，可能会不那么准确。

偏置指标

返回指示潜在偏差水平的数值的函数。

偏差报告

给定数据集的偏差指标集合，或者数据集和模型的组合。

正标签值

标签值对样本中观察到的人口统计群体有利。换句话说，将样本指定为具有正结果.

负标签值

对样本中观察到的人口统计群体不利的标签值。换句话说，将样本指定为具有负结果.

组变量

数据集的类别列，用于组成用于衡量条件人口差异 (CDD) 的子组。仅对于辛普森的悖论而言，此指标是必需的。

分面

包含测量偏差的属性的列或要素。

分面值

偏见可能偏向或不赞成的属性的特征值。

预测的概率

正如模型所预测的那样有正面或负面结果的概率。

示例笔记本

亚马逊 SageMaker 澄清提供了以下示例笔记本用于偏差检测：

- 利用亚马逊进行可解释性和偏见检测 SageMaker 澄清— 使用 SageMaker 澄清以创建处理作业，以检测偏差并使用功能归因解释模型预测。

此笔记本已验证可在亚马逊上运行 SageMaker 仅限 Studio。如果您需要有关如何在亚马逊打开笔记本电脑的说明 SageMaker 工作室，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86). 如果系统会提示您选择内核，请选择Python 3（数据科学）.

主题

- 测量训练前偏置 (p. 544)
- 为训练前数据中的偏差生成报告 SageMaker 工作室 (p. 551)

测量训练前偏置

衡量机器学习模型中的偏差是减少偏差的第一步。每种偏见措施都对应于不同的公平概念。即使考虑简单的公平概念，也会导致许多适用于各种情况的不同措施。例如，考虑年龄的公平性，为了简单起见，中年和其他年龄组是两个相关的人口统计数据，称为方面. 在贷款机器学习模式的情况下，我们可能希望小型企业贷款发放给两种人口统计数量相同的数量。或者，在处理求职者时，我们可能希望看到雇用的每个人口中有相同数量的成员。但是，这种方法可能会假定两个年龄组的人数相等适用于这些工作，因此我们可能希望以适用的数字为条件。此外，我们可能不想考虑是否适用同等数字，而是考虑我们是否有相同数量的合格申请人。或者，我们可能会认为公平是在两个年龄人口统计中，合格申请人的接受率相同，或者，申请人的拒绝率相等，或两者兼而有。您可以在感兴趣的属性上使用具有不同比例数据的数据集。这种不平衡可能会混淆你选择的偏见衡量标准。模型在对一个方面进行分类时可能比另一个方面更准确。因此，您需要选择在概念上适合应用程序和情况的偏见指标。

我们使用以下符号来讨论偏差指标。此处描述的概念模型用于二进制分类，其中事件被标记为样本空间中只有两种可能的结果，称为正（值为 1）和负（值为 0）。这个框架通常可以直接扩展到多类别分类，或者在需要时可扩展到涉及持续有价值结果的情况。在二进制分类案例中，正标签和负面标签将分配给原始数据集中记录的结果，以获取受欢迎的方面一个而且对于一个不喜欢的方面D. 这些标签 y 被称为观察的标签将它们与预测的标签'是由机器学习模型在机器学习生命周期的训练或推断阶段分配的。这些标签用于定义概率分布 $P_a(y)$ 和 $P_d(y)$ 他们各自的方面结果。

- 标签：
  - y 表示训练数据集中事件结果的 n 个观测标签。
  - y 表示经过训练的模型对数据集中 n 个观测到的标签的预测标签。
- 结果：
  - 样本的积极结果（值为 1），例如申请接受。
    - $n^{(1)}$是观察到的积极结果（接受）标签的数量。
    - $n'^{(1)}$是积极结果（接受）的预计标签数量。
  - 样本的负面结果（值为 0），例如申请拒绝。
    - $n^{(0)}$是观察到的负面结果（拒绝）标签的数量。
    - $n'^{(0)}$是负面结果（拒绝）的预计标签数量。
- 分面值：
  - 特点一个— 定义偏见偏向的人口群体的功能值。
    - n_a是偏爱的小面值观察到的标签数量：$n_a n = {}_a^{(1)} + n_a^{(0)}$价值方面的正面和负面观察到的标签的总和一个.
    - n'_a是偏爱的刻面值的预测标签数量：$n'_a n'{}_a^{(1)} + n'{}_a^{(0)}$面值的正面和负面预测结果标签的总和一个. 请注意 $n'_a n = {}_a$.
  - 特点D— 定义偏向不利的人口群体的功能值。
    - n_d是不喜欢的小面值观察到的标签数量：$n_d n = {}_d^{(1)} + n_d^{(0)}$小平面值观测到的正值和负面标签的总和D.
    - n'_d是不喜欢的平面值的预测标签数量：$n'_d n'{}_d^{(1)} + n'{}_d^{(0)}$小面值的正值和负面预测标签的总和D. 请注意 $n'_d n = {}_d$.
- 标记的小面数据结果的概率分布：
  - $P_a(y)$ 是刻面观测到的标签的概率分布一个. 对于二进制标记的数据，此分布由刻面样本数量的比率给出一个对总数标记为积极结果，$P_a y^1 n)_a^{(1)}/n_a$，以及结果为负面的样本数量与总数之比，$P_a y^0 n)_a^{(0)}/n_a$.
  - $P_d(y)$ 是刻面观测到的标签的概率分布D. 对于二进制标记的数据，此分布由平面中的样本数量给出D对总数标记为积极结果，$P_d y^1 n) = {}_d^{(1)}/n_d$，以及结果为负面的样本数量与总数之比，$P_d y^0 n) = {}_d^{(0)}/n_d$.

对受人口差异偏差偏差的数据进行培训的模型可能会学习甚至加剧 为了在花费资源对模型进行模型训练之前识别数据中的偏差， SageMaker 澄清提供了数据偏差指标，您可以在训练前对原始数据集进行计算 所有训

练前指标都与模型无关，因为它们不依赖于模型输出，因此对任何模型都有效。第一个偏见指标检查面面不平衡，但不是结果。它决定了应用程序所需的培训数据量在多大程度上在不同方面具有代表性。剩余的偏差指标以不同的方式比较结果标签的分布情况一个和D在数据中。超过负值的指标可以检测到负面偏差。下表包含快速指导的备忘单以及指向培训前偏差指标的链接。

预训练偏置指标

偏置指标	描述	示例问题	解释指标值
班级失衡 (CI) (p. 547)	衡量不同面值之间成员数量的不平衡。	由于没有足够的中年人口数据，是否会存在基于年龄的偏见？	标准化范围：[-1, +1] 解释： • 正值表示方面一个数据集中有更多训练样本。 • 接近零的值表示小平面在数据集中的训练样本数量中是平衡的。 • 负值表示方面D数据集中有更多训练样本。
标签比例的差异 (DPL) (p. 547)	衡量不同方面价值之间积极结果的不平衡。	由于数据中刻面值的偏差标记，机器学习预测中是否存在基于年龄的偏见？	标准化二进制和多类别平面标签的范围：[-1, +1] 连续标签的范围：(-∞, +∞) 解释： • 正值表示刻面一个具有较高比例的积极结果。 • 接近零的值表示各个方面之间的积极结果比例更相等。 • 负值表示刻面D具有较高比例的积极结果。
KullBack-Leibler 发散（KL）(p. 548)	衡量不同方面的结果分布在熵上相互差异的程度。	不同人口群体的贷款申请结果分配有多不同？	二进制、多类别、连续的范围：[0, +∞) 解释： • 接近零的值表示标签的分布情况类似。 • 正值表示标签分布差异，差异越大，正值越大。
Jensen-Sannon 背离（JS）(p. 548)	衡量不同方面的结果分布在熵上相互差异的程度。	不同人口群体的贷款申请结果分配有多不同？	二进制、多类别、连续的范围：[0, +∞) 解释： • 接近零的值表示标签的分布情况类似。

偏置指标	描述	示例问题	解释指标值
			• 正值表示标签分布差异，差异越大，正值越大。
L_p-规范（LP）(p. 548)	测量与数据集中不同方面相关的结果的不同人口分布之间的 p 范数差异。	对于不同的人口统计数据，贷款申请结果的分配有多不同？	二进制、多类别、连续的范围：[0, +∞) 解释： • 接近零的值表示标签的分布情况类似。 • 正值表示标签分布差异，差异越大，正值越大。
总变化距离 (TVD) (p. 549)	测量 L 的一半$_1$-与数据集中不同方面相关的结果的不同人口分布之间的标准差异。	对于不同的人口统计数据，贷款申请结果的分配有多不同？	二进制、多类别和连续结果的范围：[0, +∞) • 接近零的值表示标签的分布情况类似。 • 正值表示标签分布差异，差异越大，正值越大。
科尔莫哥罗夫-斯米尔诺夫（KS）(p. 549)	测量数据集中不同方面的分布结果之间的最大差异。	哪些大学申请结果显示了按人口群体划分的最大差异？	二进制、多类别和连续结果的 KS 值范围：[0, +1] • 接近零的值表示标签在所有结果类别中的小面之间均匀分布。 • 接近一个值表示一个类别的标签都在一个方面，因此非常不平衡。 • 间歇性值表示最大标签不平衡的相对程度。
有条件的人口统计差异 (CDD) (p. 550)	衡量整个不同方面之间的结果差异，但也可以按子组衡量结果的差距。	有些群体的大学入学成绩被拒绝的比例是否超过他们的录取比例？	CDD 范围：[-1, +1] • 正值表示一种结果，其中面D被拒绝超过接受。 • 接近零表示平均来说没有人口差异。 • 负值表示一个结果，其中面一个被拒绝超过接受。

有关偏置指标的其他信息，请参阅金融 Machine Learning 的公平性措施.

主题

- 班级失衡 (CI) (p. 547)

- 标签比例的差异 (DPL) (p. 547)
- KullBack-Leibler 发散（KL）(p. 548)
- Jensen-Sannon 背离（JS）(p. 548)
- Lp-规范（LP）(p. 548)
- 总变化距离 (TVD) (p. 549)
- 科尔莫哥罗夫-斯米尔诺夫（KS）(p. 549)
- 有条件的人口统计差异 (CDD) (p. 550)

班级失衡 (CI)

分面值时会出现类别失衡 (CI) 偏差D与另一个方面相比，训练样本较少一个在数据集中。这是因为模型最好适合较大的面而牺牲较小的面面，因此可能会导致小平面的训练误差较高D. 模型也面临过度拟合较小的数据集的风险较高，这可能会导致小方面的测试错误更大D. 考虑机器学习模型主要根据中年人的数据进行培训的例子（方面 a），在进行涉及年轻人和老年人的预测时可能会不那么准确（方面 d）。

（标准化）小面不平衡衡量标准的公式：

$$CI = (n)_a\text{-}n_dn)/(n_a+n_d)$$

n 在哪里$_a$是 Facet 的成员数量一个和 n_d分面的数量D. 它的值在时间间隔 [-1, 1] 范围内。

- 正 CI 值表示方面一个数据集中有更多训练样本，值为 1 表示数据仅包含小平面的成员一个.
- CI 值接近零表示小平面之间的成员分布更为平等，值为零表示小平面之间的分配完全相等，并表示训练数据中样本的平衡分布。
- 负 CI 值表示方面D数据集中有更多训练样本，值为 -1 表示数据仅包含小平面的成员d.
- 接近 -1 或 1 的极端值之一的 CI 值都非常不平衡，并且存在进行偏见预测的巨大风险。

如果发现各方面之间存在严重的小面不平衡，则可能需要在样本上训练模型之前重新平衡样本。

标签比例的差异 (DPL)

标签比例的差异（DPL）将观察到的结果的比例与平面的正标签比例进行比较。D观察到的结果中有正面标签的比例一个在训练数据集中。例如，你可以用它来比较中年人的比例（平面）一个）和其他年龄组（方面）D) 已批准用于金融贷款。机器学习模型试图尽可能密切地模拟训练数据决策。因此，在 DPL 较高的数据集上训练的机器学习模型可能会在 future 的预测中反映同样的不平衡。

标签比例差异的公式如下：

$$DPL = (q_a\text{-}q_d)$$

其中：

- $Q_an_a^{(1)}/n_a$是平面的比例一个他们的标签值为 1。例如，获得贷款批准的中年人口比例。n 在此$_a^{(1)}$表示 facet 的成员数量一个谁得到了积极的结果和 n_a是 Facet 的成员数量一个.
- $Q_dn_d^{(1)}/n_d$是平面的比例D他们的标签值为 1。例如，中年人口以外获得贷款批准的人口比例。n 在此$_d^{(1)}$表示小平面的成员数D谁得到了积极的结果和 n_d是小平面的成员数量D.

如果 DPL 足够接近 0，那么我们说人口平等已实现。

对于二进制和多类别平面标签，DPL 值在间隔内 (-1, 1) 范围。对于连续标签，我们设置了将标签折叠为二进制标签的阈值。

- 正 DPL 值表示该方面一个与方面相比，是具有更高的积极成果比例D.

- 接近零的 DPL 值表示各个方面之间的积极结果比例更相等，值为零表示完美的人口统计平等。
- 负 DPL 值表示该方面D与方面相比，积极成果的比例更高一个.

大规模的 DPL 是否存在问题，因情况而异。在一个有问题的情况下，高规模的 DPL 可能是数据中潜在问题的信号。例如，DPL 较高的数据集可能反映了对基于年龄的人口群体的历史偏见或偏见，这些偏见或偏见对于模型学习来说是不可取的。

KullBack-Leibler 发散（KL）

KullBack-Leiber 背离度（KL）衡量了观察到的刻面的标签分布程度一个P, $P_a(y)$, 不同于刻面的分布DP, $P_d(y)$。它也称为 P 的相对熵。$_a(y)$ 关于 $P_d(y)$ 并量化从 P 移动时丢失的信息量$_a(y)$ 到 $P_d(y)$。

KullBack-Leiber 背离的公式如下：

$$L(P)_a|P_d) = \Sigma_y P_a y) * Log [P_a(y) / P_d(y)]$$

这是对概率 P 之间的对数差异的期望$_a(y)$ 和 $P_d(y)$，其中预期由概率 P 加权$_a(y)$。这不是分布之间的真正距离，因为它是不对称的，不能满足三角形的不平等。该实现使用自然对数，以 nats 为单位给予 KL。使用不同的对数基数可以得出成比例的结果，但单位不同。例如，使用底数 2 给 KL 以位为单位。

例如，假设一组贷款申请人的批准率为 30%（方面）D）和其他申请人的批准率（小面）一个) 是 80%。KullBack-Leibler 公式为您提供了刻面的标签分布差异一个从分面D如下所示：

$$吉隆坡 = 0.8 * \ln (0.8/0.3) + 0.2 * \ln (0.2/0.7) = 0.53$$

这里的公式中有两个术语，因为在本例中标签是二进制的。除了二进制标签之外，此度量还可以应用于多个标签。例如，在大学入学情况中，假设申请人可以被分配三个类别标签之一：$y_i y \{y_0 y_1 y_2\}$ = {已拒绝、等候名单、已接受}。

二进制结果、多类别和连续结果的 KL 指标值范围为 [0, +∞)。

- 接近零的值意味着结果在不同方面的分布类似。
- 正值意味着标签分布差异，分歧越积极就越大。

Jensen-Sannon 背离（JS）

Jensen-Shannon 背离（JS）衡量不同方面的标签分布在熵上相互差异的程度。它基于 KullBack-Leiber 背离，但它是对称的。

延森-香农背离的公式如下：

$$JS = ½ * P [L(P)_a || P) + L(P_d || P)]$$

哪里 $P = ½ (P_a + _d)$，跨方面的平均标签分布一个和D.

二进制、多类别、连续结果的 JS 值范围为 [0, ln (2))。

- 接近零的值表示标签的分布情况类似。
- 正值意味着标签分布差异，分歧越积极就越大。

该指标表明其中一个标签在各个方面是否存在很大差异。

L_p-规范（LP）

L_p-norm (LP) 测量训练数据集中观察到的标签的小面分布之间的 p 范数距离。该指标是非负面的，因此无法检测到反向偏差。

L 的公式$_p$-norm 如下所示：

$$L_pP(P)_aP, P_d) = (\Sigma_y|P_a\text{-}P_d||^P)^{p\ 1/p}$$

其中点 x 和 y 之间的 p 范数距离定义如下：

$$L_p(x, y) = (|x)_1\text{-}Y_1|^P + |x_2\text{-}Y_2|^P + \ldots + |x_n\text{-}Y_n|^P)^{p\ 1/p}$$

2 标准是欧几里得标准。假设你有三个类别的结果分布，例如$_i$y {$y_0y_1y_2$} = 在大学招生多类别场景中 {已接受、等候名单、拒绝}。分面的结果计数之间的差异的平方和一个和D. 生成的欧几里得距离的计算方式如下：

$$L_2P(P)_aP, P_d) = [(n_a^{0)}\text{-}n_d^{0)})^2 + (n_a^{(1)}\text{-}n_d^{(1)})^2 + (n_a^{2)}\text{-}n_d^{2)})^2]^{1/2}$$

其中：

- $n_a^{i)}$是第 i 个类别结果的数量一个: 例如 $n_a^{0)}$是方面的数量一个接受。
- $n_d^{i)}$是第 i 个类别结果的数量D: 例如 $n_d^{2)}$是方面的数量D拒绝。

二进制、多类别和连续结果的 LP 值范围为 [0, √2)，其中：

- 接近零的值表示标签的分布情况类似。
- 正值意味着标签分布差异，分歧越积极就越大。

总变化距离 (TVD)

总变化距离数据偏差指标 (TVD) 为 L 的一半$_1$-规范。TVD 是方面标签结果的概率分布之间可能最大的差异一个和D. L_1-norm 是 Hamming 距离，这是一个指标，通过确定将一个字符串更改为另一个字符串所需的最小替换次数来比较两个二进制数据字符串。如果字符串是彼此的副本，它将决定复制时发生的错误数量。在偏见检测背景下，TVD 量化了多少方面的结果一个必须改变才能与各方面的结果相匹配D.

总变体距离的公式如下：

$$TVD = ½*L_1P(P)_aP, P_d)$$

例如，假设您有三个类别的结果分布，$_i$y {$y_0y_1y_2$} = 在大学招生多类别场景中 {已接受、等候名单、拒绝}。你把平面计数之间的差异一个和D用于计算 TVD 的每个结果。结果如下：

$$L_1P(P)_aP, P_d) = |n_a^{0)}\text{-}n_d^{0)}|n_a^{(1)}\text{-}n_d^{(1)}|n_a^{2)}\text{-}n_d^{(2)}|$$

其中：

- $n_a^{i)}$是第 i 个类别结果的数量一个: 例如 $n_a^{0)}$是方面的数量一个接受。
- $n_d^{i)}$是方面 d 中第 i 个类别结果的数量：例如 $n_d^{2)}$是方面的数量D拒绝。

二进制、多类别和连续结果的 TVD 值范围为 [0, 1)，其中：

- 接近零的值表示标签的分布情况类似。
- 正值意味着标签分布差异，分歧越积极就越大。

科尔莫哥罗夫-斯米尔诺夫（KS）

Kolmogorov-Smiirnov 偏差指标 (KS) 等于小面分布中标签之间的最大差异一个和D的数据集。实施的双样本 KS 测试 SageMaker 澄清通过找到最不平衡的标签来补充标签不平衡的其他措施。

Kolmogorov-Smiirnov 指标的公式如下：

$$KS = \max(|P_a(y) - P_d(y)|)$$

例如，假设一组申请人（平面）一个) 大学被拒绝、候补名单或录取分别为 40%、40%、20%，而且其他申请人的这些费率（方面）D) 是 20%、10%、70%。然后 Kolmogorov-Smiirnov 偏差指标值如下：

KS = 最大值 (|0.4-0.2|, |0.4-0.1|, |0.2-0.7|) = 0.5

这告诉我们，平面分布之间的最大差异为 0.5，并且发生在接受率中。公式中有三个术语，因为标签是多类基数三。

二进制、多类别和连续结果的 LP 值范围为 [0, +1]，其中：

- 接近零的值表示标签在所有结果类别中的小面之间均匀分布。例如，申请贷款的两个方面都获得了 50% 的接受和 50% 的拒绝。
- 接近一个值表示一个结果的标签都在一个方面。例如，方面一个获得了 100% 的接受和方面D没有。
- 间歇性值表示最大标签不平衡的相对程度。

有条件的人口统计差异 (CDD)

人口差异指标 (DD) 确定了数据集中某个方面的拒绝结果的比例是否大于接受结果的比例。在构成数据集的两个方面（例如男性和女性）的二进制案例中，不受欢迎的一个被标记为平面D而且最喜欢的标签被标记为方面一个. 例如，在大学录取的情况下，如果女性申请人占被拒申请人的 46%，而且只占被接受申请人的 32%，我们说人口差异因为妇女被拒绝的比率超过了接受她们的比率. 女性申请人被标签为方面D在本例中，如果男性申请人占被拒申请人的 54% 和被接受申请人的 68%，那么这方面的人口统计差异就不存在，因为拒绝率低于接受率。男性申请者被标记为方面一个在本例中，

较不受青睐的方面的人口差异的公式D如下所示：

$$DD_d n = {}_d{}^{(0)}/n^{(0)} - n_d{}^{(1)}/n^{(1)}P = {}_d{}^{R}y^{0}P)_d{}^{-个}y^{1})$$

其中：

- $n^{(0)}n = {}_a{}^{(0)} + n_d{}^{(0)}$是受欢迎的方面的数据集中被拒绝的结果总数一个和处境不利的方面D.
- $n^{(1)}n = {}_a{}^{(1)} + n_d{}^{(1)}$是受欢迎的方面的数据集中可接受的结果总数一个和处境不利的方面D.
- $P_d{}^{R}y^{0}$）是被拒结果（值为 0）的比例D.
- $P_d{}^{-个}y^{1}$）是可接受的结果（值 1）在方面的比例D.

对于大学录取的例子，女性的人口差异是 DD_d= 0.46-0.32 = 0.14。对于男士 DD_a= 0.54-0.68 =-0.14。

为了排除辛普森的悖论，需要一个有条件的人口统计差异 (CDD) 指标，该指标将 DD 限制在数据集上定义子组层的属性。重组可以让人们深入了解较不受欢迎的方面造成明显人口差异的原因。典型的例子出现在伯克利入院的案例中，男子被接受的总体比率高于女性。该案例的统计数据用于 DD 的示例计算。但是，在对部门分组进行检查时，根据部门的条件，妇女的入学率显示比男子高。解释是，女性向接受率低于男性的部门提出申请。审查分组的接受率表明，对于接受率较低的部门，实际上接受女性的比率高于男子。

CDD 指标通过平均数据集属性定义的子组中发现的所有差异提供了一个单一的度量。它被定义为人口差异的加权平均值（$DD)_i$）对于每个子组，每个子组差异按包含中的观测值数成比例加权。有条件的人口差异的公式如下：

$$CDD = (1/n) * \Sigma_i n_i * DD_i$$

其中：

- $\Sigma_i n_i$= n 是观测值的总数，n_i是每个子组的观测值数。
- $DD_i n = {}_i{}^{(0)}/n^{(0)} - n_i{}^{(1)}/n^{(1)}P = {}_i{}^{R}y^{0}P)_i{}^{-个}y^{1})$ 是第 i 个子组的人口差异。

子组的人口差异（DD）$_i$）是被拒绝结果的比例与每个子组接受结果的比例之间的差异。

完整数据集 DD 的二进制结果的 DD 值范围$_d$或者对于其条件化的子组 DD$_i$是 [-1, +1]。

- +1：当方面没有拒绝的时候一个或子组而且不接受方面D或子组
- 正值表明存在着人口差异作为一个方面D或子组在数据集中被拒绝的结果的比例高于接受的结果。价值越高，平面越偏爱，差异就越大。
- 负值表示不存在人口差异的方面D或子组在数据集中所接受的结果的比例大于被拒绝结果的比例。价值越低，平面就越有利。
- -1：当方面没有拒绝的时候D或子组而且不接受方面一个或子组

如果你不以任何条件为条件，那么如果并且仅当 DPL 为零时，CDD 才为零。

这一指标有助于探索欧盟和英国不歧视法律和判例中的直接和间接歧视概念以及客观合理的概念。有关其他信息，请参阅为什么公平性不能自动化. 本 paper 还包含伯克利招生案例的相关数据和分析，这些数据和分析显示了对部门入学率亚组的条件规定如何说明辛普森的悖论。

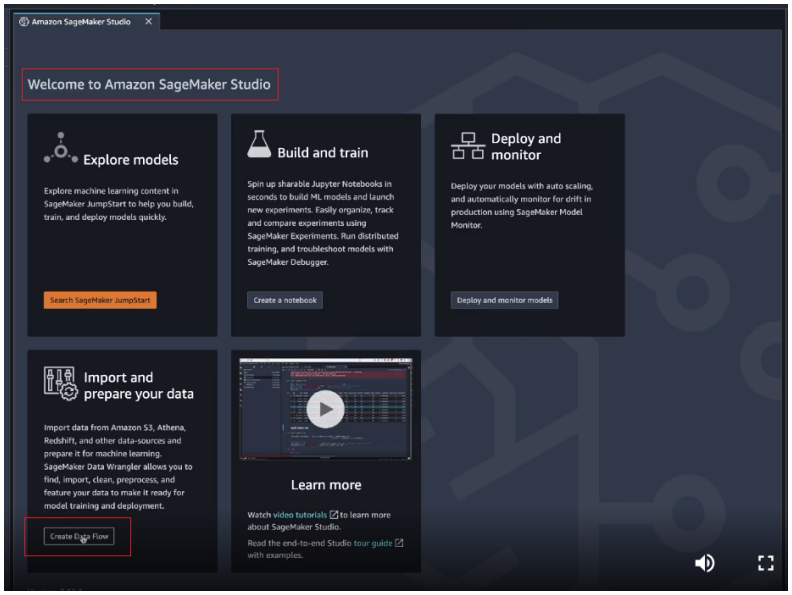
为训练前数据中的偏差生成报告 SageMaker 工作室

SageMaker 澄清与亚马逊集成 SageMaker Data Wrangler，它可以帮助您识别数据准备过程中的偏差，而无需编写自己的代码。数据 Wrangler 提供了 end-to-end 使用亚马逊导入、准备、转换、特色化和分析数据的解决方案 SageMaker Studio。有关 Data Wrangler 数据准备工作流的概述，请参阅使用亚马逊准备 ML 数据 SageMaker DATA Wrangler (p. 556). 您可以指定感兴趣的属性，例如性别或年龄，并 SageMaker 澄清运行一组算法来检测这些属性中是否存在偏见。算法运行后， SageMaker 澄清提供了一份直观报告，其中描述了可能偏见的来源和严重程度，以便您可以规划缓解措施。例如, 在一个财务数据集中, 与其他年龄组相比, 向一个年龄组提供商业贷款的例子很少, SageMaker 标记不平衡状态，以便您可以避免使用不利于该年龄组的模型。

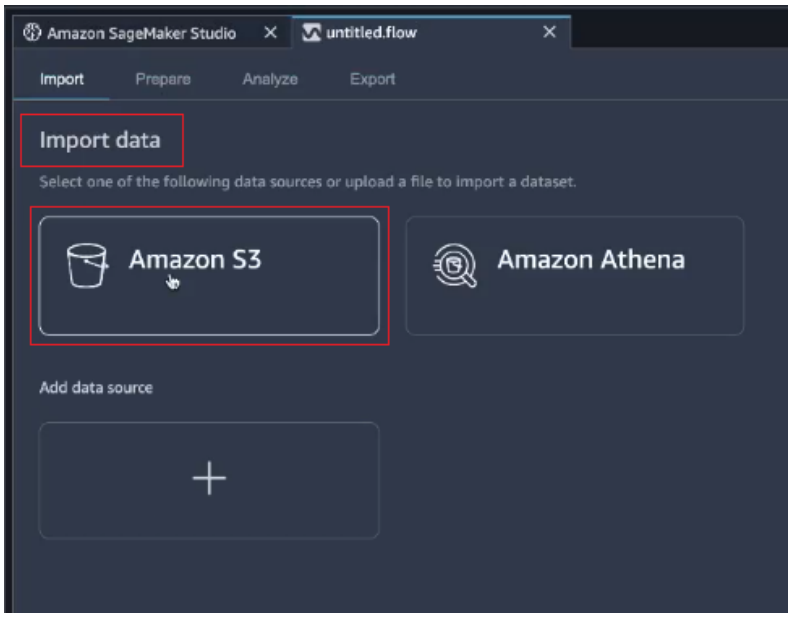
分析和报告数据偏差

要开始使用 Data Wrangler，请参阅首先使用数据 Wrangler (p. 557).

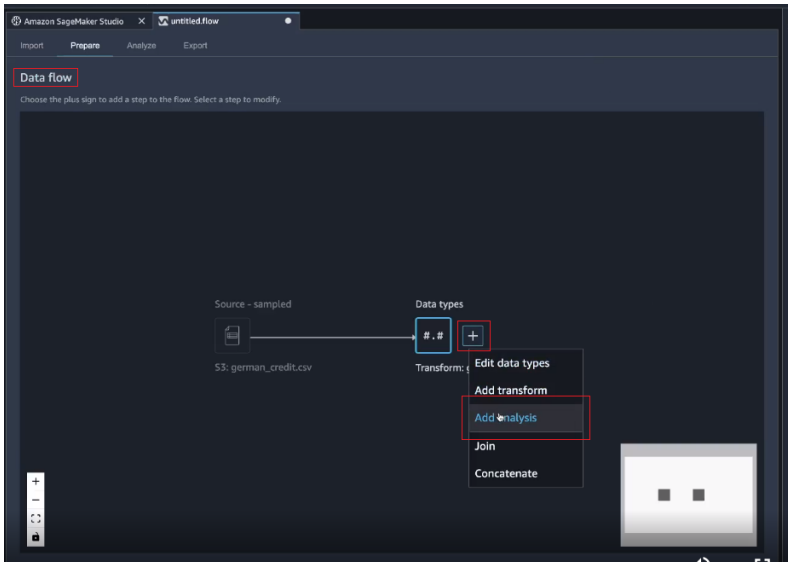
1. 打开亚马逊 SageMaker 选择录音室创建数据流来自 的导入和准备数据瓷砖。



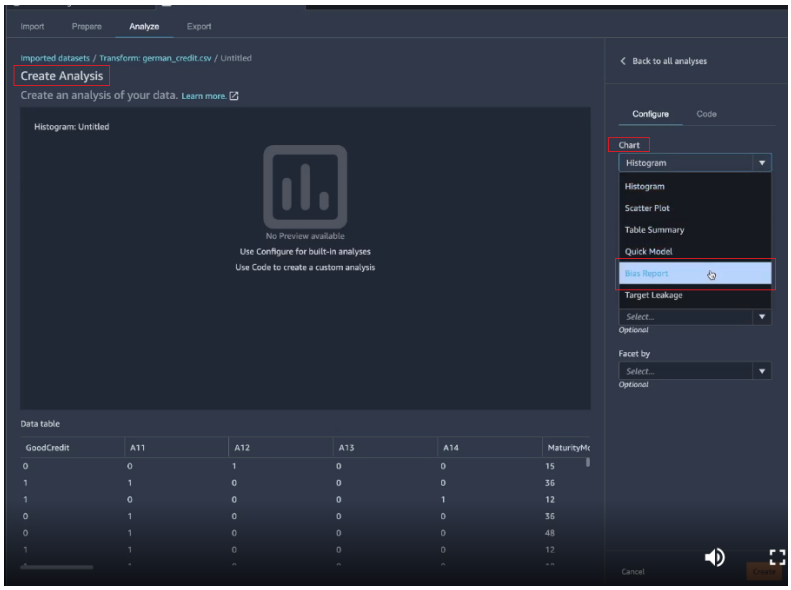
2. 从导入数据选项卡上，选择Amazon S3然后在数据源/S3 源页.



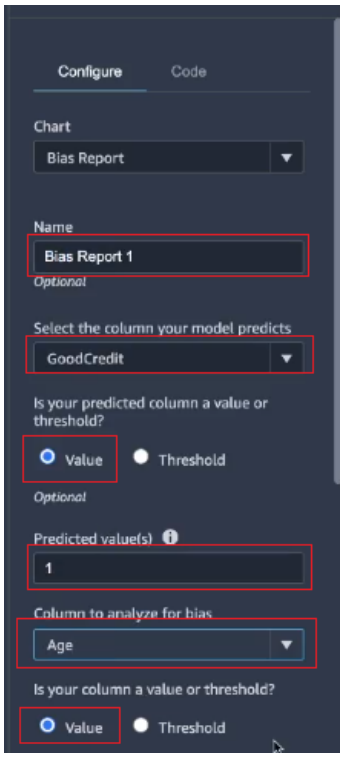
3. 导入数据后，请在数据流页面然后选择添加分析.



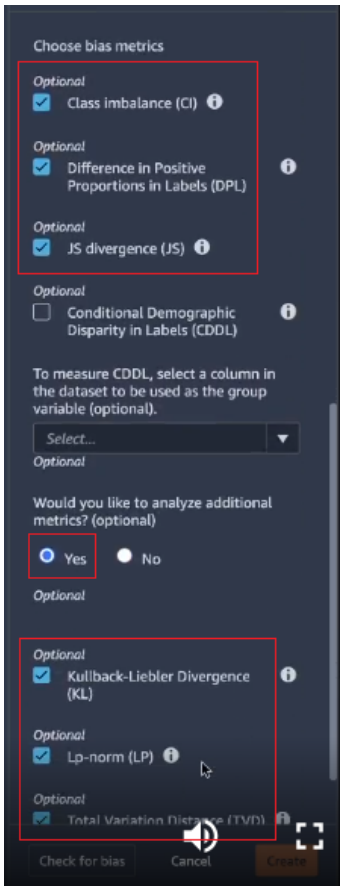
4. 在存储库的创建分析页面，转至配置面板然后选择偏置报告来自 的图表菜单。



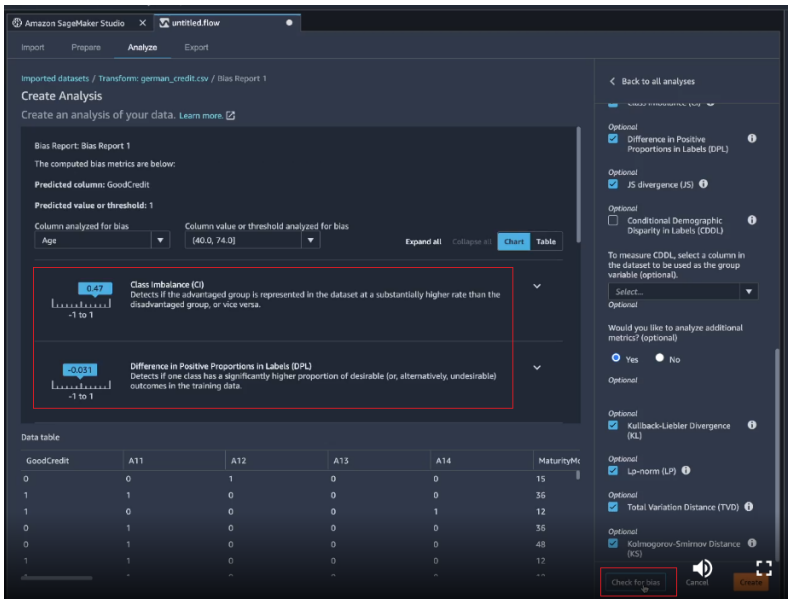
5. 通过提供名称、要预测的列以及它是值还是阈值、要分析偏差的列（小面）以及它是值还是阈值。



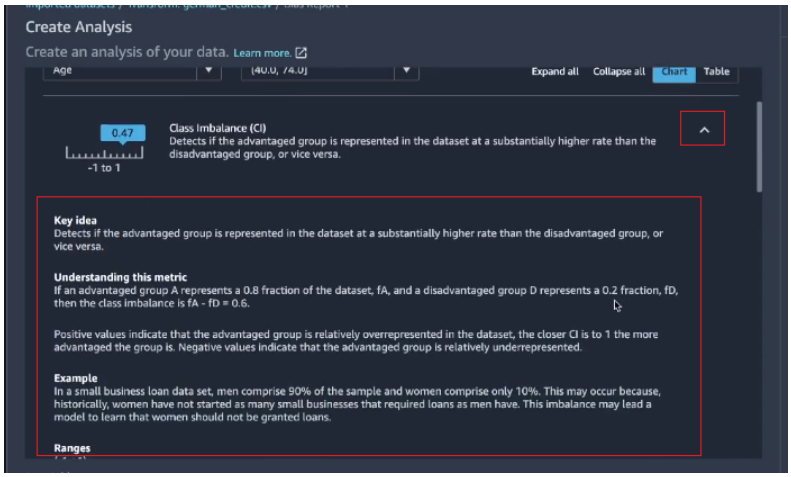
6. 通过选择偏差指标继续配置偏差报告。



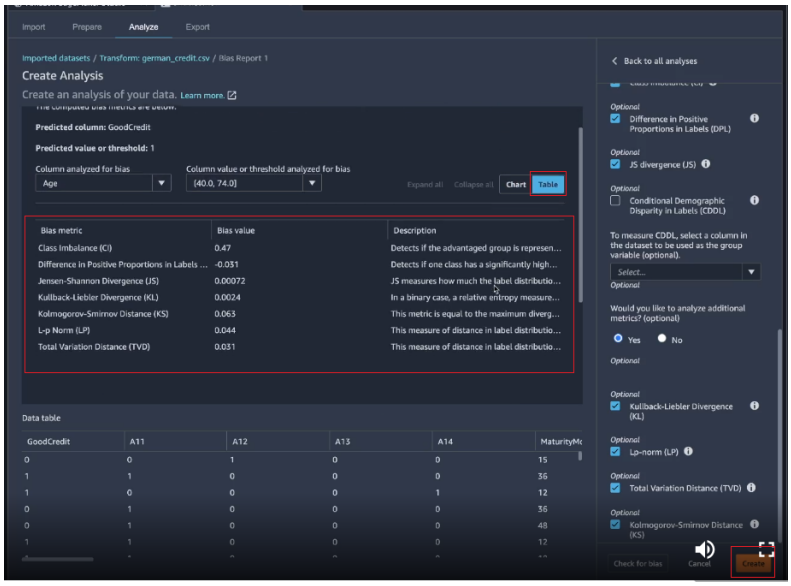
7. 选择检查偏差以生成和查看偏见报告。向下滚动以查看所有报告。



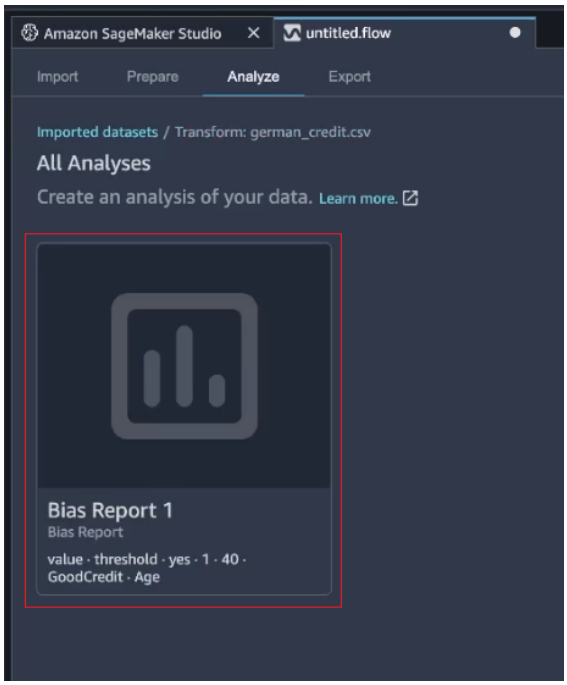
8. 选择偏差指标描述右侧的插入符号，以查看可以帮助您解释指标值重要性的文档。



9. 要查看偏差指标值的表格摘要，请选择表格，您可以通过选择以保存报告以供导出Create位于页面的右下角。



10. 在存储数据偏差报告的页面上，选择Export选项卡以下载报告。



使用亚马逊准备 ML 数据 SageMaker DATA Wrangler

亚马逊 SageMaker 数据 Wrangler（Data Wrangler）是亚马逊的一项功能 SageMaker 工作室提供 end-to-end 用于导入、准备、转换、特征化和分析数据的解决方案。您可以将 Data Wrangler 数据流集成到机器学习 (ML) 工作流程中，以简化和简化数据预处理和功能工程，只需很少或不使用编码即可。您还可添加自己的 Python 脚本和转换以自定义工作流。

Data Wrangler 提供以下核心功能来帮助您分析和准备机器学习应用程序的数据。

- 导入— Connect 到 Amazon Simple Storage Service (Amazon S3) 和从中导入数据，Amazon Athena（Athena）和 Amazon Redshift。
- 数据流— 创建数据流以定义一系列机器学习数据准备步骤。您可以使用流程合并来自不同数据源的数据集，确定要应用于数据集的转换数量和类型，并定义可以集成到 ML 管道中的数据准备工作流程。
- 转换— 使用标准清理和转换数据集变换例如字符串、矢量和数字数据格式化工具。使用文本和日期/时间嵌入以及分类编码等转换来为数据添加特色。
- 分析— 在流程中任意点分析数据集中的要素。Data Wrangler 包括内置的数据可视化工具，例如散点图和直方图，以及目标泄漏分析和快速建模等数据分析工具，以了解特征相关性。
- Export— Data Wrangler 提供向其他出口选项 SageMaker 服务，包括数据 Wrangler 职位、亚马逊 SageMaker 功能商店和管道，使您能够将数据准备流程集成到机器学习工作流程中。您还可以将数据 Wrangler 流程导出为 Python 代码。

要开始使用 Data Wrangler，请参阅首先使用数据 Wrangler (p. 557).

主题

- 首先使用数据 Wrangler (p. 557)
- 导入 (p. 566)

- 创建和使用数据 Wrangler 流 (p. 601)
- 获取有关数据和数据质量的见解 (p. 611)
- 转换数据 (p. 623)
- 可视化 (p. 658)
- Export (p. 664)
- 安全和权限 (p. 676)
- 发行说明 (p. 684)
- 问题排查 (p. 685)
- 更新数据 Wrangler (p. 560)
- 关闭数据 Wrangler (p. 690)

首先使用数据 Wrangler

亚马逊 SageMaker 数据 Wrangler 是亚马逊的一项功能 SageMaker Studio。使用此部分可以了解如何访问和开始使用 Data Wrangler。执行以下操作：

1. 完成中的每一步先决条件 (p. 557).
2. 按照中过程操作访问数据 Wrangler (p. 557)开始使用数据 Wrangler。

先决条件

要使用 Data Wrangler，您必须满足以下先决条件。

1. 要使用 Data Wrangler，你需要访问m5.4xlargeAmazon Elastic Compute Cloud (Amazon EC2) 实例。要了解如何查看配额以及如有必要，请求提高配额，请参阅Amazon服务配额.
2. 配置中所述的所需权限安全和权限 (p. 676).

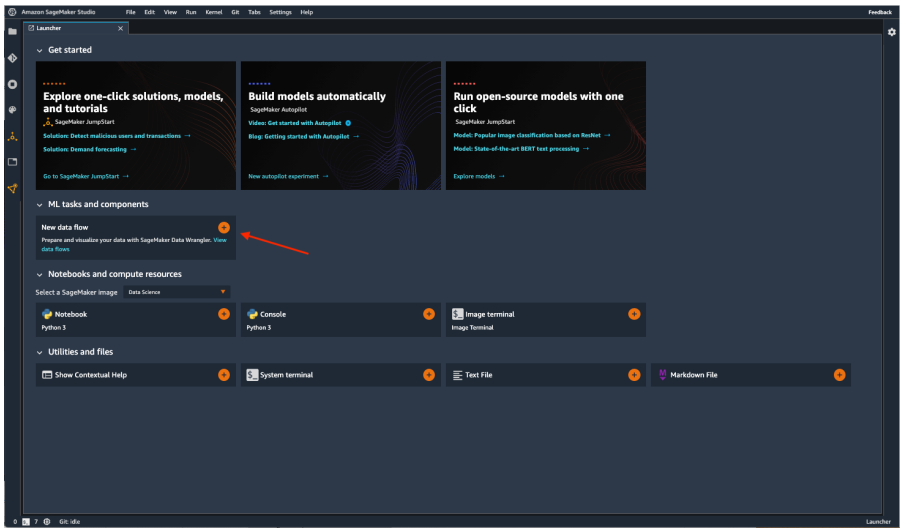
要使用 Data Wrangler，您需要一个活动的 Studio 实例。要了解如何启动新实例，请参阅注册到 Amazon SageMaker 域 (p. 32). 当你的 Studio 实例是什么时候Ready中的说明，使用中的说明访问数据 Wrangler (p. 557).

访问数据 Wrangler

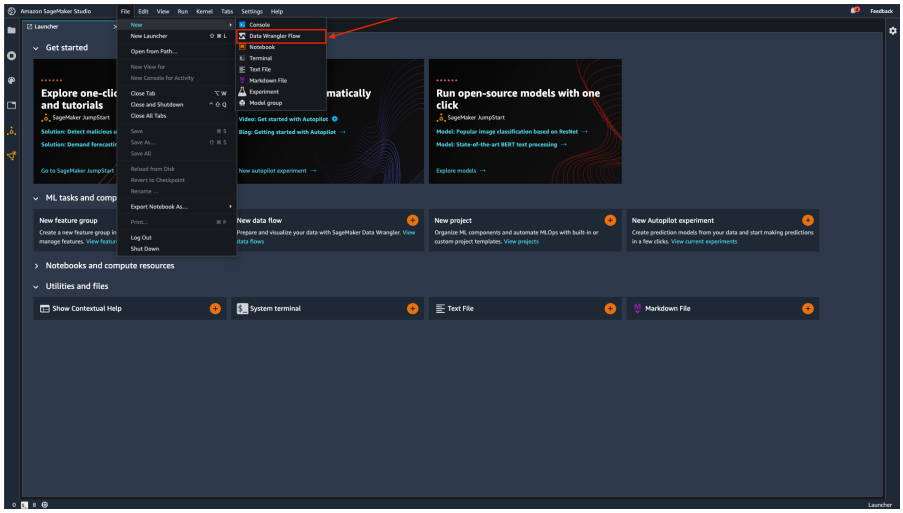
以下过程假定您已完成先决条件 (p. 557).

要在 Studio 中访问数据牧马人：

1. 在要用于启动 Studio 的用户旁边，选择打开 Studio.
2. 当 Studio 打开时，选择+登录新的数据流下卡ML 任务和组件. 这将在 Studio 中创建一个包含 .flow 文件的新目录，其中包含数据流。.flow 文件将在 Studio 中自动打开。



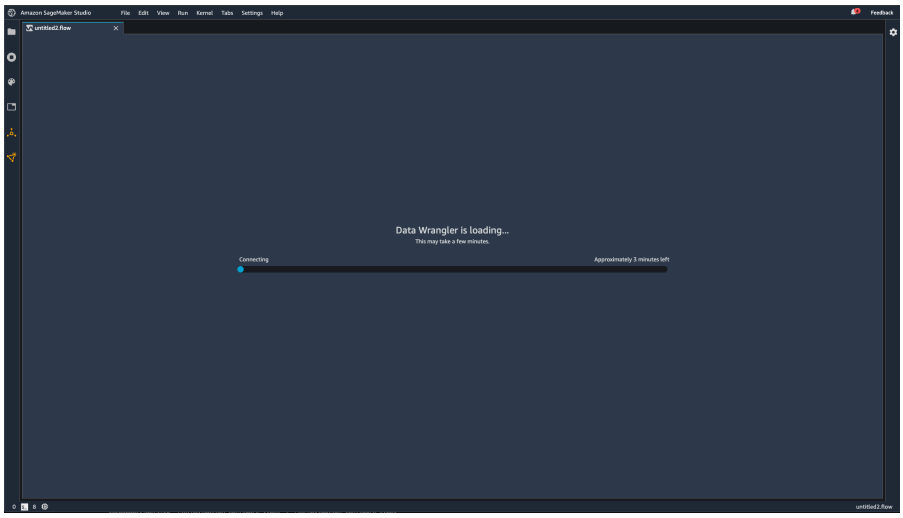
您还可以通过选择以下方式创建新流File (文件)，那么New，然后选择数据 Wrangler Flow在顶部导航栏中。



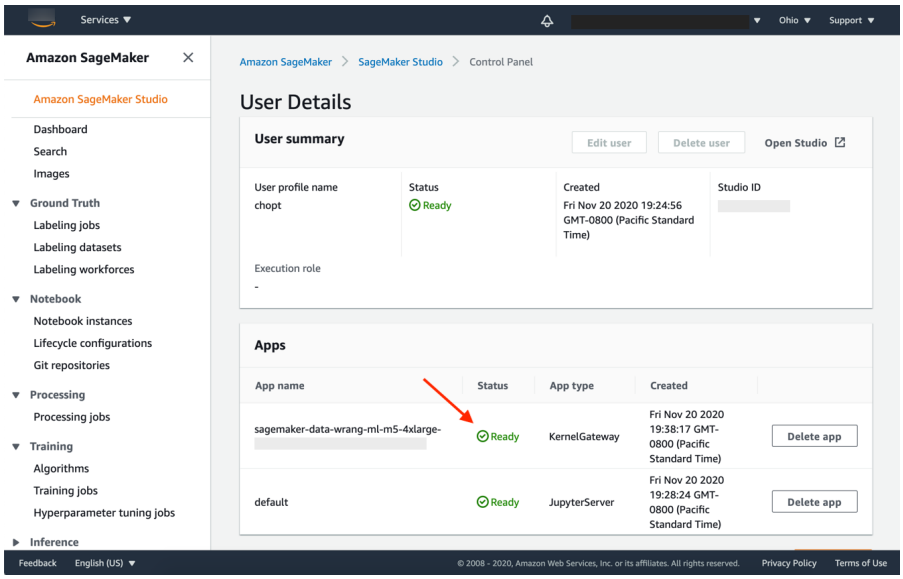
3. （可选）重命名新目录和 .flow 文件。
4. 在 Studio 中创建新的 .flow 文件时，您可能会在DATA Wrangler上面显示以下内容：

   数据 Wrangler 正在加载...

   该过程可能需要几分钟.



只要KernelGateway应用程序Details页面是Pending. 要查看此应用程序的状态，请在亚马逊 SageMaker 工作室页面上，选择您用于访问 Studio 的用户的名称。在存储库的Details页面，你看到KernelGateway应用程序应用程序. 等到此应用程序状态为Ready开始使用数据 Wrangler。首次启动 Data Wrangler 可能需要 5 分钟左右。



5. 要开始使用，请选择一个数据源并使用它导入数据集。请参阅导入 (p. 566)了解更多信息。

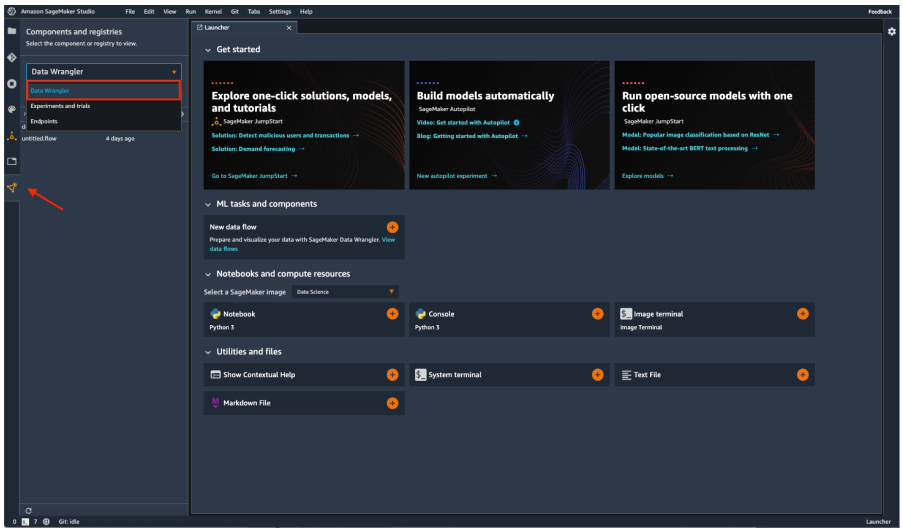
   导入数据集时，它将显示在数据流中。要了解更多信息，请参阅 创建和使用数据 Wrangler 流 (p. 601)。

6. 导入数据集后，Data Wrangler 会自动推断每列中的数据类型。选择+在旁边数据类型步骤然后选择编辑数据类型.

   Important

   将转换添加到数据类型步骤，你不能使用批量更新列类型更新类型.

7. 使用数据流添加变换和分析。要了解详情，请参阅转换数据 (p. 623)和可视化 (p. 658).
8. 要导出完整的数据流，请选择Export然后选择导出选项。要了解更多信息，请参阅 Export (p. 664)。
9. 最后，选择组件和注册表图标，然后选择DATA Wrangler从下拉列表中查看您创建的所有 .flow 文件。您可以使用此菜单查找数据流并在数据流之间移动。



启动 Data Wrangler 后，您可以使用以下部分介绍如何使用 Data Wrangler 创建机器学习数据准备流程。

更新数据 Wrangler

我们建议您定期更新 Data Wrangler Studio 应用程序以访问最新功能和更新。Data Wrangler 应用程序名称以sagemaker-数据响起. 要了解如何更新 Studio 应用，请参阅关闭并更新 Studio 应用 (p. 130).

演示：Data Wrangler 泰坦尼克号数据集演练

以下部分提供演练，可帮助您开始使用 Data Wrangler。本演练假定您已执行以下操作：访问数据 Wrangler (p. 557)并打开一个新的数据流文件，你打算用于演示。你可能想要将此 .flow 文件重命名为类似的内容**`titanic-demo.flow`**.

此演练使用Titanic 数据集. 该数据集包含首航的乘客的存活状况、年龄、性别和级别（作为经济地位的代表）RMS Titanic1912 年。

在本教程中，您将执行以下步骤。

- 上传Titanic 数据集转到 Amazon Simple Storage Service (Amazon S3)，然后将此数据集导入数据 Wrangler。
- 使用 Data Wrangler 分析来分析此数据集。
- 使用 Data Wrangler 数据转换来定义数据流。
- 将流程导出到可用于创建 Data Wrangler 作业的 Jupyter 笔记本。
- 处理你的数据，然后启动 SageMaker 培训工作来训练 xgBoost 二进制分类器。

将数据集上传到 S3 然后导入

首先，请下载Titanic 数据集并将其上传到中的 Amazon S3 (Amazon S3) 存储桶Amazon您要在其中完成此演示的区域。

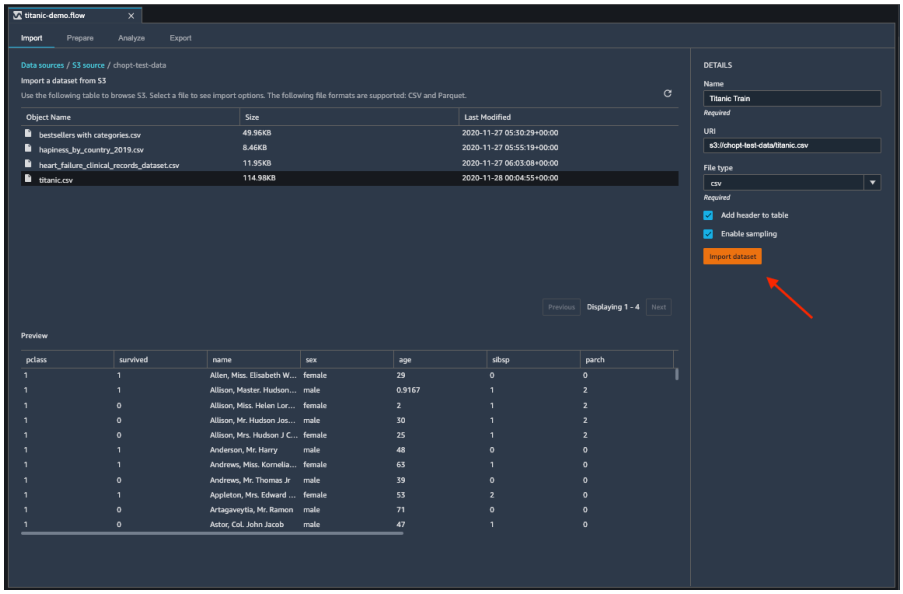
如果您是 Amazon S3 的新用户，则可以在 Amazon S3 控制台中使用拖放操作来执行此操作。要了解如何操作，请参阅使用拖放功能上传文件和文件夹在 Amazon Simple Storage Service 用户指南中。

> Important
>
> 将您的数据集上传到同一个 S3 存储桶Amazon您要使用的区域来完成此演示。

当您的数据集成功上传到 Amazon S3 后，您可以将其导入 Data Wrangler 中。

将泰坦尼克号数据集导入 Data Wrangler

1. 选择导入数据 Wrangler 流程文件中的选项卡。
2. SelectAmazon S3.
3. 使用从 S3 导入数据集表来查找添加泰坦尼克号数据集的存储桶。选择泰坦尼克号数据集 CSV 文件以打开详细信息窗格。
4. 在详细信息，文件类型应该是 CSV。选择将标头添加到表以指定数据集的第一行为标头。您还可以将数据集命名为更友好的名称，例如**`Titanic-train`**.
5. Select导入数据集.



将数据集导入 Data Wrangler 后，它将显示在数据流中。您可以随时通过选择数据流选项卡。您可以双击某个节点进入节点详细信息视图，该视图允许您添加转换或分析；否则，您可以使用加号图标进行快速导航。在下一部分中，您将使用此数据流添加分析和转换步骤。

数据流

在数据流部分中，数据流中唯一的步骤是最近导入的数据集和数据类型步骤。应用转换后，您可以返回此选项卡查看数据流的外观。现在，在准备和分析选项卡。

准备和可视化

Data Wrangler 具有内置的转换和可视化效果，您可以用它们来分析、清理和转换数据。

这些区域有：数据节点详细信息视图的选项卡列出了右侧面板中的所有内置转换，其中还包含一个可以在其中添加自定义转换的区域。以下用例展示了如何使用这些转换。

数据探究

首先，使用分析创建数据表摘要。执行以下操作：

1. 选择+在旁边数据类型步入数据流并选择添加分析.
2. 在分析区域，选择表摘要从下拉列表中选择。
3. 给表格摘要 a名称.
4. Select预览以预览将要创建的表格。

5. 选择Create将其保存到数据流中。它出现在所有分析.

使用您看到的统计数据，您可以对此数据集进行类似以下内容的观测：

- 平均票价（平均值）约为 33 美元，而最高价格超过 500 美元。此列可能有异常值。
- 此数据集使用?以指示缺少的值。许多列缺少值：舱、着手, 和home.dest
- 年龄类别缺少 250 多个值。

选择准备返回到数据流。接下来，使用从这些统计数据中获得的见解清理数据。

删除未使用的列

使用上一节中的分析，清理数据集以便为训练做好准备。要向数据流添加新转换，请选择+在旁边数据类型步入你的数据流然后选择添加转换.

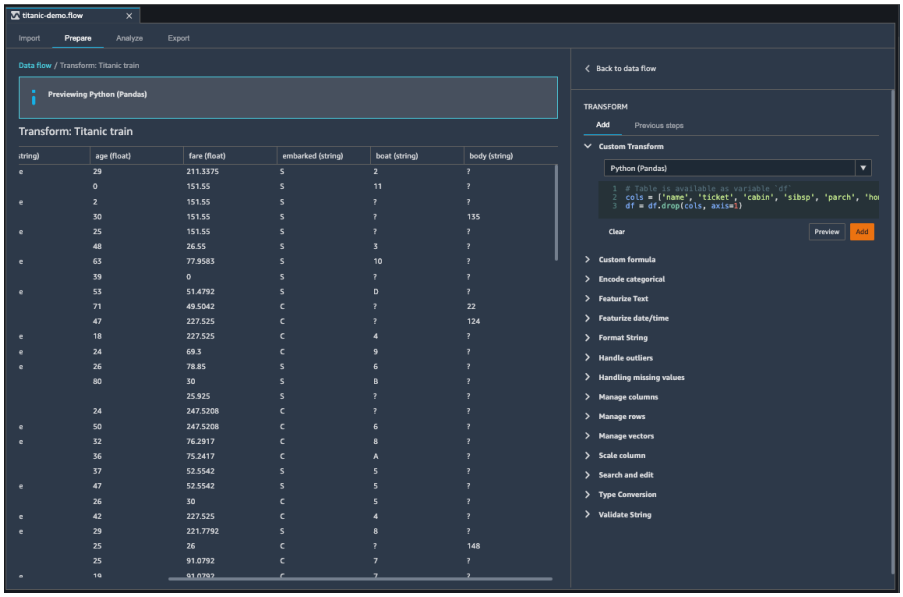
首先，删除您不想用于训练的列。您可以使用熊猫数据分析库来做到这一点，或者你可以使用其中一个内置的变换。

要使用 Pandas 执行此操作，请按照以下步骤操作。

1. 在自定义转换部分，选择蟒蛇（熊猫）从下拉列表中选择。
2. 在代码框中输入以下内容。

```
cols = ['name', 'ticket', 'cabin', 'sibsp', 'parch', 'home.dest','boat', 'body']
df = df.drop(cols, axis=1)
```

3. 选择预览以预览更改，然后选择Add来添加转换。



要使用内置转换，请执行以下操作：

1. 选择管理列从右侧面板中找到。
2. 适用于输入栏，选择舱，然后选择预览.
3. 验证舱列已被删除，然后选择Add.
4. 对以下列重复这些步骤：票、名称、sibsp、帕奇、home.dest、船, 和身体.

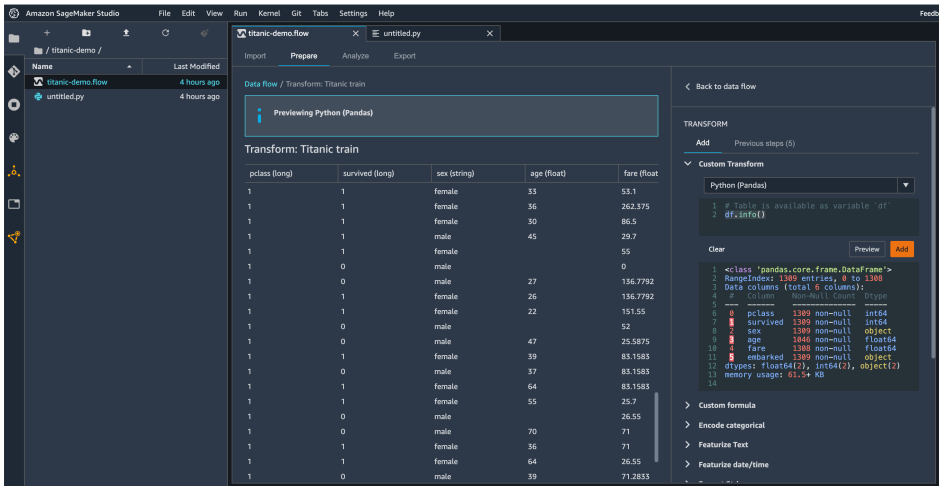
清理缺失值

现在，清理缺失的值。您可以使用处理缺失值转换组。

许多列缺少值。在剩余的列中，年龄和票价包含缺少的值。使用自定义转换.

使用蟒蛇（熊猫）选项中，使用以下命令快速查看每列中的条目数：

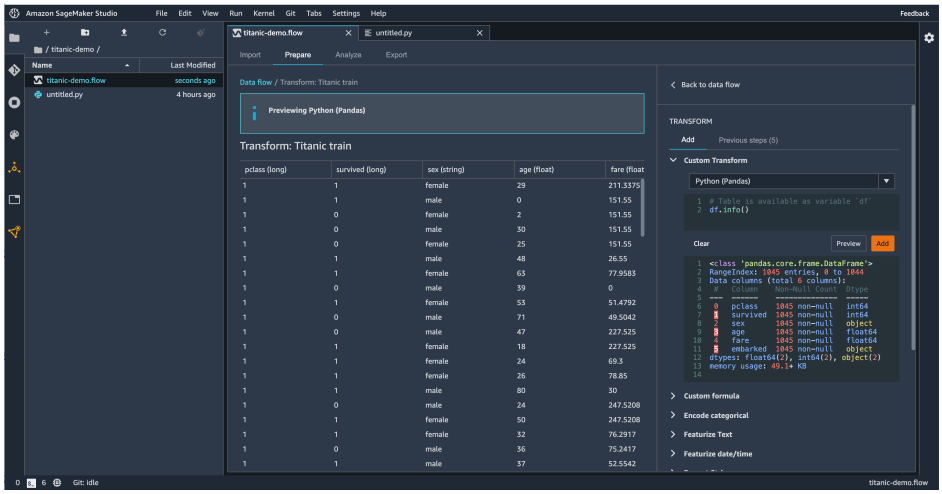
```
df.info()
```



在中删除缺少值的行年龄类别，请执行以下操作：

1. 选择处理缺失值.
2. 选择丢失掉(对于)变压器.
3. 选择删除行(对于)维度.
4. 选择年龄(对于)输入栏.
5. 选择预览要查看新的数据框，然后选择Add将转换添加到流程中。
6. 重复相同的过程票价.

您可以使用`df.info()`中的自定义转换部分，以确认所有行现在都有 1,045 个值。



自定义熊猫：编码

尝试使用 Pandas 进行平面编码。对分类数据进行编码是为类别创建数字表示形式的过程。例如，如果你的类别`Dog`和`Cat`，你可以将这些信息编码为两个向量：`[1,0]`要表示`Dog`, 和`[0,1]`要表示`Cat`.

1. 在自定义转换部分，选择蟒蛇（熊猫）从下拉列表中选择。
2. 在代码框中输入以下内容。

```
import pandas as pd

dummies = []
cols = ['pclass','sex','embarked']
for col in cols:
    dummies.append(pd.get_dummies(df[col]))

encoded = pd.concat(dummies, axis=1)

df = pd.concat((df, encoded),axis=1)
```

3. 选择预览以预览更改。每列的编码版本将添加到数据集中。
4. 选择Add来添加转换。

自定义 SQL：SELECT

现在，选择要使用 SQL 保留的列。对于此演示，请选择以下列出的列`SELECT`网页。由于幸存下来是你的训练目标专栏，把那个专栏放在第一位。

1. 在自定义转换部分，选择SQL（PySpark SQL）从下拉列表中选择。
2. 在代码框中输入以下内容。

```
SELECT survived, age, fare, 1, 2, 3, female, male, C, Q, S FROM df;
```

3. 选择预览以预览更改。你的`SELECT`语句是唯一剩下的列。
4. 选择Add来添加转换。

Export

完成数据流创建后，您有许多导出选项。以下部分介绍如何导出到 Data Wrangler 作业笔记本。Data Wrangler 作业用于使用数据流中定义的步骤处理数据。要了解所有导出选项的更多信息，请参阅Export (p. 664).

导出到数据 Wrangler Job 笔记本

当您使用数据 Wrangler 任务，该过程会自动创建一个 Jupyter 笔记本。此笔记本会在您的 Studio 实例中自动打开，并配置为运行 SageMaker 处理作业来运行 Data Wrangler 数据流，这被称为 Data Wrangler 作业。

1. 保存数据流。SelectFile (文件)然后选择保存数据 Wrangler 流.
2. 选择Export选项卡。
3. 选择数据流中的最后一步。

4. 选择数据 Wrangler Job. 这将打开 Jupyter 笔记本。
5. 选择任何Python 3（数据科学）的内核内核.
6. 当内核启动时，将笔记本电脑中的单元格运行到开始 SageMaker 培训 Job（可选）.
7. 或者，您可以在中运行单元启动 SageMaker 培训 Job（可选）如果你想创建 SageMaker 培训作业来训练 xgBoost 分类器。您可以找到运行成本 SageMaker 在训练作业亚马逊 SageMaker 定价.

   或者，您可以添加中找到的代码块训练 XGBoost 分类器 (p. 565)转到笔记本电脑然后运行它们以使用XGBoost用于训练 xgBoost 分类器的开源库。
8. 取消注释并运行下面的单元格清除然后运行它来还原 SageMaker Python SDK 至其原始版本。

你可以在 SageMaker 中的控制台处理选项卡。此外，您可以使用 Amazon CloudWatch 监控数据 Wrangler 作业。有关其他信息，请参阅监控 Amazon SageMaker 使用处理任务 CloudWatch 日志和指标.

如果你开始了培训工作，你可以使用 SageMaker 控制台下训练任务中的训练部分.

训练 XGBoost 分类器

在启动 Data Wrangler 工作的同一笔笔记本中，您可以在最少的数据准备的情况下使用准备好的数据来提取数据并训练 XGBoost Binary Classifier Lassifier。

1. 首先，使用升级必要的模块pip然后删除 _SUCCESS 文件（最后一个文件在使用时有问题awswrangler)。

```
! pip install --upgrade awscli awswrangler boto sklearn
! aws s3 rm {output_path} --recursive  --exclude "*" --include "*_SUCCESS*"
```

2. 从 Amazon S3 读取数据。您可以使用awswrangler以递归方式读取 S3 前缀中的所有 CSV 文件。然后将数据拆分为要素和标注。标签是数据框的第一列。

```
import awswrangler as wr

df = wr.s3.read_csv(path=output_path, dataset=True)
X, y = df.iloc[:,:-1],df.iloc[:,-1]
```

   - 最后，创建 dMatrices（数据的 xgBoost 原始结构）并使用 xgBoost 二进制分类进行交叉验证。

```
import xgboost as xgb

dmatrix = xgb.DMatrix(data=X, label=y)

params = {"objective":"binary:logistic",'learning_rate': 0.1, 'max_depth': 5,
 'alpha': 10}

xgb.cv(
    dtrain=dmatrix,
    params=params,
    nfold=3,
    num_boost_round=50,
    early_stopping_rounds=10,
    metrics="rmse",
    as_pandas=True,
    seed=123)
```

关闭数据 Wrangler

使用完 Data Wrangler 后，我们建议您关闭它运行的实例，以避免产生额外费用。要了解如何关闭 Data Wrangler 应用程序和关联的实例，请参阅关闭数据 Wrangler (p. 690).

导入

你可以使用亚马逊 SageMaker Data Wrangler 从以下导入数据数据源：Amazon Simple Storage Service (Amazon S3)、Amazon Athena、Amazon Redshift 和 Snowflake。您导入的数据集最多可以包含 1000 列。

主题

- 从 Amazon S3 导入数据 (p. 566)
- 从 Athena 导入数据 (p. 570)
- 从 Amazon Redshift 导入数据 (p. 570)
- 从 Databricks (JDBC) 导入数据 (p. 574)
- 从 Snowflake 导入数据 (p. 576)
- 导入的数据存储 (p. 600)

有些数据源允许您添加多个数据连接：

- 您可以连接到多个 Amazon Redshift 集群。每个集群都会变成数据源。
- 您可以查询账户中的任何 Athena 数据库以从该数据库导入数据。

从数据源导入数据集时，该数据集将显示在数据流中。Data Wrangler 会自动推断您的数据集中每一列的数据类型。要修改这些类型，请选择数据类型步骤然后选择编辑数据类型.

当您从 Athena 或 Amazon Redshift 导入数据时，导入的数据将自动存储在默认值中 SageMaker 用于 Amazon您正在使用 Studio 的区域。此外，Athena 将您在 Data Wrangler 中预览的数据存储在此存储桶中。要了解更多信息，请参阅 导入的数据存储 (p. 600)。

> Important
>
> 默认 Amazon S3 存储桶可能没有最不允许的安全设置，例如存储桶策略和服务器端加密 (SSE)。强烈建议您添加存储桶策略以限制对导入到 Data Wrangler 的数据集的访问.
>
> Important
>
> 此外，如果您为 SageMaker 使用托管策略，我们强烈建议您将其范围缩小到允许您执行使用案例的限制性最强的策略。有关更多信息，请参阅 授予 IAM 角色使用 Data Wrangler 的权限 (p. 677)。

从 Amazon S3 导入数据

您可以使用 Amazon Simple Storage Service (Amazon S3) 随时在 Web 上的任何位置存储和检索的任意大小的数据。您可以使用Amazon Web Services Management Console，这是一个简单直观的 Web 界面，以及 Amazon S3 API。如果您已在本地存储数据集，我们建议您将其添加到 S3 存储桶以导入到 Data Wrangler 中。要了解如何操作，请参阅将对象上传到存储桶（Amazon Simple Storage Service 用户指南）。

数据 Wangler 使用S3 Select允许您在 Data Wrangler 中预览 Amazon S3 文件。每次文件预览都需要支付标准费用。要了解有关定价的更多信息，请参阅请求和数据检索选项卡Amazon S3 定价.

> Important
>
> 如果您计划导出数据流并启动 Data Wrangler 作业，请将数据提取到 SageMaker feature store 或创建 SageMaker 管道，请注意，这些集成要求 Amazon S3 输入数据位于同一个Amazon区域。
>
> Important
>
> 如果要导入 CSV 文件，请确保它满足以下要求：
>
> - 数据集中的记录不能超过一行。
> - 一个反斜杠，\，是唯一有效的转义字符。

- 您的数据集必须使用以下分隔符之一：
  - 逗号 —,
  - 冒号 —:
  - 分号 —;
  - Pipe —|
  - Tab-[TAB]

为了节省空间，您可以导入压缩的 CSV 文件。

Data Wrangler 使您能够导入整个数据集或对其中的一部分进行采样。它提供了以下抽样选项：

- 无 — 导入整个数据集。
- 第一个 K — 对数据集的前 K 行进行采样，其中 K 是您指定的整数。
- 随机化 — 获取您指定大小的随机样本。

导入数据后，还可以使用采样变换器从整个数据集中获取一个或多个样本。有关采样变压器的更多信息，请参阅采样 (p. 650).

您可以将单个文件或多个文件作为数据集导入。当数据集被分区为单独的文件时，可以使用多文件导入操作。它从 Amazon S3 目录中获取所有文件，并将其作为单个数据集导入。有关可导入的文件类型以及如何导入这些文件的信息，请参阅以下内容。

Single File Import

可以导入以下格式的单个文件：

- 逗号分隔值 (CSV)
- Parquet
- Javascript 对象表示法 (JSON)
- ORC 优化的行列式 (ORC)

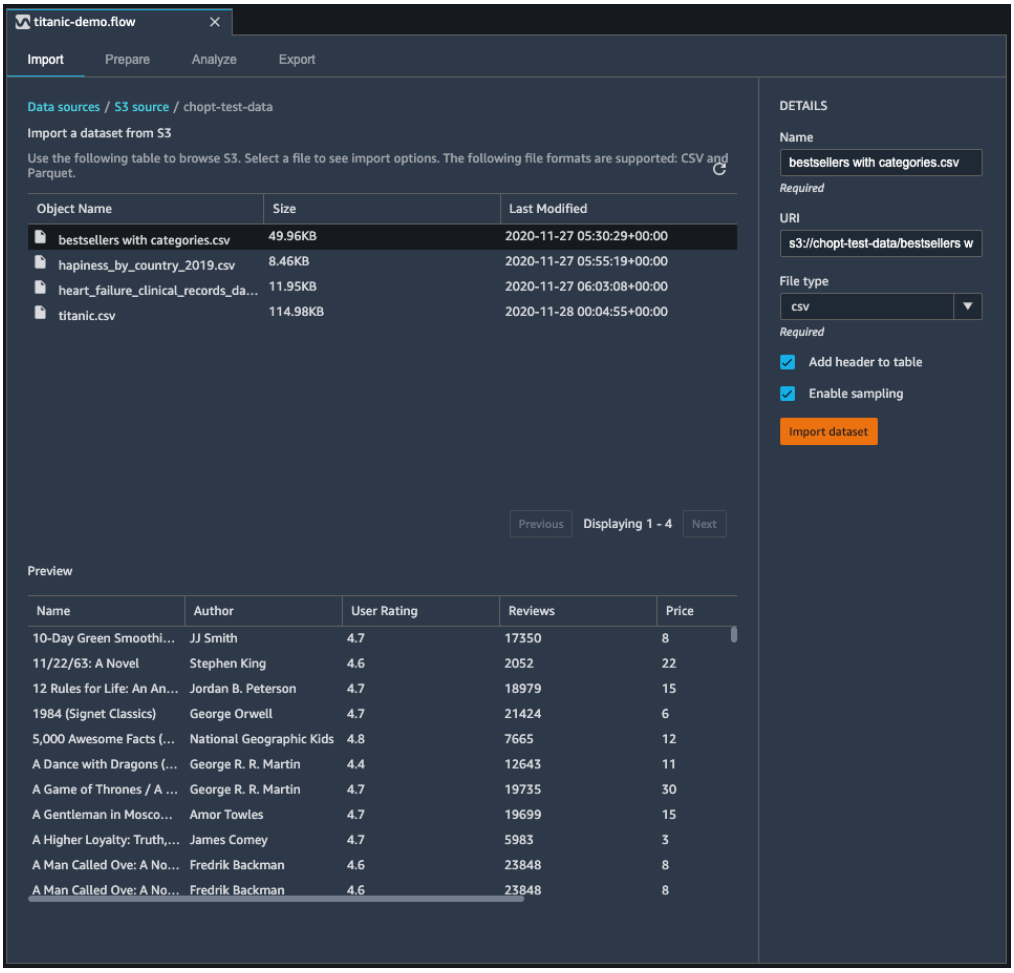
对于以 JSON 格式化的文件，Data Wrangler 同时支持 JSON 行 (.jsonl) 和 JSON 文档 (.json)。当您预览数据时，它会自动以表格格式显示 JSON。对于大于 5 MB 的嵌套 JSON 文档，Data Wrangler 将结构架和数组的架构显示为数据集中的值。使用展平结构和爆炸数组运算符以表格格式显示嵌套值。有关更多信息，请参阅 取消嵌套 JSON 数据 (p. 656) 和 分解数组 (p. 657)。

选择数据集时，可以对其进行重命名、指定文件类型并将第一行标识为标题。

您可以在单个导入步骤中导入已分区为 Amazon S3 存储桶中的多个文件的数据集。

要从存储在 Amazon S3 中的单个文件将数据集导入 Data Wrangler，请执行以下操作：

1. 如果你目前没有使用导入选项卡上，选择导入.
2. UNDER数据准备，选择Amazon S3要查看导入 S3 数据源查看。
3. 从可用 S3 存储桶表中，选择存储桶并导航到要导入的数据集。
4. 选择要导入的文件。如果您的数据集没有 .csv 或 .parquet 扩展名，请从文件类型下拉列表。
5. 如果您的 CSV 文件具有标题，请选中旁边的复选框。将标头添加到表.
6. 使用预览表来预览数据集。此表最多显示 100 行。
7. 在详细信息窗格中，验证或更改名称和文件类型对于您的数据集。如果你添加名称如果包含空格，则导入数据集时，这些空格将被下划线替换为下划线。
8. 指定要使用的采样配置。
9. 选择导入数据集.



Multifile Import

以下是导入多个文件的要求：

- 这些文件必须位于 Amazon S3 存储桶的同一文件夹中。
- 这些文件必须共享相同的标头或者没有标题。

每个文件必须采用以下格式之一：

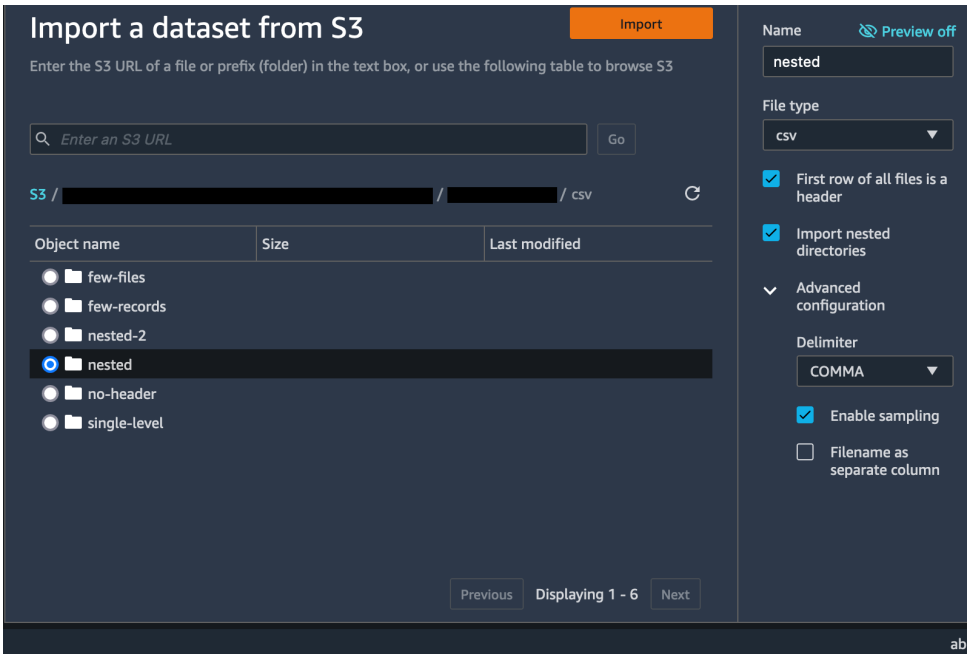
- CSV
- Parquet
- ORC 优化的行列式 (ORC)
- JSON

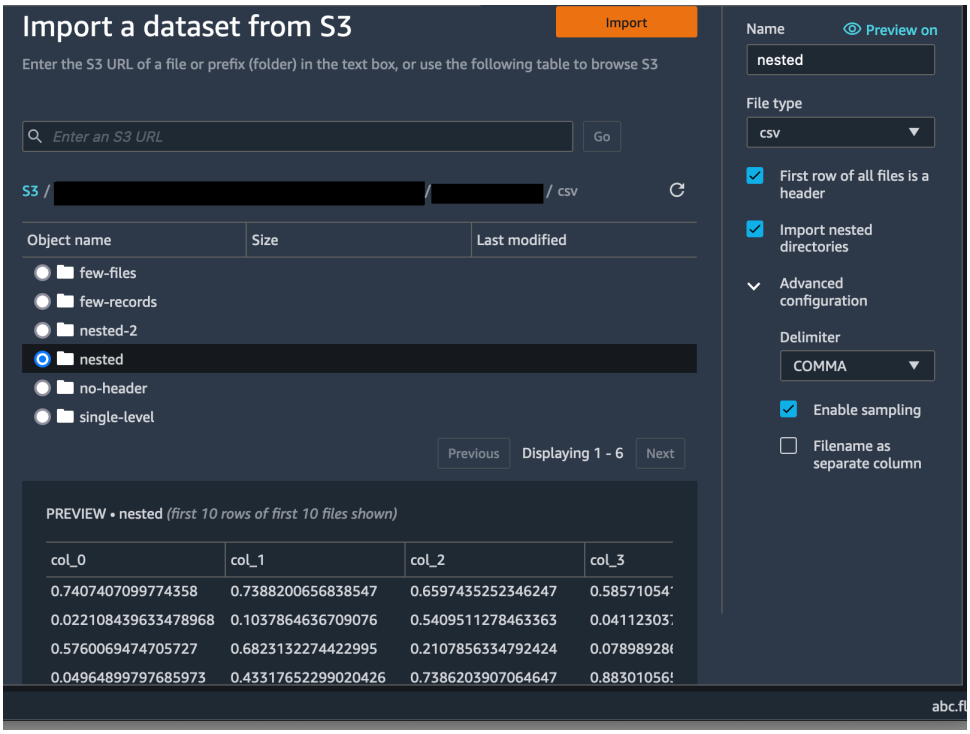
可以使用以下过程导入多个文件。

从存储在 Amazon S3 目录中的多个文件导入数据集到 Data Wrangler

1. 如果你目前没有使用导入选项卡上，选择导入.
2. UNDER数据准备，选择Amazon S3要查看导入 S3 数据源查看。
3. 从可用 S3 存储桶表中，选择包含要导入的文件夹的存储桶。

4. 选择包含要导入的文件的文件夹。每个文件必须采用支持的格式之一。您的文件必须是相同的数据类型。
5. 如果您的文件夹包含带标题的 CSV 文件，请选中旁边的复选框。第一行是标题.
6. 如果您的文件嵌套在其他文件夹中，请选中旁边的复选框。包括嵌套目录.
7. (可选) 选择添加文件名列向数据集添加一列，该列显示每个观测值的文件名。
8. （可选）默认情况下，Data Wrangler 不会向您显示文件夹的预览。你可以通过选择蓝色来激活预览关闭预览按钮。预览显示文件夹中前 10 个文件的前 10 行。下图显示了如何激活从嵌套目录创建的数据集的预览。





9. 在详细信息窗格中，验证或更改名称和文件类型对于您的数据集。如果你添加名称如果包含空格，则导入数据集时，这些空格将被下划线替换为下划线。
10. 指定要使用的采样配置。
11. 选择导入数据集.

从 Athena 导入数据

Amazon Athena 是一种交互式查询服务，让您能够使用标准 SQL 直接在 Amazon S3 中轻松分析数据。只需在 Amazon Web Services Management Console 中执行几项操作，即可将 Athena 指向 Amazon S3 中存储的数据，并开始使用标准 SQL 运行临时查询，然后在几秒钟内获得结果。要了解更多信息，请参阅什么是 Amazon Athena？（在 Amazon Athena 用户指南中）。

您可以查询 Athena 数据库并将结果导入 Data Wrangler。要使用此导入选项，您必须在 Athena 中至少创建一个数据库。要了解如何操作，请参阅开始使用（在 Amazon Athena 用户指南中）。

请注意以下有关 Data Wrangler 中 Athena 导入选项的信息：

- Data Wrangler 支持使用 Athena 工作组来管理Amazonaccount. 有关更多信息，请参阅 。使用工作组控制查询访问和成本.
- Data Wrangler 不支持联合身份查询。

Data Wrangler 在同一中使用默认 Amazon S3 存储桶。AmazonStudio 实例所在的区域用于存储 Athena 查询结果。它在此数据库中创建临时表以将查询输出移动到此 Amazon S3 存储桶。它会在导入数据之后删除这些表；但是数据库，`sagemaker_data_wrangler`，仍然存在。要了解更多信息，请参阅 导入的数据存储 (p. 600)。

如果您使用Amazon Lake Formation使用 Athena，请确保您的 Lake Formation IAM 权限不会覆盖数据库的 IAM 权限`sagemaker_data_wrangler`.

从 Athena 将数据集导入 Data Wrangler

1. 在存储库的导入数据屏幕上，选择Amazon Athena.
2. 适用于Data Catalog中，选择一个数据目录。
3. 使用数据库下拉列表以选择要查询的数据库。选择数据库时，可以使用表s 在下面列出详细信息.
4. 选择高级配置.

   启用采样默认处于选中状态。激活采样后，Data Wrangler 会对查询的数据进行采样并导入大约 50%。取消选中此复选框可关闭采样。
5. 指定的值工作组如果您使用的是。
6. 在查询编辑器中输入您的查询，然后使用运行按钮来运行查询。成功查询后，您可以在编辑器下预览结果。
7. 要导入查询结果，请选择导入.

完成上述过程后，您查询和导入的数据集将显示在 Data Wrangler 流程中。

从 Amazon Redshift 导入数据

Amazon Redshift 是一种完全托管的 PB 级云中数据仓库服务。创建数据仓库的第一步是启动一组节点（称为 Amazon Redshift 集群）。在您预置集群后，您可以上传数据集，然后执行数据分析查询。

您可以在 Data Wrangler 中连接和查询一个或多个 Amazon Redshift 集群。要使用此导入选项，您必须在 Amazon Redshift 中至少创建一个集群。要了解如何操作，请参阅Amazon Redshift 入门.

您可以在以下位置之一输出 Amazon Redshift 查询的结果：

- 默认 Amazon S3 存储桶
- 您指定的 Amazon S3 输出位置

默认 Amazon S3 存储桶位于同一个状态Amazon您的 Studio 实例所在的区域用于存储 Amazon Redshift 查询结果。有关更多信息，请参阅 导入的数据存储 (p. 600)。

对于默认 Amazon S3 存储桶或您指定的存储桶，您都有以下加密选项：

- 默认Amazon使用 Amazon S3 托管密钥 (SSE-S3) 进行服务端加密
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon Key Management Service您指定的 (KMS) 密钥

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon KMSkey 是您创建和管理的加密密钥。有关 KMS 密钥的更多信息，请参阅Amazon Key Management Service.

您可以指定Amazon KMS密钥使用你的钥匙 ARN 或 ARNAmazonaccount.

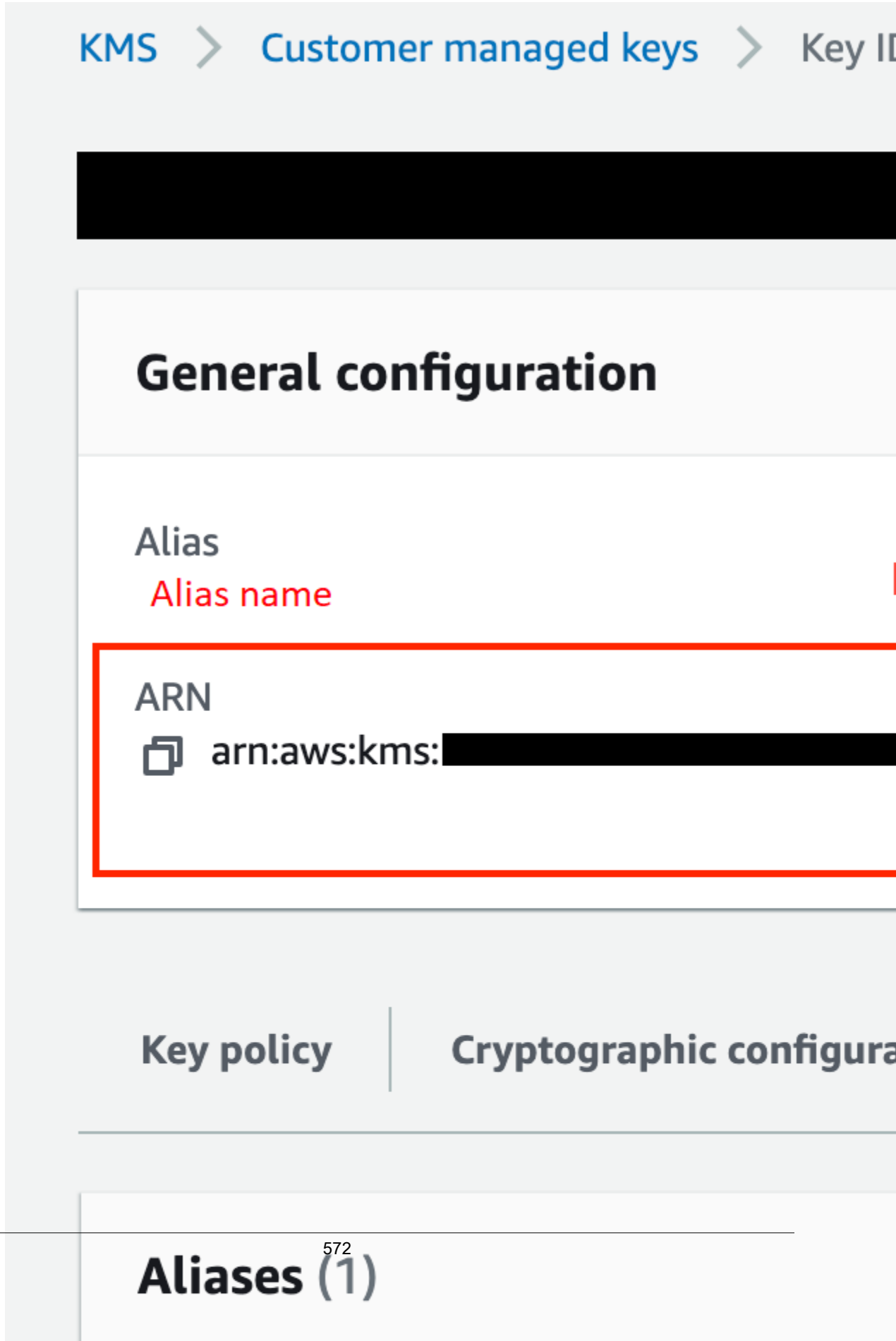
如果您使用 IAM 托管策略，`AmazonSageMakerFullAccess`，要授予在 Studio 中使用 Data Wrangler 的角色权限，您的数据库用户名称必须具有前缀`sagemaker_access`.

使用以下过程了解如何添加新集群。

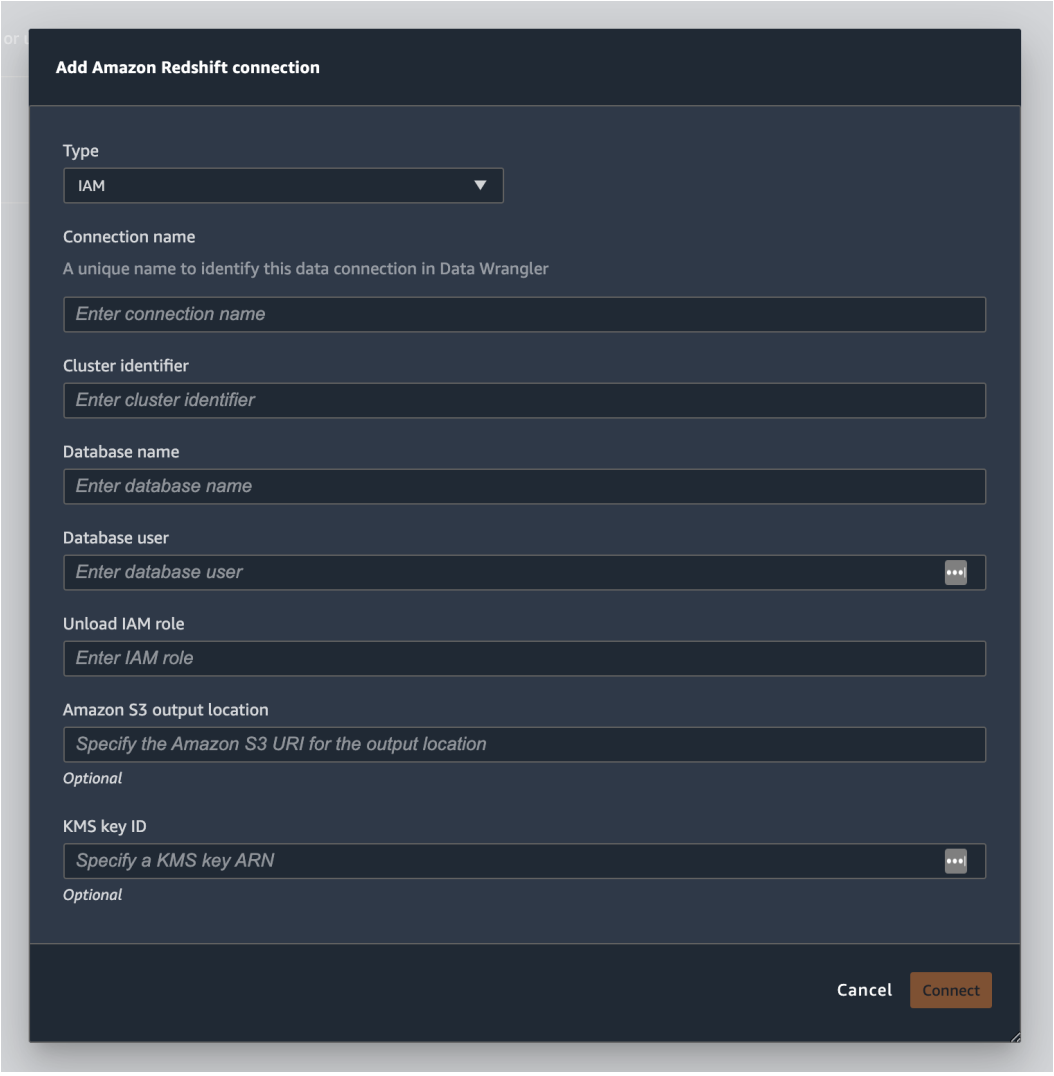
> **Note**
>
> Data Wrangler 将 Amazon Redshift 数据 API 与临时凭证结合使用。要了解有关此 API 的更多信息，请参阅使用 Amazon Redshift 数据 API在 Amazon Redshift 集群管理指南中。

连接到 Amazon Redshift 集群

1. 选择 Import (导入)。
2. 选择+下添加数据连接.
3. 选择Amazon Redshift.
4. 选择临时证书 (IAM)为了类型.
5. 输入连接名称. 这是 Data Wrangler 用来识别此连接的名称。
6. 输入Cluster Identifier指定要连接到哪个集群。注意：仅输入集群标识符，而不是 Amazon Redshift 集群的完整端点。
7. 输入Database Name要连接到的数据库的数据库。
8. 输入数据库用户以确定要用于连接到数据库的用户。
9. 适用于卸载 IAM 角色中，输入 Amazon Redshift 集群在向 Amazon S3 移动和写入数据时应担任的角色的 IAM 角色 ARN。有关该角色的更多信息，请参阅授权 Amazon Redshift 访问其他Amazon代表您提供服务在 Amazon Redshift 集群管理指南中。
10. 选择 Connect（连接）。
11. （可选）对于Amazon S3 输出位置中，指定 S3 URI 以存储查询结果。
12. （可选）对于KMS 密钥 ID，指定的 ARNAmazon KMS密钥或别名。下图显示了在哪里可以在Amazon Web Services Management Console.


KMS
Customer managed keys
Key ID
[redacted]
General configuration
Alias
Alias name
ARN
arn:aws:kms:[redacted]
Key policy
Cryptographic configura
Aliases (1)

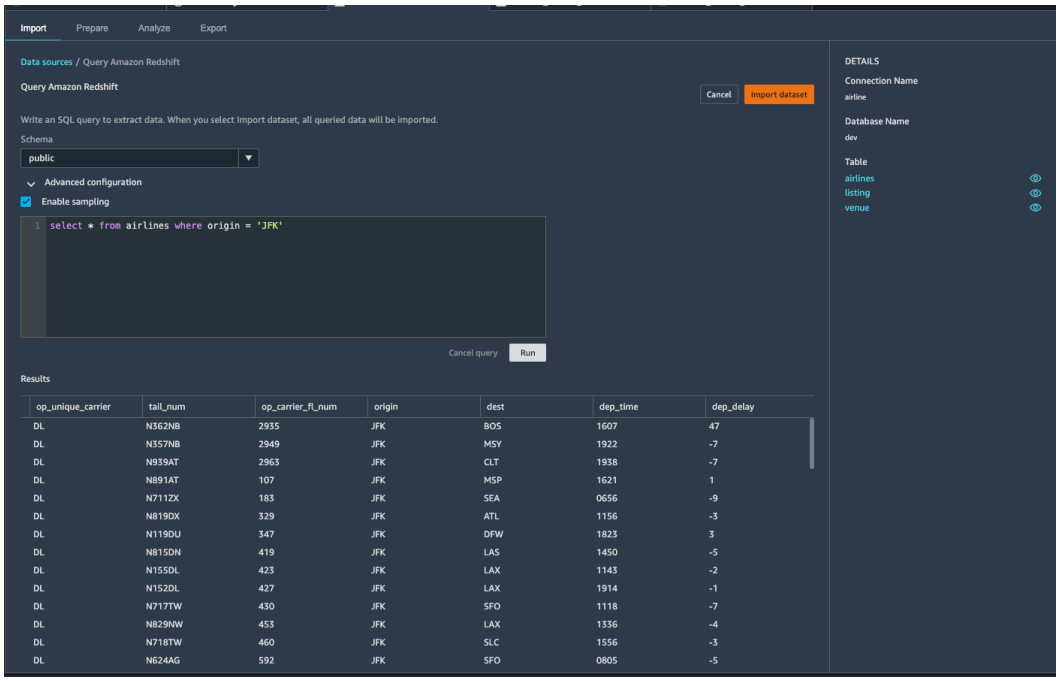
下图显示了上述过程中的所有字段。



成功建立连接后，它将显示为下面的数据源导入数据. 选择此数据源可查询数据库并导入数据。

从 Redshift 查询和导入数据

1. 选择要查询的连接数据源.
2. 选择架构. 要了解有关 Amazon Redshift 模式的更多信息，请参阅Schemas在 Amazon Redshift 数据库开发人员指南中。
3. UNDER高级配置、启用采样默认处于选中状态。如果不取消选中此框，Data Wrangler 会对查询的数据进行抽样并导入大约 50% 的查询数据。取消选中此复选框可关闭采样。
4. 在查询编辑器中输入您的查询，然后使用运行按钮来运行查询。成功查询后，您可以在编辑器下预览结果。
5. Select导入数据集以导入已查询的数据集。
6. 输入数据集名称. 如果你添加数据集名称如果包含空格，则导入数据集时，这些空格将被下划线替换为下划线。
7. 选择 Add（添加）。



从 Databricks (JDBC) 导入数据

您可以使用 Databricks 作为亚马逊的数据源 SageMaker 数据 Wrangler 流。要从 Databricks 导入数据集，请使用 JDBC（Java 数据库连接）导入功能访问 Databricks 数据库。访问数据库后，请指定 SQL 查询以获取数据并导入数据。

我们假设您有一个正在运行的 Databricks 集群，并且您已经为其配置了 JDBC 驱动程序。有关更多信息，请参阅以下 Databricks 文档页面：

- JDBC 驱动程序
- JDBC 配置和连接参数
- 身份验证参数

数据 Wrangler 将你的 JDBC URL 存储在Amazon Secrets Manager. 你必须给你的亚马逊 SageMaker Studio IAM 执行角色使用 Secrets Manager 的权限。可以使用以下过程授予权限。

要将权限授予 Secrets Manager，请执行以下操作。

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择 Roles（角色）。
3. 在搜索栏中，指定亚马逊 SageMaker Amazon 的执行角色 SageMaker 工作室正在使用。
4. 选择 角色。
5. 选择 Add permissions（添加权限）。
6. 选择创建内联策略.
7. 适用于服务，请指定Secrets Manager然后选择它。
8. 适用于操作中，选择旁边的箭头图标权限管理.
9. 选择PutResourcePolicy.
10. 适用于资源，选择特定.

11. 选中旁边的复选框此账户中的任何.
12. 选择 Review policy (审核策略)。
13. 适用于名称，指定名称。
14. 选择 Create policy（创建策略）。

您可以使用分区更快地导入数据。分区使 Data Wrangler 能够 parallel 处理数据。默认情况下，Data Wrangler 使用 2 个分区。对于大多数用例，2 个分区可为您提供近乎最佳的数据处理速度。

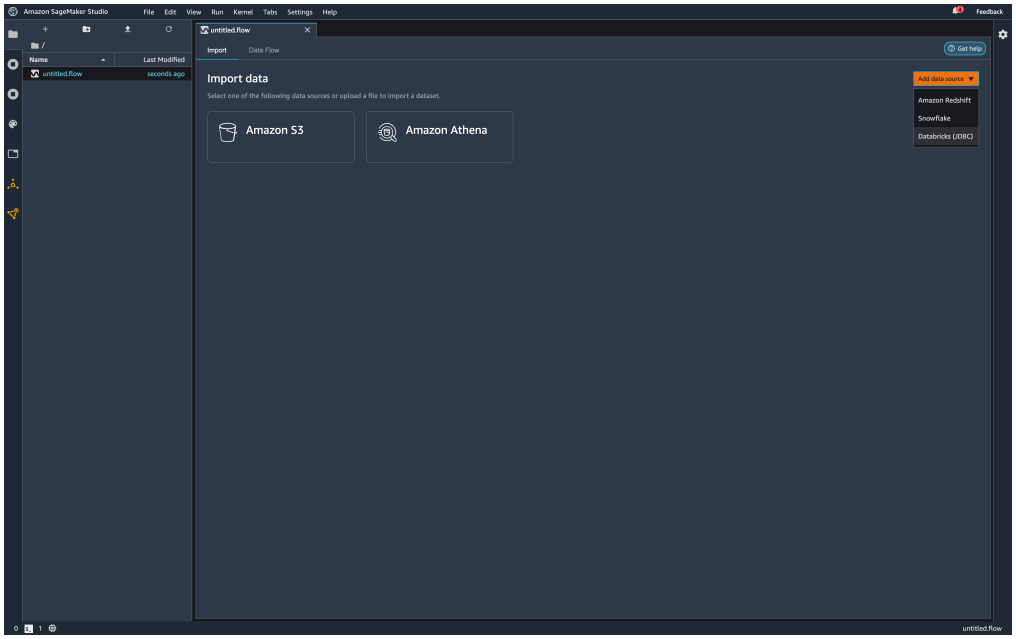
如果选择指定 2 个以上的分区，还可以指定一列来对数据进行分区。列中值的类型必须是数字或日期。

我们建议仅在您了解数据的结构以及如何处理数据时才使用分区。

使用以下过程从 Dabricks 数据库中导入您的数据。

要从 Databricks 导入数据，请执行以下操作。

1. 登录Amazon SageMaker 控制台.
2. 选择工作室.
3. 选择启动应用.
4. 从下拉菜单中，选择工作室.
5. 从导入数据数据 Wrangler 流程的选项卡，选择添加数据源.
6. SelectDatabricks (JDBC).



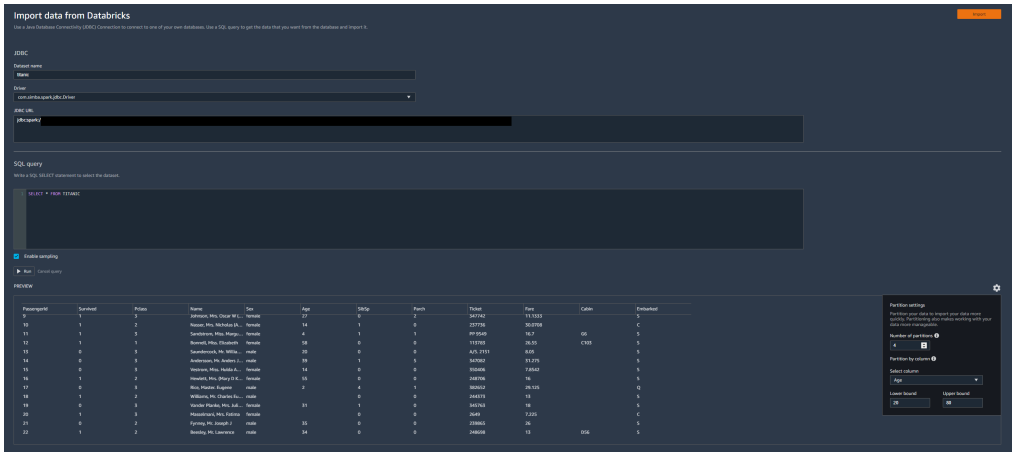
7. 指定以下字段：

   - 数据集名称— 要用于 Data Wrangler 流中的数据集的名称。
   - 驱动程序–com.simba.spark.jdbc.驱动程序.
   - JDBC URL— Databricks 数据库的 URL。Databricks 实例的 URL 格式可能会有所不同。有关查找 URL 并在其中指定参数的信息，请参阅JDBC 配置和连接参数. 以下是如何格式化 URL 的示例：jdbc: spark: //aws-sagemaker-datawrangler.cloud.data ricks.com:443/ 默认; 传输模式 = HTTP; ssl=1; HTTPath = SQL/协议 v1/O/3121950175/0909-200301-cut318; authmech=3; UIDH=1; httppath=SQL/协议v1/O3261950175/0909-200301-cut318; uidmech=3##; PWD=######.

8. 指定 SQL SELECT 语句。

9. （可选）启用采样使用数据集的前 50,000 行。这是导入数据的默认设置。对于大型数据集，如果不对数据进行采样，可能需要很长时间才能导入数据。要导入整个数据集，请关闭采样。
10. 选择 Run（运行）。下图显示了激活采样的查询。

11. （可选）对于预览，选择要打开的齿轮分区设置. 下图显示了指定了可选数据分区设置的查询。



- 指定分区的数量。如果指定分区数，则可以按列进行分区：
  - 输入分区数— 指定一个大于 2 的值。
  - （可选）按列分区— 指定以下字段。如果指定了值，则只能按列进行分区输入分区数.
    - Select 列— 用于数据分区的列。列的数据类型必须是数字或日期。
    - 上限— 从你指定的列中的值中，你在分区中使用的上限值。您指定的值不会更改要导入的数据。它只会影响导入的速度。为了获得最佳性能，请指定接近列最大值的上限。
    - 下限— 从你指定的列中的值中，你在分区中使用的下限值。您指定的值不会更改要导入的数据。它只会影响导入的速度。为了获得最佳性能，请指定接近该列最低值的下限。
12. 选择 Import (导入)。

从 Snowflake 导入数据

你可以使用 Snowflake 作为数据源 SageMaker Data Wrangler 用 Snowflake 为机器学习准备数据。

利用 Snowflake 作为 Data Wrangler 中的数据源，无需编写一行代码即可快速连接到 Snowflake。此外，您可以将 Snowflake 中的数据与存储在 Amazon S3 中的数据以及通过 Amazon Athena 和 Amazon Redshift 查询的数据加入 Snowflake 中，以便为机器学习准备数据。

连接后，您可以通过一组强大的预配置可视化模板以交互方式查询存储在 Snowflake 中的数据、使用 300 多种预配置的数据转换来转换数据、了解数据并识别潜在错误和极端值，快速识别数据中的不一致性准备工作流程，并在模型部署到生产之前诊断问题。最后，您可以将数据准备工作流程导出到 Amazon S3 以便与其他 SageMaker 亚马逊之类的功能 SageMaker Autopilot，亚马逊 SageMaker Amazon Feature Store 和 Amazon SageMaker 模型构建管道。

您可以使用加密查询的输出Amazon Key Management Service您创建的密钥。有关 Amazon KMS 的更多信息，请参阅Amazon Key Management Service。

管理员指南

Important

要了解有关精确访问控制和最佳实践的更多信息，请参安全访问控制.

本部分适用于从内部设置对 Snowflake 的访问权限的 Snowflake 管理员 SageMaker DATA Wangler。

Important

管理员负责管理和监控 Snowflake 中的访问控制。这包括用户可以访问哪些数据、用户可以使用的存储集成以及用户可以运行哪些查询。Data Wrangler 不会添加与 Snowflake 相关的访问控制层。

Important

请注意，授予监视器权限可以允许用户查看对象内的详细信息，例如查询或仓库中的使用情况。

使用数据牧马人配置 Snowflake

要从 Snowflake 导入数据，Snowflake 管理员必须使用 Amazon S3 配置来自 Data Wrangler 的访问权限。

此功能目前在选择加入区域不可用。

要配置访问，请按照以下步骤操作。

1. 配置 S3 存储桶的访问权限。

   Amazon访问控制要求

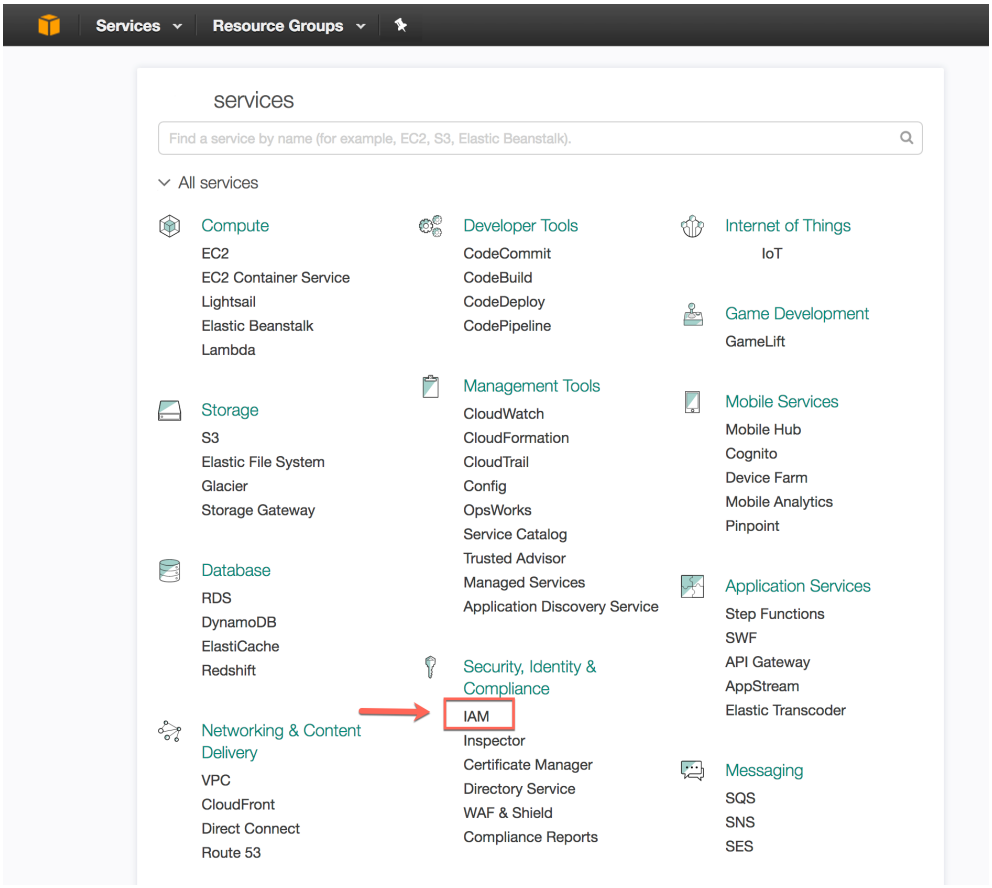
   Snowflake 需要对 S3 存储桶和目录具有以下权限才能访问目录中的文件。

   - `s3:GetObject`
   - `s3:GetObjectVersion`
   - `s3:ListBucket`
   - `s3:ListObjects`
   - `s3:GetBucketLocation`

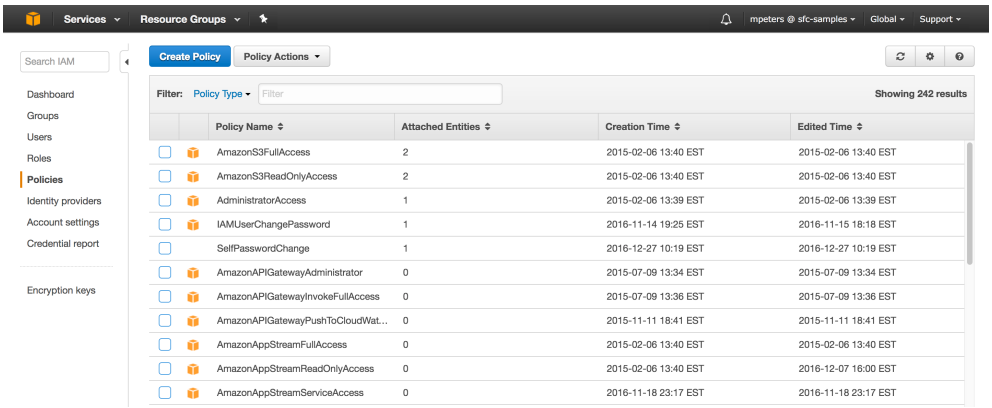
   创建 IAM 策略

   以下步骤介绍了如何在您的Amazon管理控制台以便您可以使用 S3 存储桶加载和卸载数据：

   - 登录到Amazon管理控制台。
   - 从主控制面板中，选择IAM.



- 选择策略。
- 请选择创建策略。



- 选择 JSON 选项卡。
- 添加允许 Snowflake 访问 S3 存储桶和目录的策略文档。

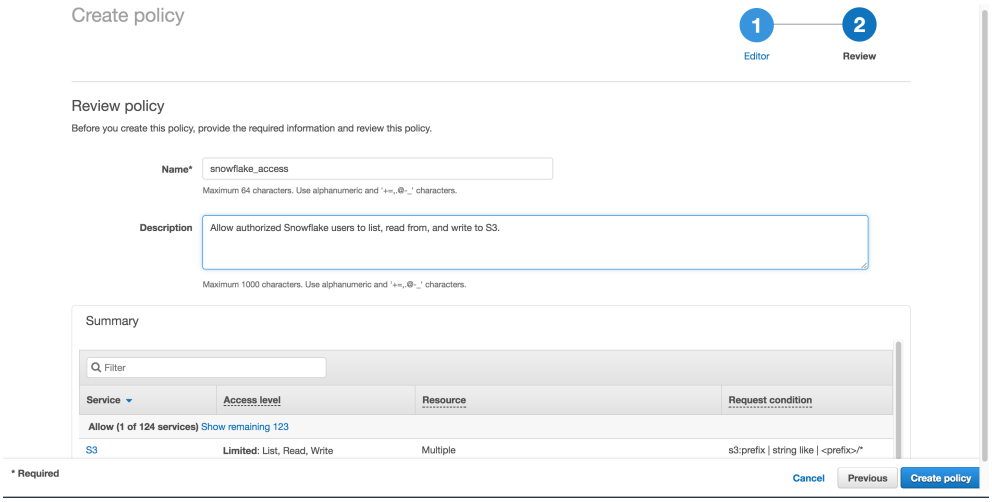
  以下策略（JSON 格式）为 Snowflake 提供了使用单个存储桶和目录路径加载和卸载数据所需的权限。请确保更换`bucket`和`prefix`带有实际存储桶名称和目录路径前缀。

```
# Example policy for S3 write access
# This needs to be updated
{
"Version": "2012-10-17",
```

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion"
    ],
    "Resource": "arn:aws:s3:::bucket/prefix/*"
  },
  {
    "Effect": "Allow",
    "Action": [
        "s3:ListBucket"
    ],
    "Resource": "arn:aws:s3:::bucket/",
    "Condition": {
        "StringLike": {
            "s3:prefix": ["prefix/*"]
        }
    }
  }
 ]
}
```

- 选择 Next:。标签。
- 选择 Next:。审核。

输入策略名称（如`snowflake_access`）和可选的描述。选择 Create policy（创建策略）。



2. 在创建 IAM 角色Amazon.
3. 在 Snowflake 中创建云存储集成.
4. 检索Amazon您的 Snowflake 账户的 IAM 用户.
5. 授予 IAM 用户访问存储桶的权限.
6. 向数据科学家的 Snowflake 角色授予存储集成使用权限。

   - 在 Snowflake 控制台中，运行`GRANT USAGE ON INTEGRATION integration_name TO snowflake_role;`
     - `integration_name`是存储集成的名称。
     - `snowflake_role`默认值的名称Snowflake 角色给数据科学家用户。

向数据科学家提供信息

向数据科学家提供从亚马逊访问 Snowflake 所需的信息 SageMaker DATA Wangler。

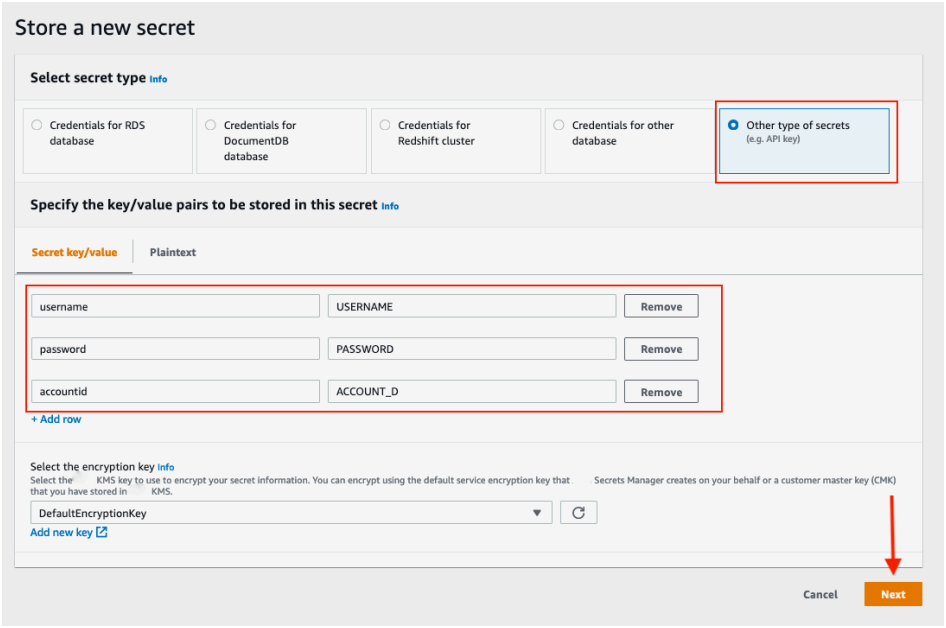
1. 允许数据科学家访问 Snowflake SageMaker Data Wrangler，为他们提供以下服务之一：

   - Snowflake 帐户名称、用户名称和密码。
   - 用创建的秘密AmazonSecrets Manager和密钥的 ARN。如果选择此选项，请使用以下步骤为 Snowflake 创建秘密。

     Important

     如果你的数据科学家使用Snowflake 凭证 (用户名和密码)选项连接到 Snowflake，请注意Secrets Manager用于将证书存储在密钥中，并作为最佳实践安全计划的一部分轮换密钥。只有在设置 Studio 用户配置文件时配置的 Studio 角色才能访问在 Secret Secrets Manager 中创建的密钥。这需要您添加此权限，`secretsmanager:PutResourcePolicy`附加到 Studio 角色的策略。
     我们强烈建议您将角色策略的范围限定为不同的 Studio 用户组使用不同的角色。您可以为 Secrets Manager 添加其他基于资源的权限。请参阅管理秘密策略对于您可以使用的条件键。

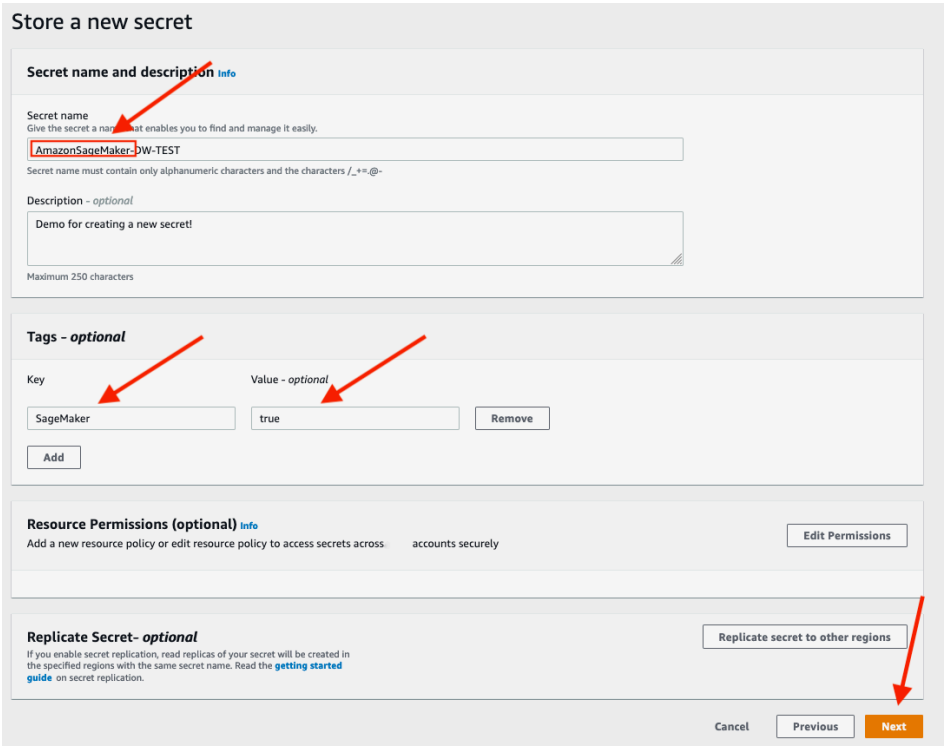
   - 为 Snowflake 创建 Secrets Manager 密钥.
     - 登录到Secrets Manager 控制台.
     - 选择 Store a new secret (存储新密钥)。
     - 在选择密钥类型部分，选择其他密钥类型.



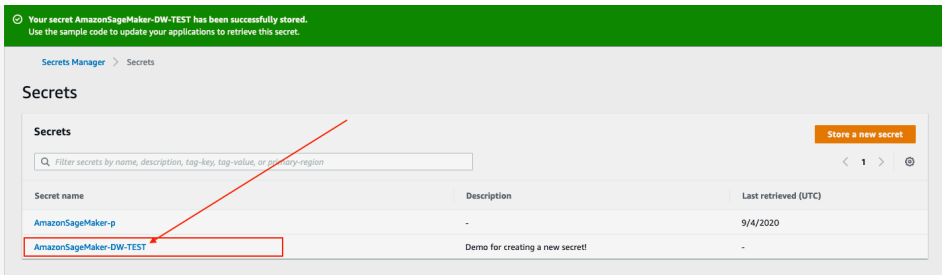
   - 将自定义密钥的详细信息指定为键值对。密钥的名称区分大小写：`username`密钥必须`username`，密码密钥必须是`password`，并且账户 ID 密钥必须是`accountid`. 如果您输入的任何错误，Data Wrangler 将引发错误。的报价`username`、`password`, 和`accountid`如果您使用的是密钥值，则不是必需的。或者，您可以选择明文选项卡，然后在 JSON 中输入密钥值，如以下示例所示：

```
{
        "username": "snowflake username",
        "password": "snowflake password",
        "accountid": "snowflake accountid"
}
```

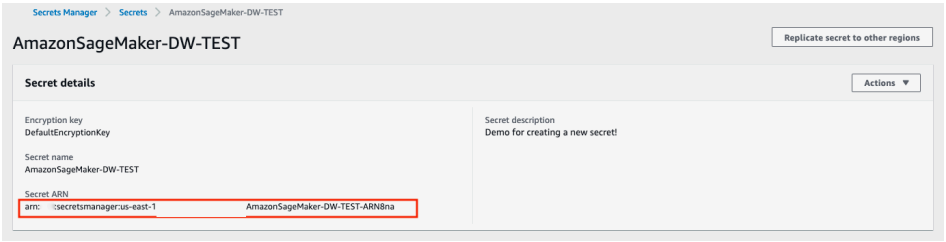
- 选择下一步，然后在下面的屏幕上，在你的秘密名称前加上`AmazonSageMaker-`. 此外，使用密钥添加标签 SageMaker （不带引号）和值：`true`（不带引号）。其余字段是可选的。您可以滚动到页面底部并选择下一步. 屏幕的其余部分是可选的。选择下一步 直到秘密被存储。



- 选择密钥名称并保存密钥的 ARN。选择 Store (存储)。
- 选择您创建的密钥。



- 你在屏幕上看到你的 ARN。如果数据科学家正在使用 ARN 连接到 Snowflake，请向他们提供 ARN。



2. 向数据科学家提供您在步骤 3 中创建的存储集成的名称：在 Snowflake 中创建云存储集成. 这是新集成的名称，被称为`integration_name`中的`CREATE INTEGRATION`您运行的 SQL 命令，如以下代码段所示：

```
CREATE STORAGE INTEGRATION integration_name
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = S3
ENABLED = TRUE
STORAGE_AWS_ROLE_ARN = 'iam_role'
[ STORAGE_AWS_OBJECT_ACL = 'bucket-owner-full-control' ]
STORAGE_ALLOWED_LOCATIONS = ('s3://bucket/path/', 's3://bucket/path/')
[ STORAGE_BLOCKED_LOCATIONS = ('s3://bucket/path/', 's3://bucket/path/') ]
```

数据科学家指南

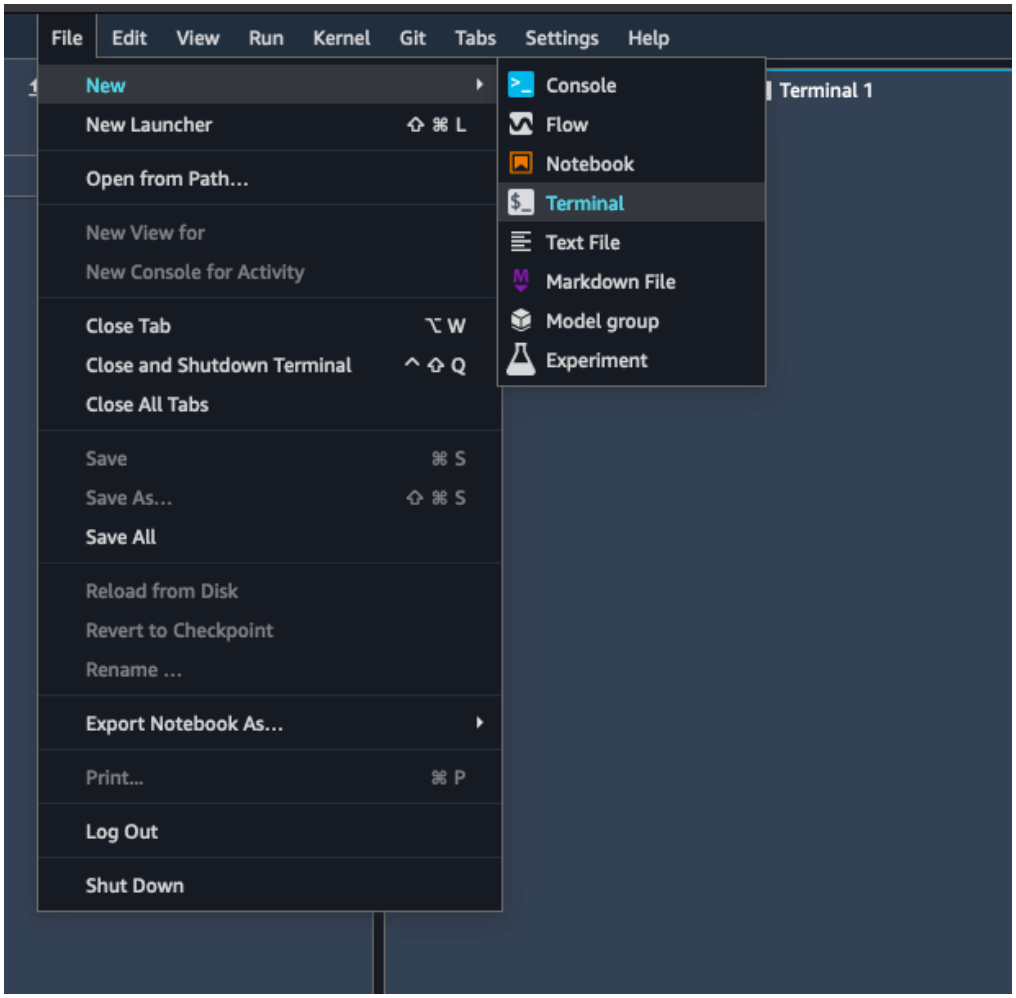
本节概述了如何从 SageMaker Data Wrangler 内部访问 Snowflake 数据仓库以及如何使用 Data Wrangler 功能。

> **Important**
>
> 注意：管理员需要遵循上一节中设置的管理指南，然后才能在 Snowflake 中使用 Data Wrangler。

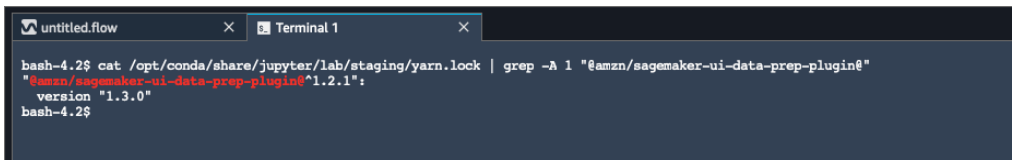
可以使用以下过程打开 Amazon SageMaker Studio 并查看您正在运行的版本。

要打开 Studio 并检查其版本，请参阅以下过程。

1. 使用中的步骤先决条件 (p. 557)通过亚马逊访问 Data Wrangler SageMaker Studio。
2. 在要用于启动 Studio 的用户旁边，选择启动应用.
3. 选择工作室.
4. 加载 Studio 后，选择File (文件)，那么New，然后终端.



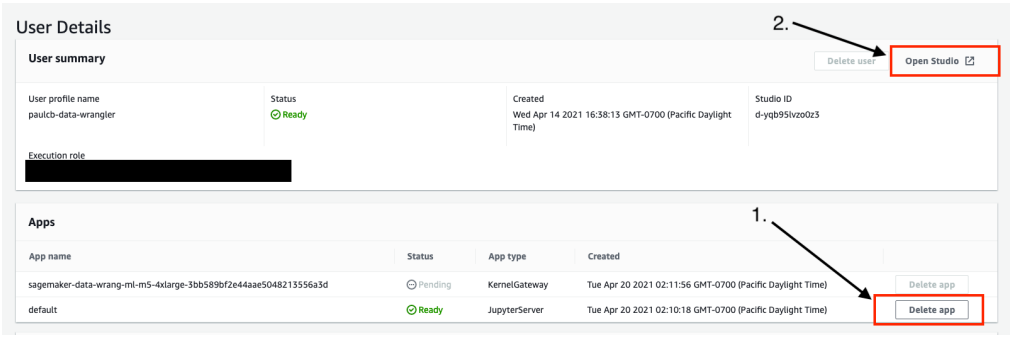
5. 启动 Studio 后，选择File (文件)，那么New，然后终端.
6. Enter`cat /opt/conda/share/jupyter/lab/staging/yarn.lock | grep -A 1 "@amzn/sagemaker-ui-data-prep-plugin@"`以打印 Studio 实例的版本。要使用 Snowflake，您必须拥有 Studio 1.3.0 版本。



请按照以下步骤检查您是否正在运行版本 1.3.0 或更高版本。

要检查 Studio 版本，请执行以下操作。

1. 如果你没有此版本，请更新你的 Studio 版本。为此，请关闭 Studio 窗口并导航到SageMaker Studio 控制台.
2. 接下来，选择用于访问 Studio 的用户，然后选择删除应用. 删除完成后，通过选择选择重新启动 Studio打开工作室.

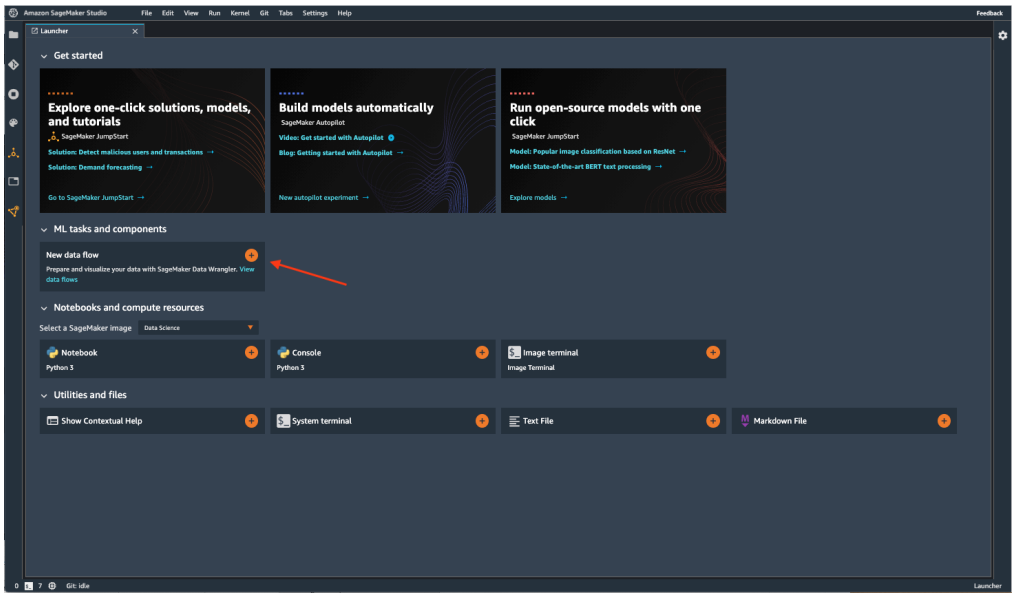


3. 再次按照步骤 3 验证您的 Studio 版本是 1.3.0。

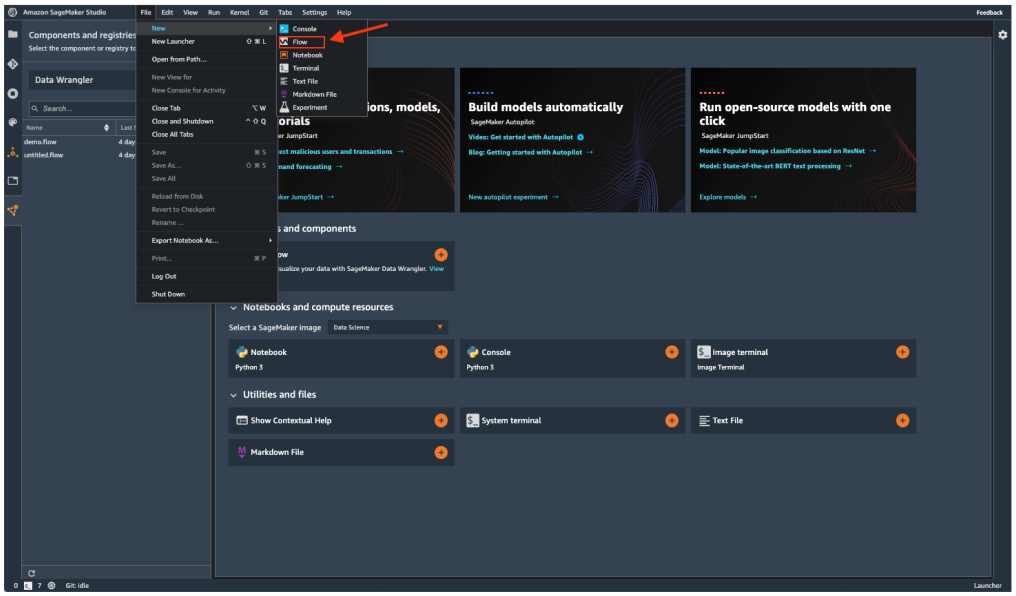
可以使用以下过程连接到 Snowflake。

1. 从 Data Wrangler 内部创建新的数据流

   从 Studio 中访问 Data Wrangler 并拥有 1.3.0 版本后，请选择+登录新的数据流下卡ML 任务和组件. 这将在 Studio 中创建一个包含 .flow 文件的新目录，其中包含数据流。.flow 文件将在 Studio 中自动打开。



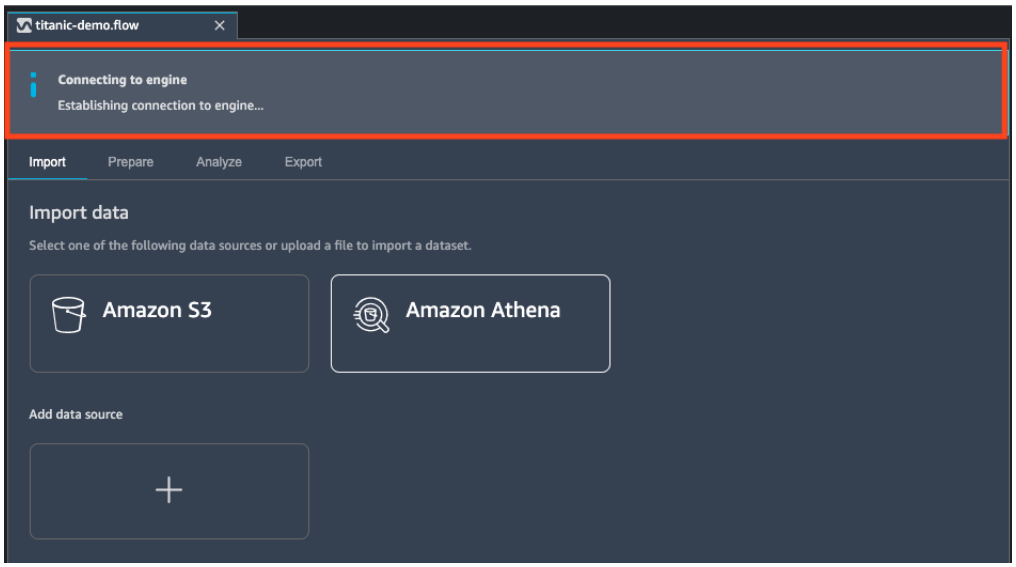
   或者，您也可以通过选择创建新流程。File (文件)，那么New，然后选择流.



在 Studio 中创建新的 .flow 文件时，您可能会在 Data Wrangler 界面顶部看到一条消息，其中显示：

连接到引擎

正在建立与引擎的连接...



2. Connect 到雪花。

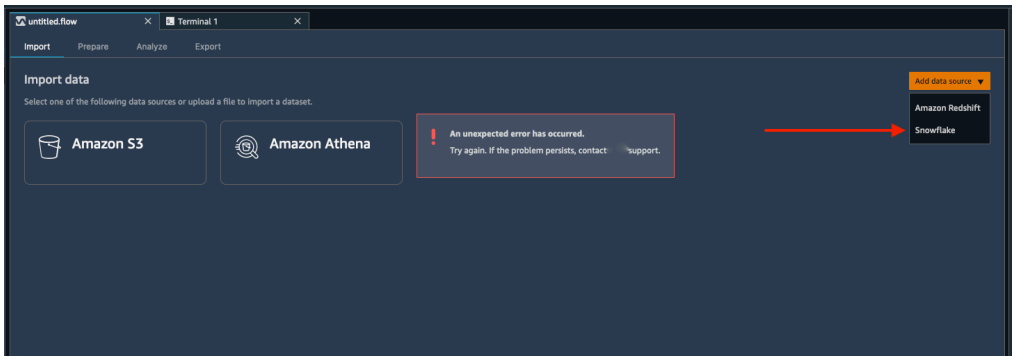
   可通过两种方式从 Data Wrangler 内部连接到 Snowflake。你只需要选择两种方法中的一种。

   1. 在 Data Wrangler 中指定 Snowflake 凭据（帐户名称、用户名和密码）。
   2. 提供密钥的 Amazon 资源名称 (ARN)。

      Important

      如果你没有 Snowflake 凭据或 ARN，请联系管理员。管理员可以告诉你以上哪种方法连接到 Snowflake。

从 “开始”导入数据屏幕然后首先选择添加数据源从下拉菜单中，选择雪花. 以下屏幕截图说明了在哪里可以找到雪花选项。



选择身份验证方法。对于此步骤，如前所述，您可以使用 Snowflake 凭据或 ARN 名称。其中之一是由您的管理员提供。

接下来，我们解释两种身份验证方法并为每种方法提供截图

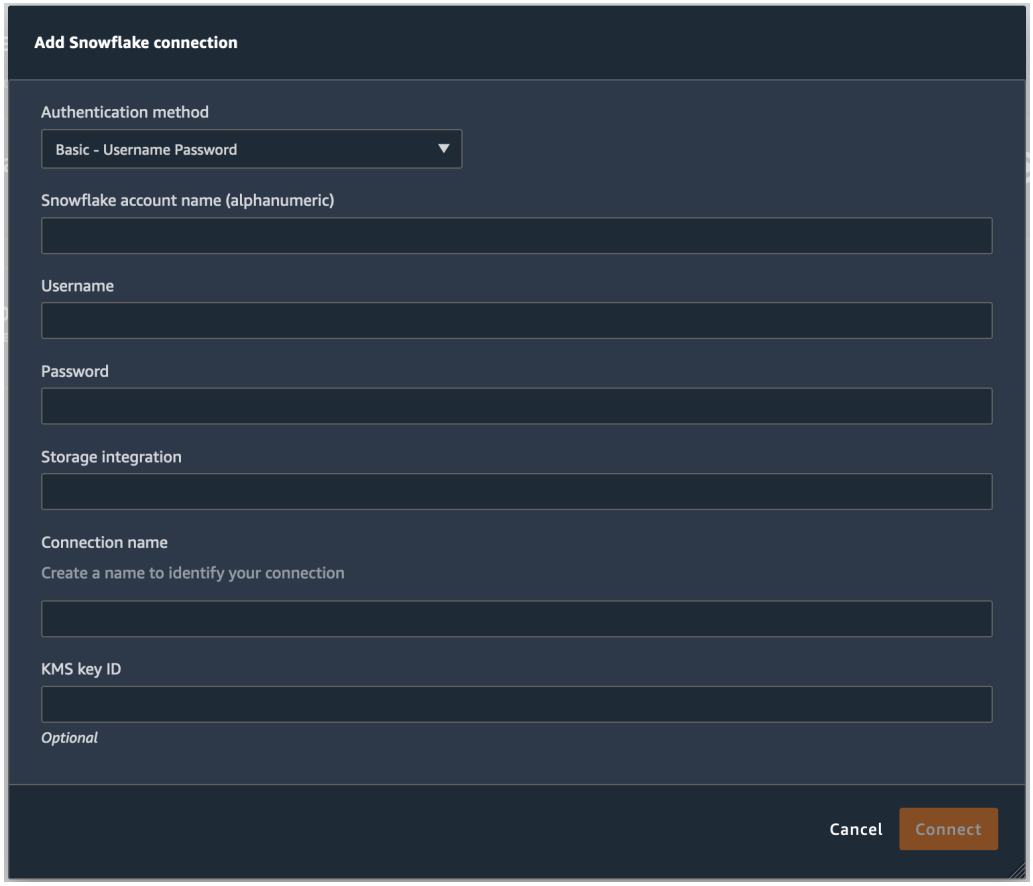
1. Snowflake 凭据选项.

   Select基本选项中的（用户名和密码）身份验证方法下拉列表。然后，在以下字段中输入您的凭证：

   - 存储集成：提供存储集成的名称。管理员提供了此名称。
   - Snowflake 账户名称：Snowflake 账户的全名。
   - 用户名：Snowflake 账户的用户名。
   - 密码：Snowflake 账户密码。
   - 连接名称：选择连接名称供您选择。
   - （可选）KMS 密钥 ID：选择Amazon KMS key对 Snowflake 查询的输出进行加密。有关 Amazon Key Management Service 的更多信息，请参阅https://docs.amazonaws.cn/kms/latest/developerguide/overview.html。如果您未指定Amazon KMS密钥，Data Wrangler 使用默认的 SSE-KMS 加密方法。

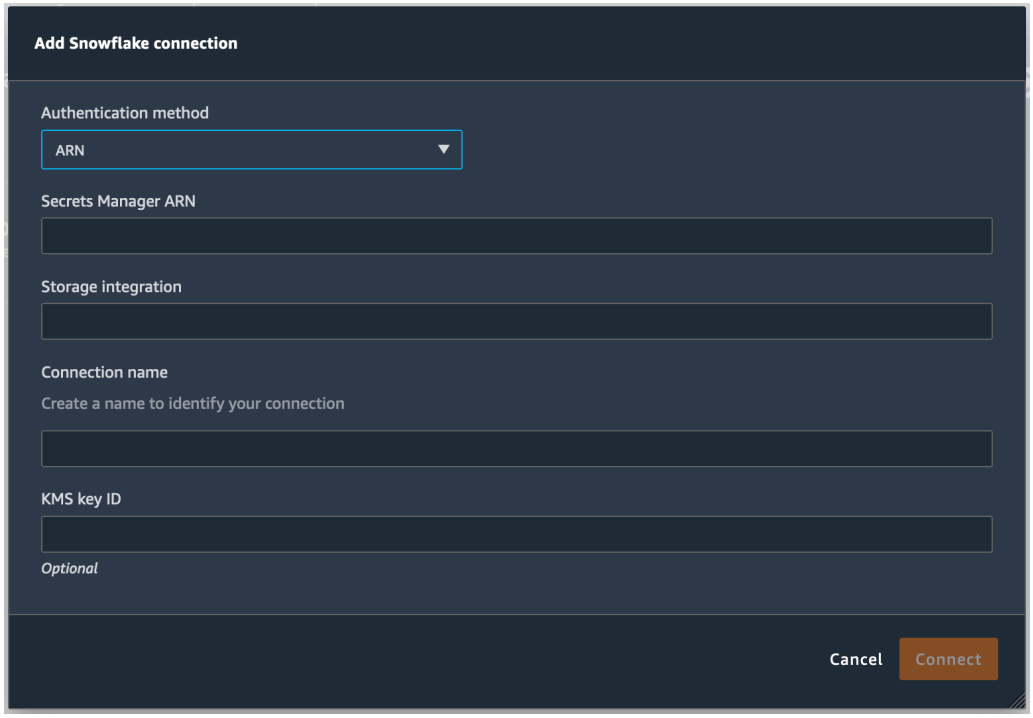
   选择 Connect（连接）。

   以下屏幕截图演示如何填写这些字段。



2. ARN 选项

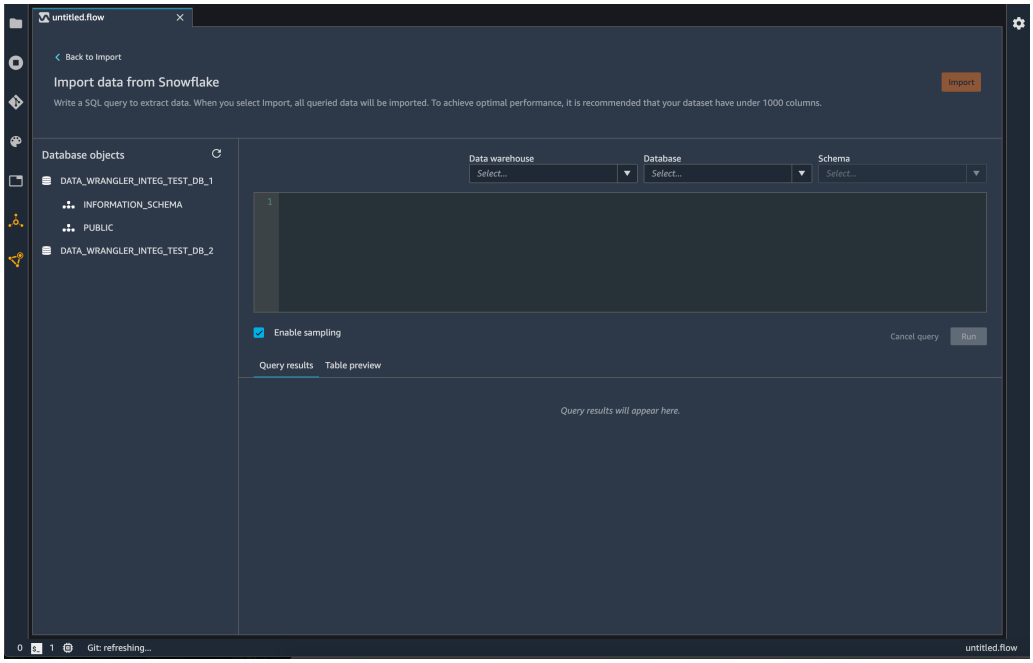
   从身份验证方法下拉列表。然后，在下面提供你的 ARN 名称ARN Secrets Manager和你的存储集成，由您的管理员提供。如果您创建了 KMS 密钥，则可以将其 ID 指定为KMS 密钥 ID. 有关 Amazon Key Management Service 的更多信息，请参阅https://docs.amazonaws.cn/kms/latest/developerguide/overview.html。创建连接名称然后选择Connect (连接)，如以下屏幕截图所示。



3. 此时的工作流程是将 Snowflake 帐户连接到 Data Wrangler，然后对数据运行一些查询，然后最后使用 Data Wrangler 执行数据转换。

   以下步骤解释了 Data Wrangler 内部的导入和查询步骤。

   在创建 Snowflake 连接之后，您将其带到从 Snowflake 导入数据屏幕，如以下屏幕截图所示。

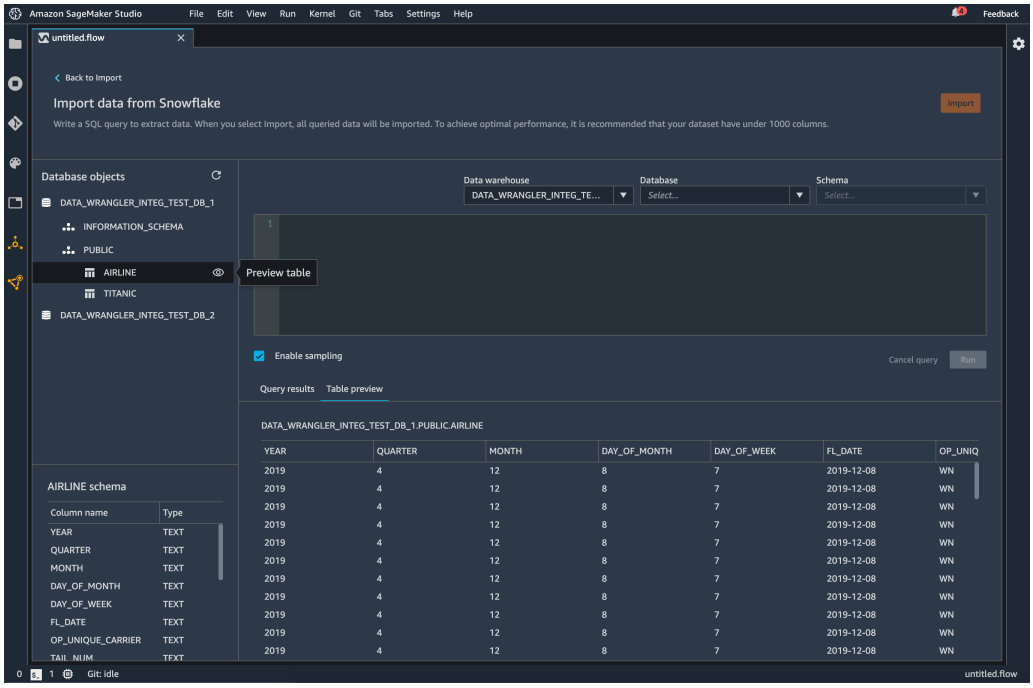


   从这里选择你的仓库。您还可以选择您的数据库和架构，在这种情况下，书面查询应该指定它们。如果数据库和架构在下拉列表中提供，书面查询不需要指定数据库和架构名称。

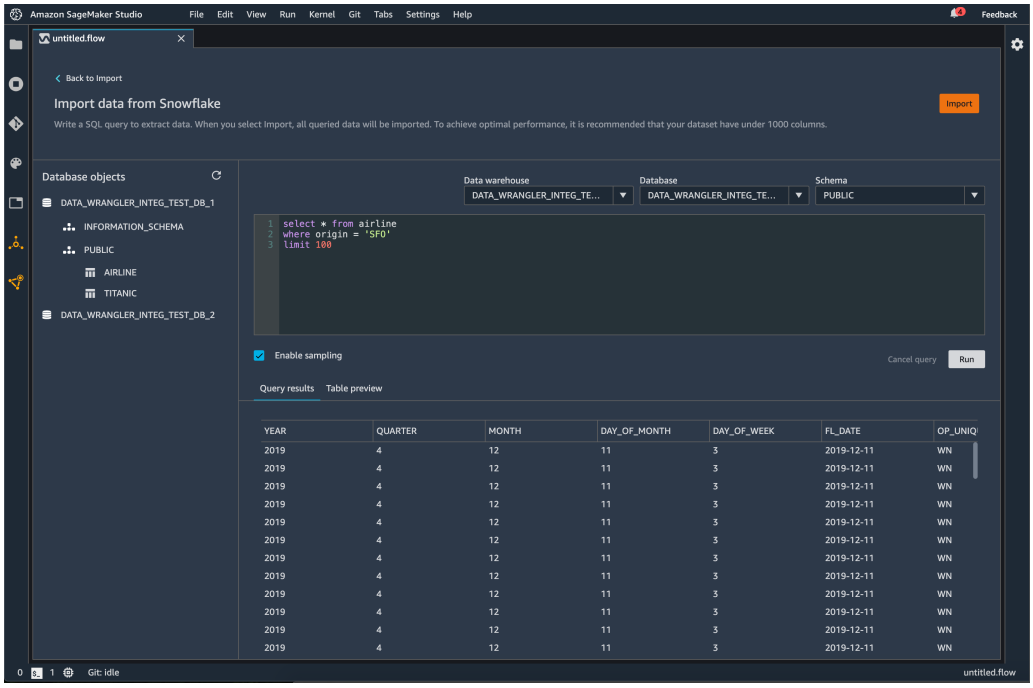
左侧面板中列出了 Snowflake 账户中的模式和表格。你可以选择和解开这些实体。选择特定表时，选择每个表名右侧的眼睛图标以预览表格。

> **Important**
>
> 如果要导入包含类型列的数据集`TIMESTAMP_TZ`要么`TIMESTAMP_LTZ`添加`::string`转到查询的列名称。有关更多信息，请参阅 。如何：将 TIMESTAMP_TZ 和 TIMESAMP_LTZ 数据卸载到实木复合地板文件.

以下屏幕截图显示了包含数据仓库、数据库和架构的面板，以及可用于预览表格的眼睛图标。一旦你选择了预览表格图标，将生成该表的模式预览。您必须先选择仓库，然后才能预览表格。

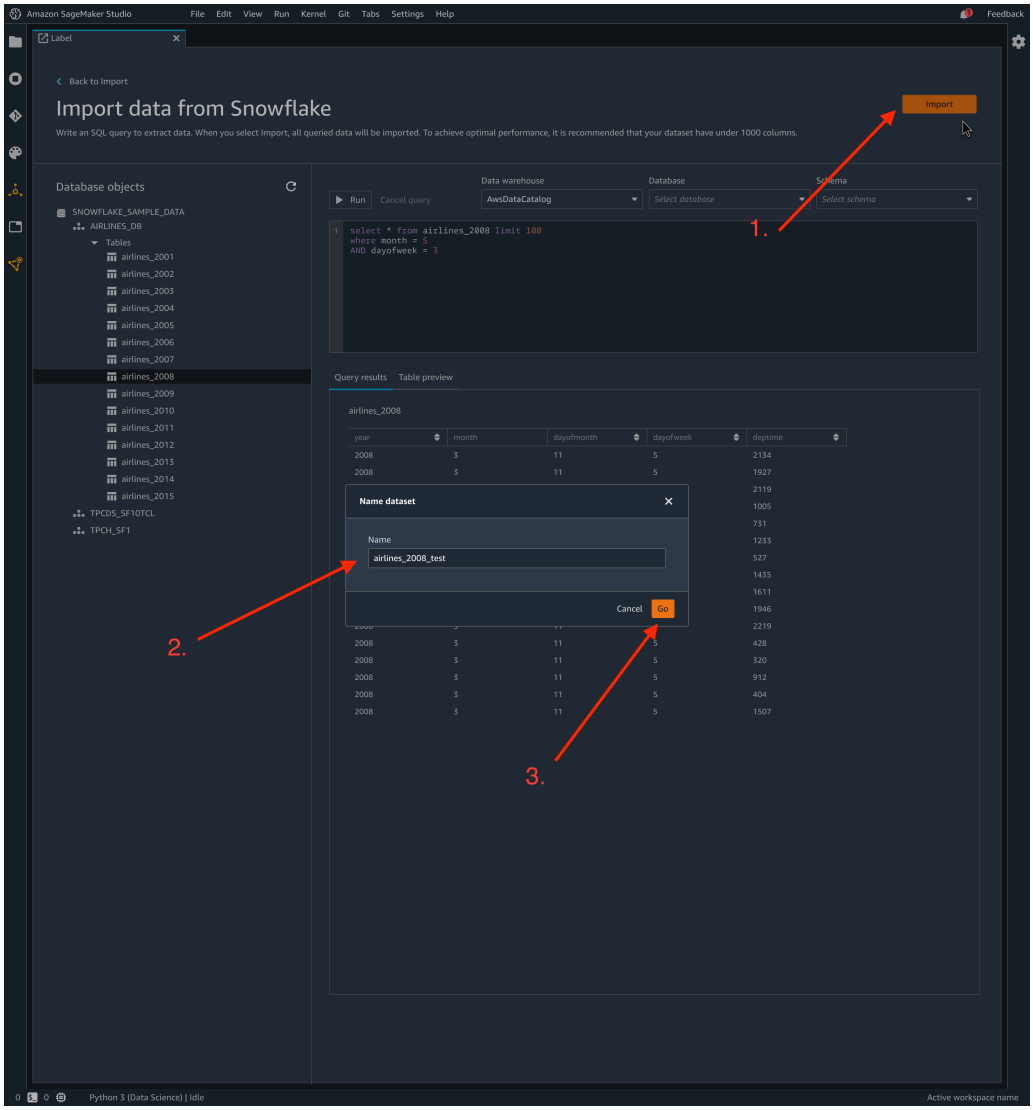


选择数据仓库、数据库和架构后，您现在可以编写查询并运行查询。查询的输出显示在查询结果，如以下屏幕截图所示。

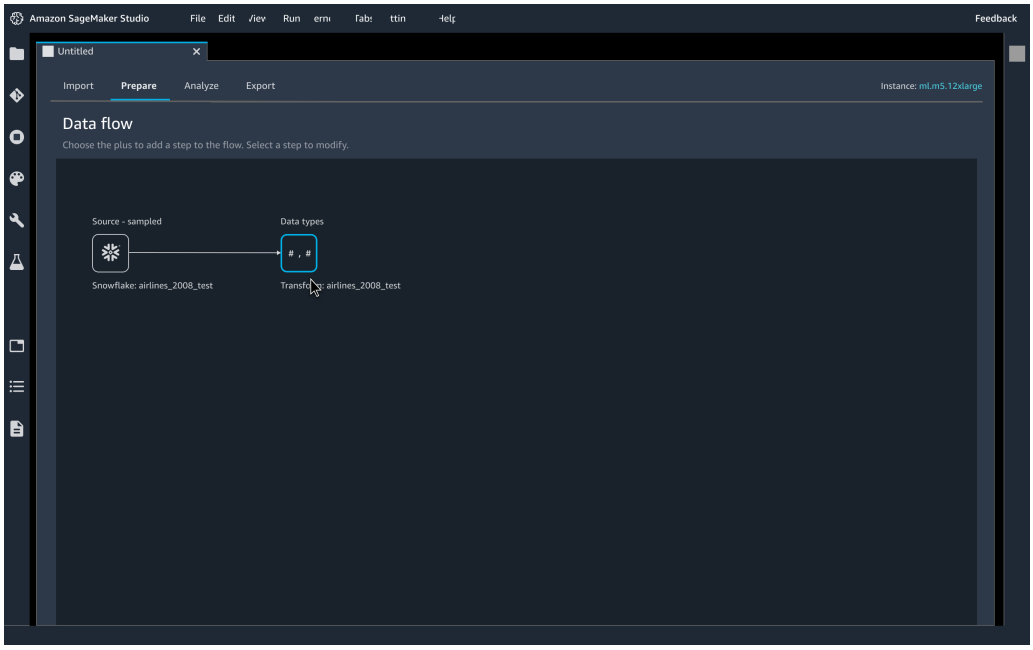


在确定查询的输出之后，您可以将查询的输出导入 Data Wrangler 流程以执行数据转换。

要执行此操作，请选择导入, t然后指定名称然后选择转到，如以下屏幕截图所示。



从这里开始，过渡到数据流屏幕以准备您的数据转换，如以下屏幕截图所示。



数据 Wrangler 和 Snowflake 之间的私人连接方式Amazon PrivateLink

本部分介绍如何使用Amazon PrivateLink在 Data Wrangler 和 Snowflake 之间建立私有连接。以下各节将介绍这些步骤。

创建 VPC

如果您没有设置 VPC，请按照创建新 VPC创建一个的说明。

选择了要用于建立私有连接的 VPC 后，请向 Snowflake 管理员提供以下凭据以启用Amazon PrivateLink：

- VPC ID.
- Amazon账户 ID。
- 您用于访问 Snowflake 的相应账户 URL。

> **Important**
>
> 根据 Snowflake 的文档，启用 Snowflake 帐户最多可能需要两个工作日。

设置 SnowflakeAmazon PrivateLink集成

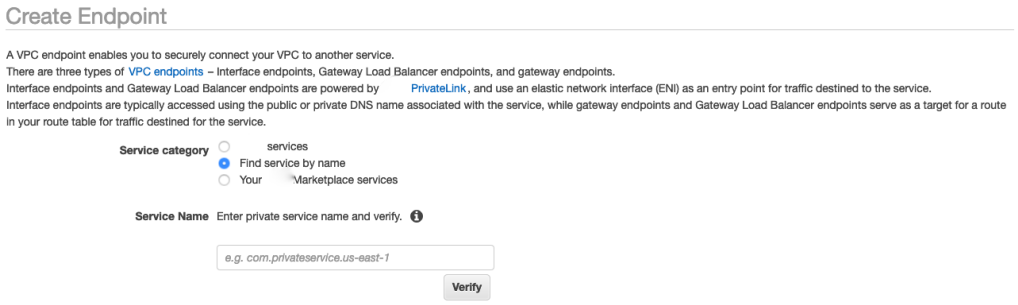
晚于Amazon PrivateLink已激活，检索Amazon PrivateLink通过在 Snowflake 工作表中运行以下命令为您所在区域进行配置。登录 Snowflakes 控制台，在工作表下输入以下内容：`select SYSTEM $GET_PRIVATELINK_CONFIG();`

1. 检索以下内容的值：`privatelink-account-name`、`privatelink_ocsp-url`、`privatelink-account-url`, 和`privatelink_ocsp-url`从生成的 JSON 对象中。下面的代码段显示了每个值的示例。将这些值存储以便将来使用。

```
privatelink-account-name: xxxxxxxx.region.privatelink
privatelink-vpce-id: com.amazonaws.vpce.region.vpce-svc-xxxxxxxxxxxxxxxxx
privatelink-account-url: xxxxxxxx.region.privatelink.snowflakecomputing.com
privatelink_ocsp-url: ocsp.xxxxxxxx.region.privatelink.snowflakecomputing.com
```

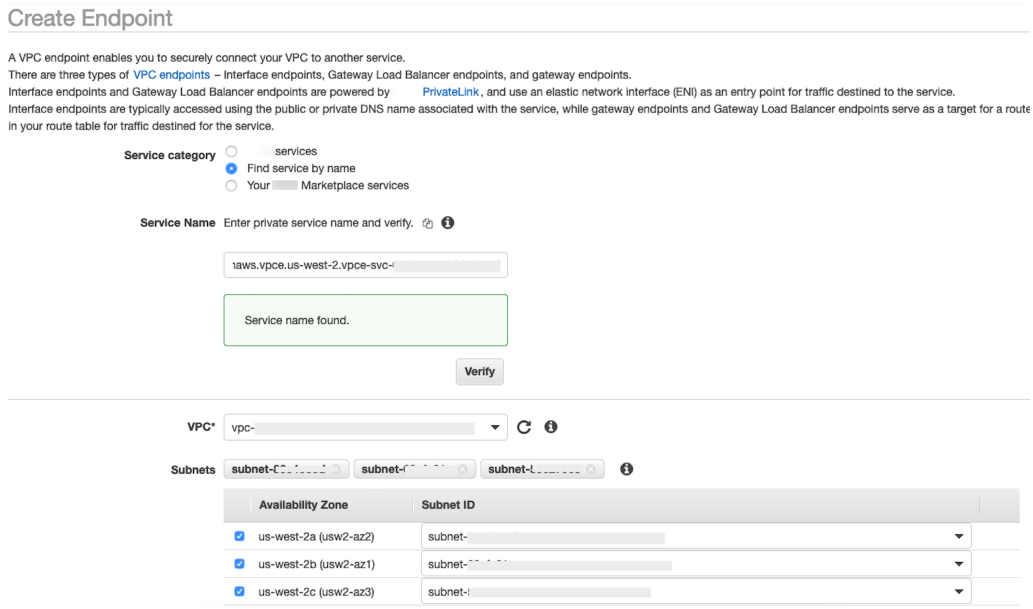
2. 切换到Amazon控制台然后导航到 VPC 菜单。
3. 从左侧面板中，选择终端节点链接以导航到VPC 终端节点设置。

   一旦到达那里，选择创建终端节点.
4. 选择单选按钮按名称查找服务，如以下屏幕截图所示。



5. 在服务名称字段中，粘贴的值`privatelink-vpce-id`在上一步中检索到的，然后选择Verify.

   如果连接成功，绿色警报表示已找到服务名称显示在您的屏幕上，VPC和子网选项将自动展开，如以下屏幕截图所示。根据您的目标区域，生成的屏幕可能会显示另一个屏幕Amazon区域名称。

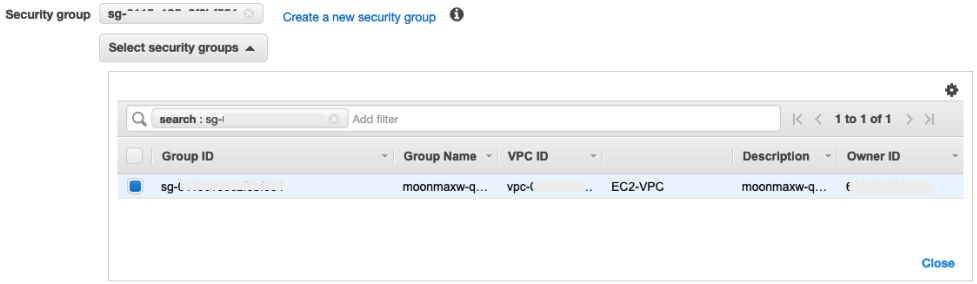


6. 从中选择您发送给 Snowflake 的同一 VPC ID。VPC下拉列表。
7. 如果尚未创建子网，请执行以下有关创建子网的说明。
8. SelectSubnets来自 的VPC下拉列表。然后选择创建子网然后按照提示在 VPC 中创建子集。确保选择您发送的 Snowflake 的 VPC ID。
9. UNDER安全组配置Select,创建新安全组打开默认设置安全组屏幕显示在新选项卡中。在这个新选项卡中，选择 t创建安全组.
10. 为新安全组提供名称（例如`datawrangler-doc-snowflake-privatelink-connection`）和描述。请务必选择您在之前的步骤中使用的 VPC ID。
11. 添加两个规则以允许从您的 VPC 内部到此 VPC 终端节点的流量。

    在下导航到您的 VPC您的 VPC在单独的选项卡中，然后为 VPC 检索 CIDR 块。然后选择 。添加规则中的入站规则部分。Select`HTTPS`对于类型，保留源如同Custom (自定义)在表单中，然后粘贴从前面的中检索到的值`describe-vpcs`呼叫（例如`10.0.0.0/16`）。

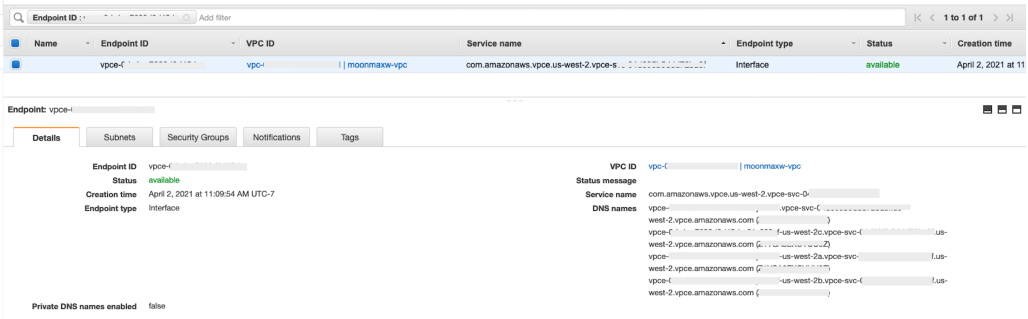
12选择 Create Security Group（创建安全组）。检索安全组 ID来自新创建的安全组（例如sg-xxxxxxxxxxxxxxxxx)。

13在VPC 终端节点配置屏幕中，删除默认安全组。将安全组 ID 粘贴到搜索字段中，然后选中该复选框。



14Select创建终端节点.

15如果终端节点创建成功，您将看到一个页面，其中包含指向 VPC 终端节点配置的链接，该页面由 VPC ID 指定。选择链接可查看完整配置。

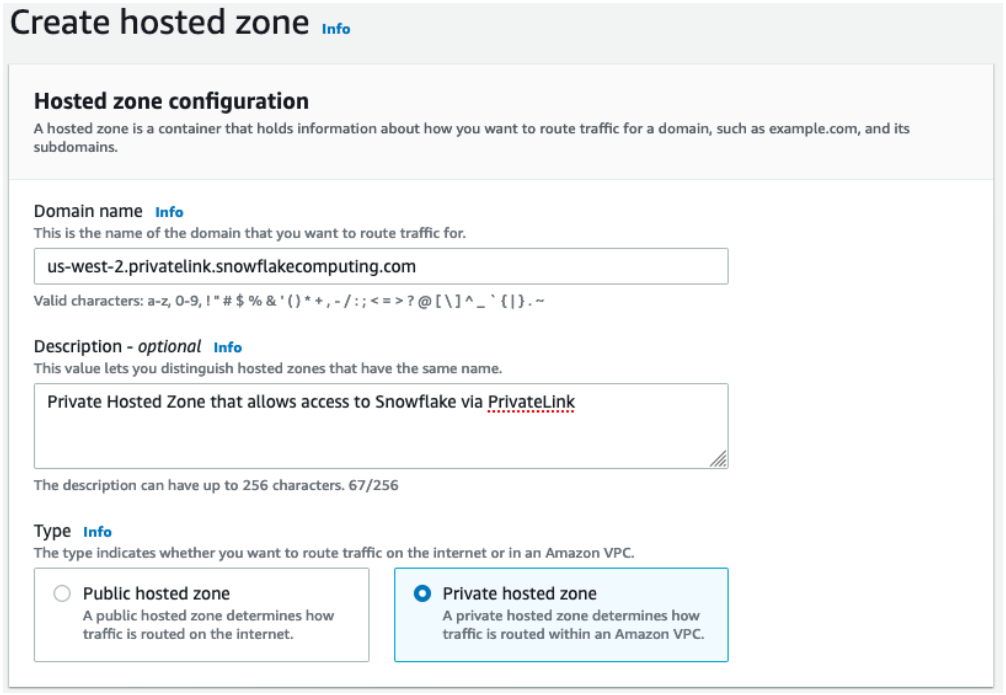


检索 DNS 名称列表中最顶层的记录。这可以与其他 DNS 名称区分开来，因为它只包含区域名称（例如us-west-2），并且没有可用区域字母表示法（例如us-west-2a)。请将此信息存储以便将来使用。

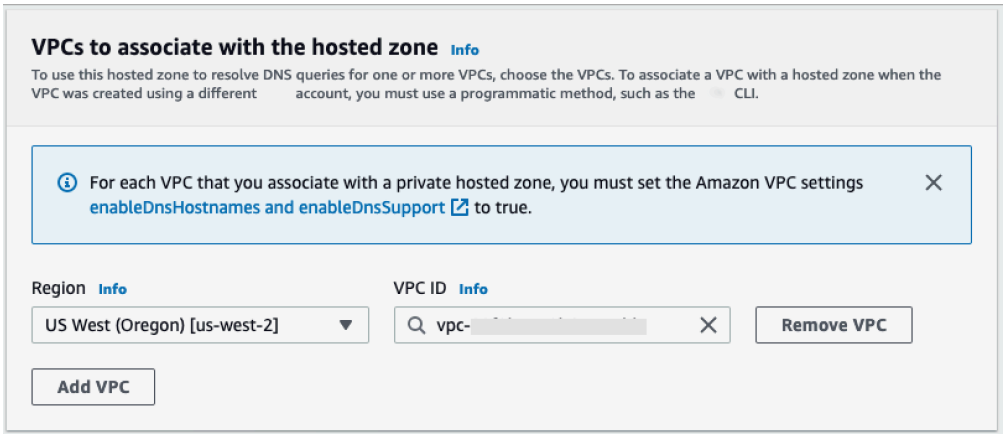
在您的 VPC 中为 Snowflake 终端节点配置 DNS

本部分介绍如何在 VPC 中为 Snowflake 终端节点配置 DNS。这允许您的 VPC 解决对 Snowflake 的请求 Amazon PrivateLink终端节点。

1. 导航到Route 53 菜单在您的Amazon控制台。
2. Select托管区域选项（如有必要，请展开左侧菜单以找到此选项）。
3. 选择 Create Hosted Zone（创建托管区域）。
   a. 在域名字段中，引用存储的值privatelink-account-url在前面的步骤中。在此字段中，您的 Snowflake 账户 ID 将从 DNS 名称中删除，并且仅使用以区域标识符开头的值。一个资源记录集稍后也为子域创建，例如，region.privatelink.snowflakecomputing.com.
   b. 选择单选按钮私有托管区域中的类型部分。你的地区代码可能不是us-west-2. 请参考 Snowflake 返回给你的 DNS 名称。

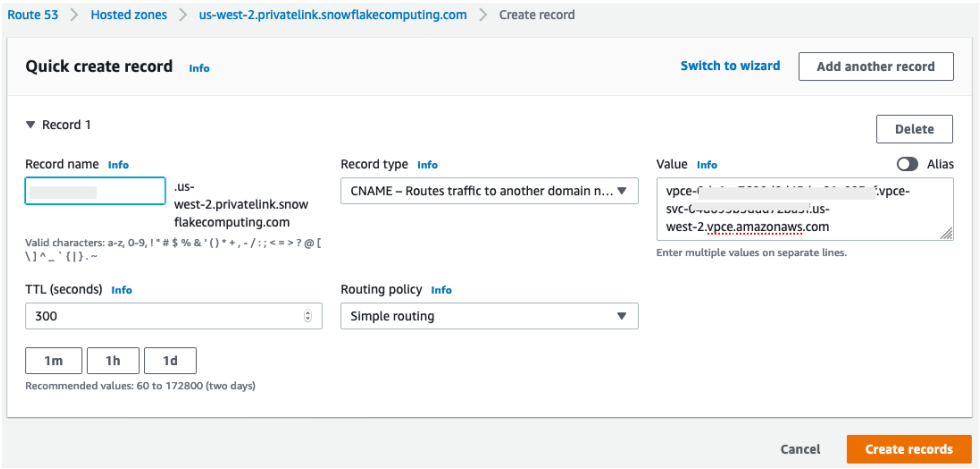


c. 在要与托管区域关联的 VPC部分中，选择 VPC 所在的区域和之前步骤中使用的 VPC ID。



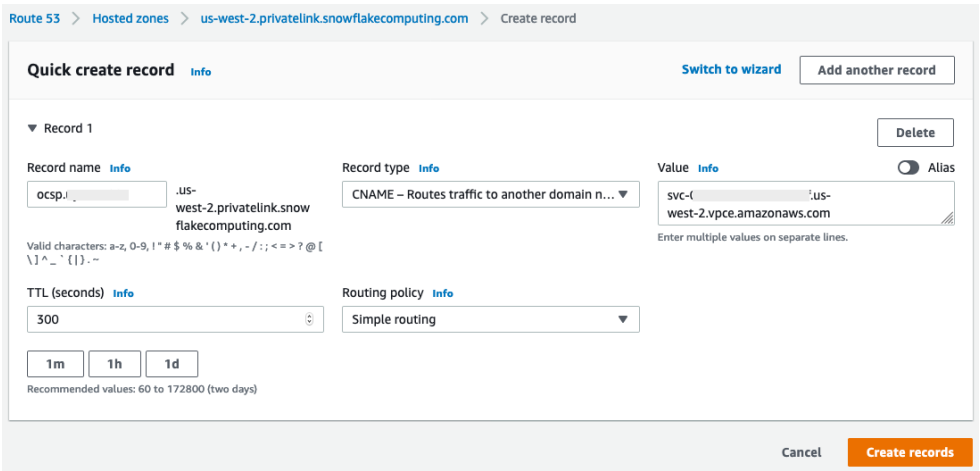
d. 选择 Create Hosted Zone（创建托管区域）。

4. 接下来，创建两条记录，一条用于`privatelink-account-url`另一个用于`privatelink_ocsp-url`.
   - 在托管区域选择菜单，选择创建记录集.
     a. UNDER记录名称中，仅输入您的 Snowflake 账户 ID（中的前 8 个字符）`privatelink-account-url`)。
     b. UNDER记录类型Select,CNAME.
     c. UNDER值中，输入您在最后一步中检索的区域 VPC 终端节点的 DNS 名称设置雪花Amazon PrivateLink集成部分。



d. 选择创建记录。

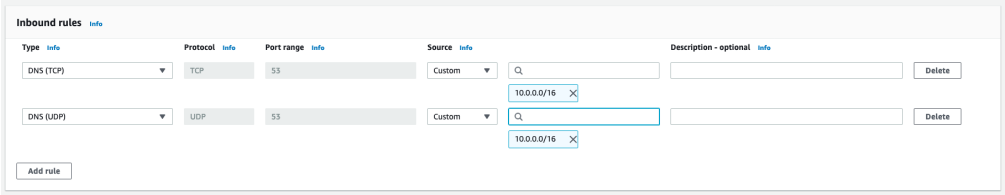
e. 对我们标记为的 OCSP 记录重复上述步骤。`privatelink-ocsp-url`，从开头`ocsp`通过 8 个字符的 Snowflake ID 作为记录名称（例如`ocsp.xxxxxxxx`）。



为您的 VPC 配置 Route 53 解析器入站终端节点

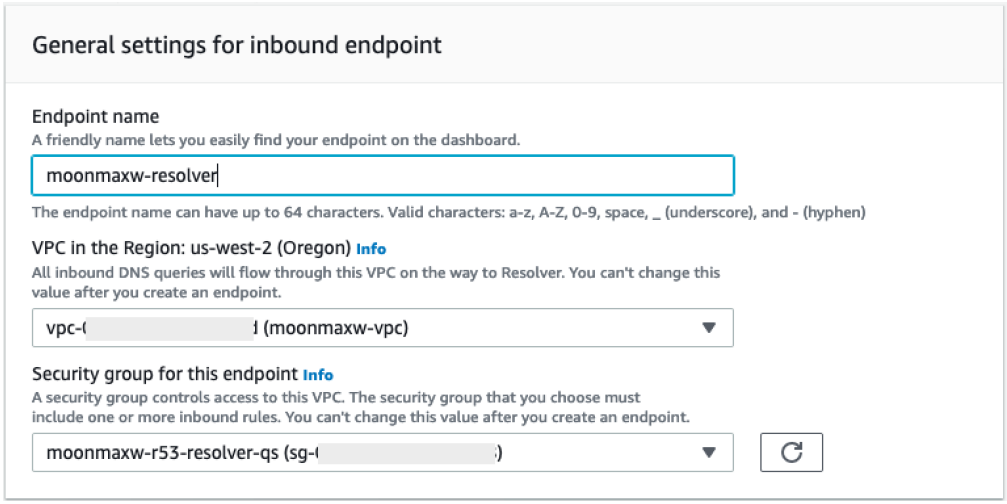
本部分介绍了如何为 VPC 配置 Route 53 解析器入站端点。

1. 导航到Route 53 菜单在您的Amazon控制台。
   - 在左侧面板中的安全部分，选择个安全组选项。
2. 选择 Create Security Group（创建安全组）。
   - 为安全组提供名称（例如`datawranger-doc-route53-resolver-sg`) 和描述。
   - 选择之前步骤中使用的 VPC ID。
   - 创建允许从 VPC CIDR 块内通过 UDP 和 TCP 进行 DNS 的规则。



- 选择 Create Security Group（创建安全组）。注意安全组 ID因为添加了一条规则以允许流量到 VPC 终端节点安全组。

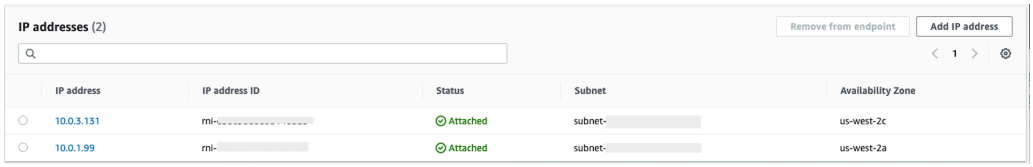
3. 导航到Route 53 菜单在您的Amazon控制台。
   - 在解析程序部分，选择入站端节点选项。
4. 选择创建入站端节点.
   - 提供终端节点名称。
   - 从该地区的 VPC下拉列表中，选择您在之前所有步骤中使用的 VPC ID。
   - 在此终端节点的安全组下拉列表中，从本部分的步骤 2 中选择安全组 ID。



   - 在IP 地址部分中，选择一个可用区，选择一个子网，然后将无线电选择器留下使用自动选择的 IP 地址为每个 IP 地址选择。

- 选择 Submit（提交）。

5. Select入站端节点在创建之后。
6. 创建入站终端节点后，请记下解析程序的两个 IP 地址。



SageMaker VPC 终端节点

本部分介绍如何为以下目的创建 VPC 终端节点：亚马逊 SageMaker Studio SageMaker 笔记本，SageMaker API， SageMaker 运行时和亚马逊 SageMaker 功能商店运行时。

创建应用于所有终端节点的安全组。

1. 导航到EC2 菜单中的Amazon控制台。
2. 在网络与安全部分，选择安全组选项。
3. 选择创建安全组。

4. 提供安全组名称和说明（例如：`datawrangler-doc-sagemaker-vpce-sg`）。稍后会添加一条规则以允许来自 HTTPS 的流量 SageMaker 到这个组中。

创建终端节点

1. 导航到VPC 菜单中的Amazon控制台。
2. Select终端节点选项。
3. 选择 Create Endpoint（创建端点）。
4. 通过在搜索字段中返回的子位置类型。
5. 从VPC下拉列表中，选择 Snowflake 所在的 VPCAmazon PrivateLink连接存在。
6. 在Subnets部分中，选择有权访问 Snowflake 的子网 PrivateLink 连接。
7. 离开启用 DNS 名称选中复选框。
8. 在个安全组部分中，选择在上一部分中创建的安全组。
9. 选择 Create Endpoint（创建端点）。

配置工作室和数据牧马人

本部分介绍如何配置 Studio 和 Data Wrangler。

1. 配置安全组。
   a. 导航到中的 Amazon EC2 菜单Amazon控制台。
   b. Select个安全组中的选项网络与安全部分。
   c. 选择 Create Security Group（创建安全组）。
   d. 为您的安全组提供名称和描述（例如：`datawrangler-doc-sagemaker-studio`）。
   e. 创建以下入站规则。
      - 与您为 Snowflake 配置的安全组的 HTTPS 连接 PrivateLink 在中创建的连接设置雪花 PrivateLink 集成Step.
      - 与您为 Snowflake 配置的安全组的 HTTP 连接 PrivateLink 在中创建的连接设置雪花 PrivateLink 集成步骤.
      - 您在第 2 步中创建的 DNS（端口 53）到 Route 53 解析程序入站端点安全组的 UDP 和 TCP为您的 VPC 配置 Route 53 解析器入站终端节点.
   f. 选择创建安全组右下角的按钮。
2. 配置 Studio。
   - 导航到 SageMaker 在菜单Amazon控制台。
   - 从左侧控制台中，选择SageMaker Studio选项。
   - 如果您没有配置任何域，开始使用已存在菜单。
   - Select标准设置选项来自开始使用菜单。
   - UNDER身份验证方法Select,AmazonIdentity and Access Management (IAM).
   - 从Permissions (权限)菜单中，您可以创建新角色或使用预先存在的角色，具体取决于您的使用案例。
     - 如果选择创建新角色，您将获得提供 S3 存储桶名称的选项，并为您生成策略。
     - 如果您已经创建了具有需要访问权限的 S3 存储桶的角色，请从下拉列表中选择角色。这个角色应该有`AmazonSageMakerFullAccess`附加到它的策略。
   - Select网络和存储用于配置 SageMaker 使用的 VPC、安全性和子网的下拉列表。
     - UNDERVPC中，选择您的 Snowflake 所在的 VPC PrivateLink 连接存在。
     - UNDER子网中，选择有权访问 Snowflake PrivateLink 连接的子网。
     - UNDERStudio 的网络访问Select,仅限 VPC.
     - UNDER安全组选择您在步骤 1 中创建的安全组。

- 选择 Submit（提交）。

3. 编辑 SageMaker 安全组。
   - 创建以下入站规则：
     - 端口 2049 到自动创建的入站和出站 NFS 安全组 SageMaker 在步骤 2 中（安全组名称包含 Studio 域 ID）。
     - 访问自己的所有 TCP 端口（仅适用于 VPC 的 SageMaker 需要）。
4. 编辑 VPC 终端节点安全组：
   - 导航到中的 Amazon EC2 菜单Amazon控制台。
   - 找到在上一步中创建的安全组。
   - 添加入站规则，允许来自步骤 1 中创建的安全组的 HTTPS 流量。
5. 创建用户配置文件。
   - 从SageMaker Studio 控制面板，选择添加用户.
   - 提供用户名称。
   - 对于执行角色，选择创建新角色或使用预先存在的角色。
     - 如果选择创建新角色，您可以选择提供 Amazon S3 存储桶名称，并为您生成策略。
     - 如果您已经创建了一个角色，具有访问您需要访问的 Amazon S3 存储桶的权限，请从下拉列表中选择角色。这个角色应该有`AmazonSageMakerFullAccess`附加到它的策略。
   - 选择 Submit（提交）。
6. 创建数据流（遵循上一节中概述的数据科学家指南）。
   - 添加 Snowflake 连接时，请输入值`privatelink-account-name`（来自的）设置雪花 PrivateLink 集成步骤）进入Snowflake 账户名称（字母数字）字段，而不是普通的 Snowflake 帐户名称。其他一切都保持不变。

导入的数据存储

> **Important**
>
> 我们强烈建议您遵循关注保护 Amazon S3 存储桶的最佳实践。安全最佳实践.

当您查询来自 Amazon Athena 或 Amazon Redshift 的数据时，查询的数据集将自动存储在 Amazon S3 中。数据存储在默认情况下 SageMaker 用于Amazon您正在使用 Studio 的区域。

默认 S3 存储桶具有以下命名约定：`sagemaker-`*`region`*`-`*`account number`*. 例如，如果您的账号是 111122223333，并且您在中使用 Studio（Studio）是：`us-east-1`，导入的数据集存储在`sagemaker-us-east-1-111122223333`。

Data Wrangler 流程取决于此 Amazon S3 数据集位置，因此在使用相关流程时，您不应在 Amazon S3 中修改此数据集。如果您确实修改了此 S3 位置，并且要继续使用数据流，则必须删除中的所有对象`trained_parameters`在你的 .flow 文件中。为此，请从 Studio 下载 .flow 文件，并为每个实例下载`trained_parameters`中，删除所有条目。完成后，`trained_parameters`应该是一个空的 JSON 对象：

```
"trained_parameters": {}
```

当您导出并使用数据流处理数据时，您导出的 .flow 文件会引用 Amazon S3 中的此数据集。请参阅以下部分了解更多信息。

Amazon Redshift 导入存储

Data Wrangler 在默认情况下将查询生成的数据集存储在 Picth 文件中 SageMaker S3 存储桶。

此文件存储在以下前缀（目录）下：redshift/*`uuid`*/data/，其中*`uuid`*是为每个查询创建的唯一标识符。

例如，如果您的默认存储桶是`sagemaker-us-east-1-111122223333`，从 Amazon Redshift 查询的单个数据集位于 s3: //sagemaker-us-east-1-111122223333/redshift/*uuid*/data/。

Amazon Athena 导入存储

当您查询 Athena 数据库并导入数据集时，Data Wrangler 会存储该数据集以及该数据集的子集，或者预览文件，在 Amazon S3 中。

通过选择导入的数据集导入数据集在 Amazon S3 中以 Parquet 格式存储。

选择时，预览文件将以 CSV 格式编写运行在 Athena 导入屏幕上，最多包含所查询数据集中的 100 行。

您查询的数据集位于前缀（目录）下：athena/*uuid*/data/，其中*uuid*是为每个查询创建的唯一标识符。

例如，如果您的默认存储桶是`sagemaker-us-east-1-111122223333`，从 Athena 查询的单个数据集位于`s3://sagemaker-us-east-1-111122223333`/athena/*uuid*/data/*# _data et.####/####*.

存储用于在 Data Wrangler 中预览数据框的数据集的子集存储在前缀：athena/。

创建和使用数据 Wrangler 流

使用 Amazon SageMaker 数据 Wrangler 流程，或数据流，以创建和修改数据准备管道。数据流将数据集、转换和分析连接起来，或者步骤，您可以创建并可用于定义管道。

实例

当您在亚马逊中创建数据 Wrangler 流程时 SageMaker Studio Wrangler 使用 Amazon EC2 实例在流程中运行分析和转换。默认情况下，Data Wrangler 使用 m5.4xlarge 实例。m5 实例是在计算和内存之间实现平衡的通用实例。您可以将 m5 实例用于各种计算工作负载。

Data Wrangler 还为您提供了使用 r5 实例的选项。r5 实例旨在提供处理内存中的大型数据集的快速性能。

我们建议您选择围绕工作负载进行最佳优化的实例。例如，r5.8xlarge 的价格可能比 m5.4xlarge 更高，但 r5.8xlarge 可能会针对您的工作负载进行更好的优化。借助更好的优化实例，您可以以更低的成本在更短的时间内运行数据流。

下表列出了可用于运行 Data Wrangler 流程的实例。

标准实例	vCPU	内存
ml.m5.4xlarge	16	64 GiB
ml.m5.8xlarge	32	128 GiB
ml.m5.16xlarge	64	256 GiB
ml.m5.24xlarge	96	384 GiB
r5.4xlarge	16	128 GiB
r5.8xlarge	32	256 GiB
r5.24xlarge	96	768 GiB

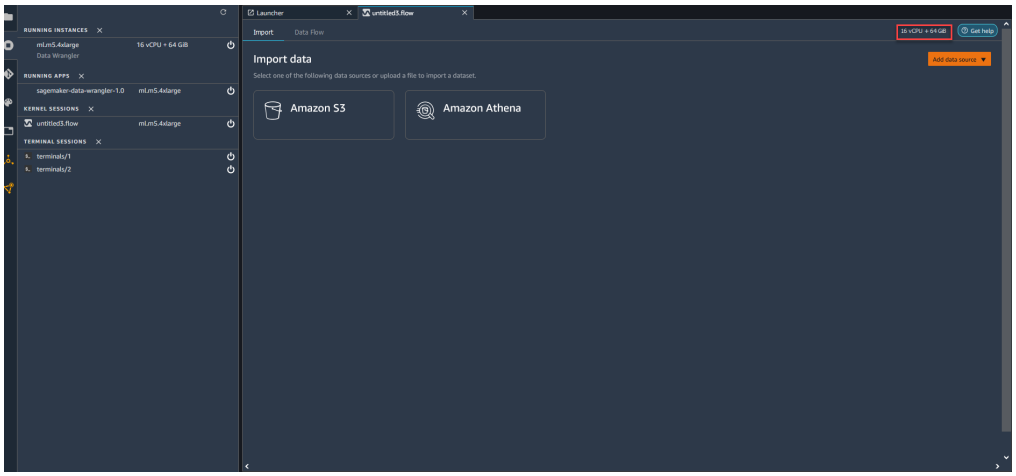
有关 r5 实例的更多信息，请参阅Amazon EC2 R5 实例. 有关 m5 实例的更多信息，请参阅Amazon EC2 M5 实例.

每个 Data Wrangler 流都有一个与之关联的 EC2 实例。您可能有多个流程与单个实例关联。

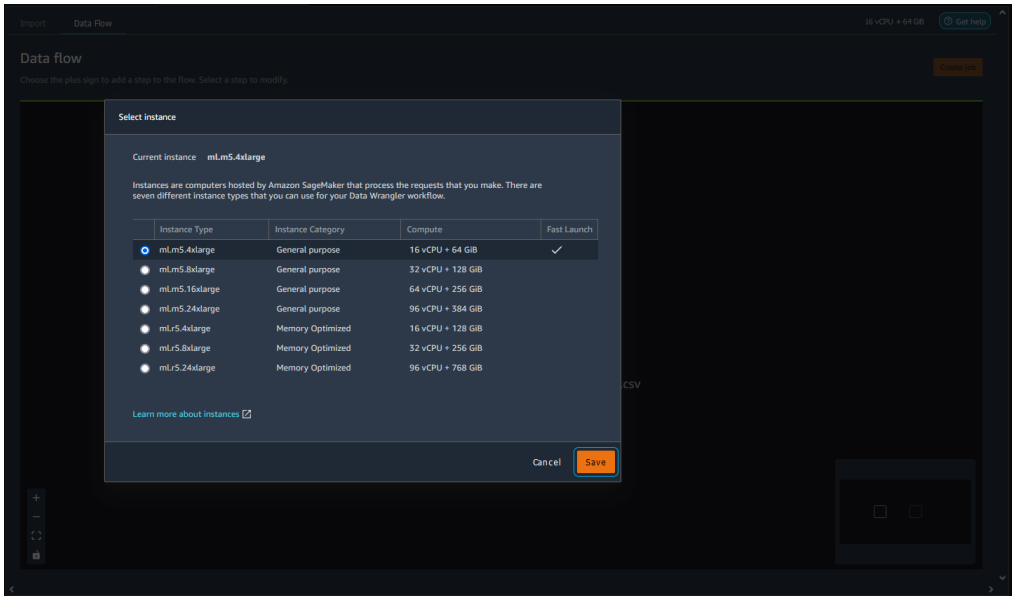
对于每个流程文件，您可以无缝切换实例类型。如果您切换实例类型，则用于运行流程的实例将继续运行。

要切换流程的实例类型，请执行以下操作。

1. 导航到当前正在使用的实例并选择它。下图显示了在哪里选择实例。



2. 选择要使用的实例类型。

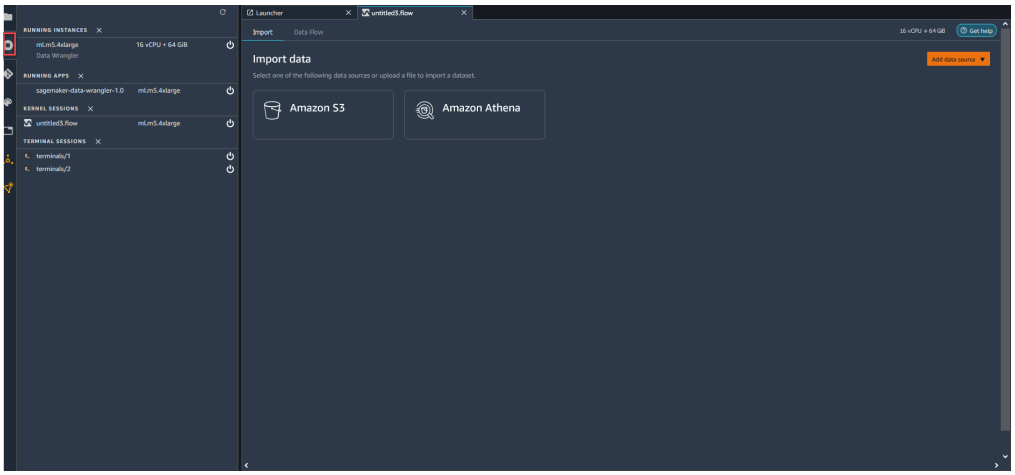


3. 选择 Save（保存）。

您需要为所有运行的实例付费。为避免产生额外费用，请关闭您没有手动使用的实例。要关闭正在运行的实例，请使用以下步骤。

关闭运行的实例。

1. 选择 UI 左侧的实例图标。下图显示了在哪里选择运行实例图标。



2. 选择关闭按钮 (位于要关闭的实例旁边)。

如果您关闭用于运行流程的实例，则无法暂时访问该流程。如果在打开运行之前关闭的实例的流程时出错，请等待大约五分钟，然后再尝试再次打开它。

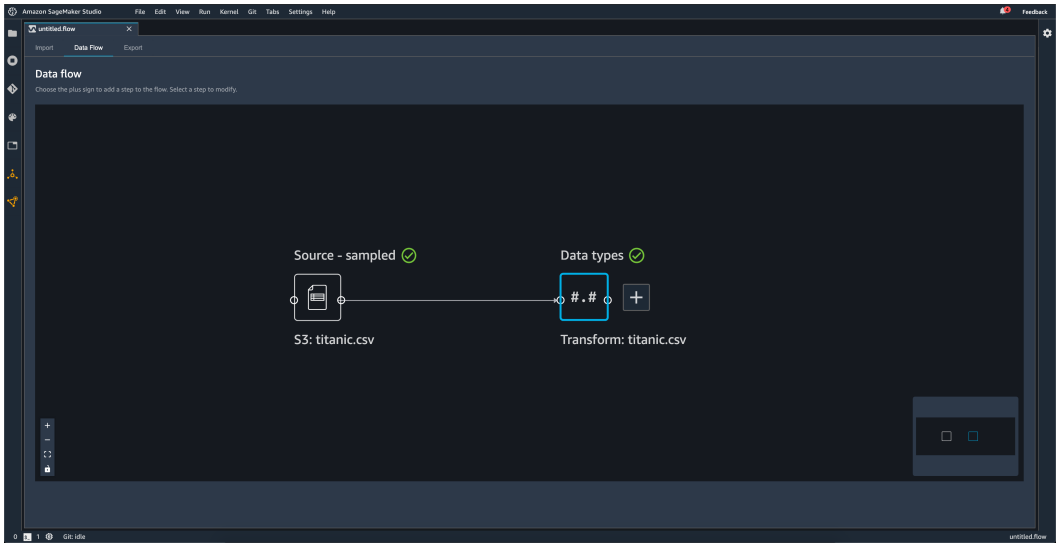
当您将数据流导出到亚马逊简单存储服务或亚马逊之类的位置 SageMaker 功能商店，Data Wrangler 运营亚马逊 SageMaker 处理作业。您可以使用以下实例之一进行处理作业。有关导出数据的更多信息，请参阅：Export (p. 664).

标准实例	vCPU	内存
ml.m5.4xlarge	16	64 GiB
ml.m5.12xlarge	48	192 GiB
ml.m5.24xlarge	96	384 GiB

有关使用可用实例类型的每小时成本的更多信息，请参阅SageMaker 定价.

数据流 UI

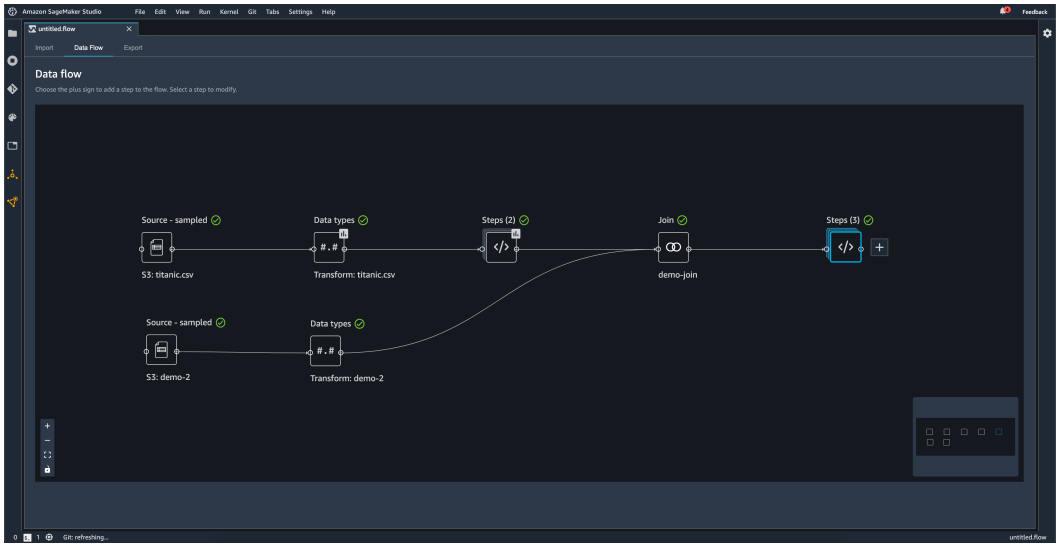
导入数据集时，原始数据集将显示在数据流中并命名为源. 如果您在导入数据时启用了采样，则此数据集将命名为源-采样. Data Wrangler 会自动推断数据集中每列的类型，并创建一个名为数据类型. 您可以选择此框架来更新推断的数据类型。上传单个数据集后，您将看到类似于下图所示的结果：



每次添加转换步骤时，都将创建一个新的数据框。当向同一数据集中添加多个转换步骤（不包括 Join 或 Concatenate）时，它们将堆叠起来。

加入和 Concatenate 创建包含新的已连接或连接数据集的独立步骤。

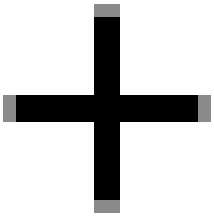
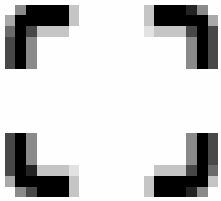
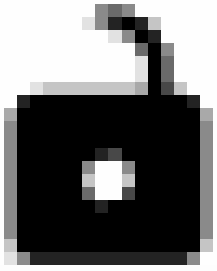
下图显示了一个数据流，其中包含两个数据集之间的连接以及两个步骤堆栈。第一个堆栈（步骤 (2)）向中推断的类型添加了两个转换数据类型数据集。这些区域有：下游堆栈或右侧的堆栈，将名为的连接所产生的转换添加到数据集演示加入.



数据流右下角的小灰色框提供了流程中堆栈和步骤的数量以及流程布局的概述。灰色框内的较亮框表示 UI 视图中的步骤。您可以使用此框查看数据流中不属于 UI 视图的部分。使用适合屏幕图标

(

) 将所有步骤和数据集填充到 UI 视图中。

左下角的导航栏包含可用于放大的图标（  ）和输出（ ），然后调整数据流的大小以适应屏幕（  ）。使用锁图标（  ）来锁定和解锁屏幕上每个步骤的位置。

向数据流添加一个步骤

Select+选择任何数据集或之前添加的步骤旁边，然后选择以下选项之一：

- 编辑数据类型（对于数据类型仅限步骤）：如果您还没有将任何转换添加到数据类型步骤，你可以选择编辑数据类型以更新导入数据集时 Data Wrangler 推断的数据类型。
- 添加转换：添加新的转换步骤。请参阅转换数据 (p. 623)以了解有关可以添加的数据转换的更多信息。
- 添加分析：添加分析。您可以使用此选项在数据流中的任何时候分析数据。将一个或多个分析添加到步骤时，一个分析图标（ ）出现在该步骤中。请参阅可视化 (p. 658)以了解有关可以添加的分析的更多信息。
- 加入：连接两个数据集并将生成的数据集添加到数据流中。要了解更多信息，请参阅 加入数据集 (p. 627)。
- 连接：连接两个数据集并将生成的数据集添加到数据流中。要了解更多信息，请参阅 连接数据集 (p. 628)。

从数据流中删除步骤

要删除步骤，请选择该步骤并选择Delete. 如果节点是具有单个输入的节点，则只删除所选步骤。删除具有单个输入的步骤并不会删除该步骤之后的步骤。如果要删除源、加入或连接节点的步骤，则它之后的所有步骤也将被删除。

要从步骤堆栈中删除步骤，请选择堆栈，然后选择要删除的步骤。

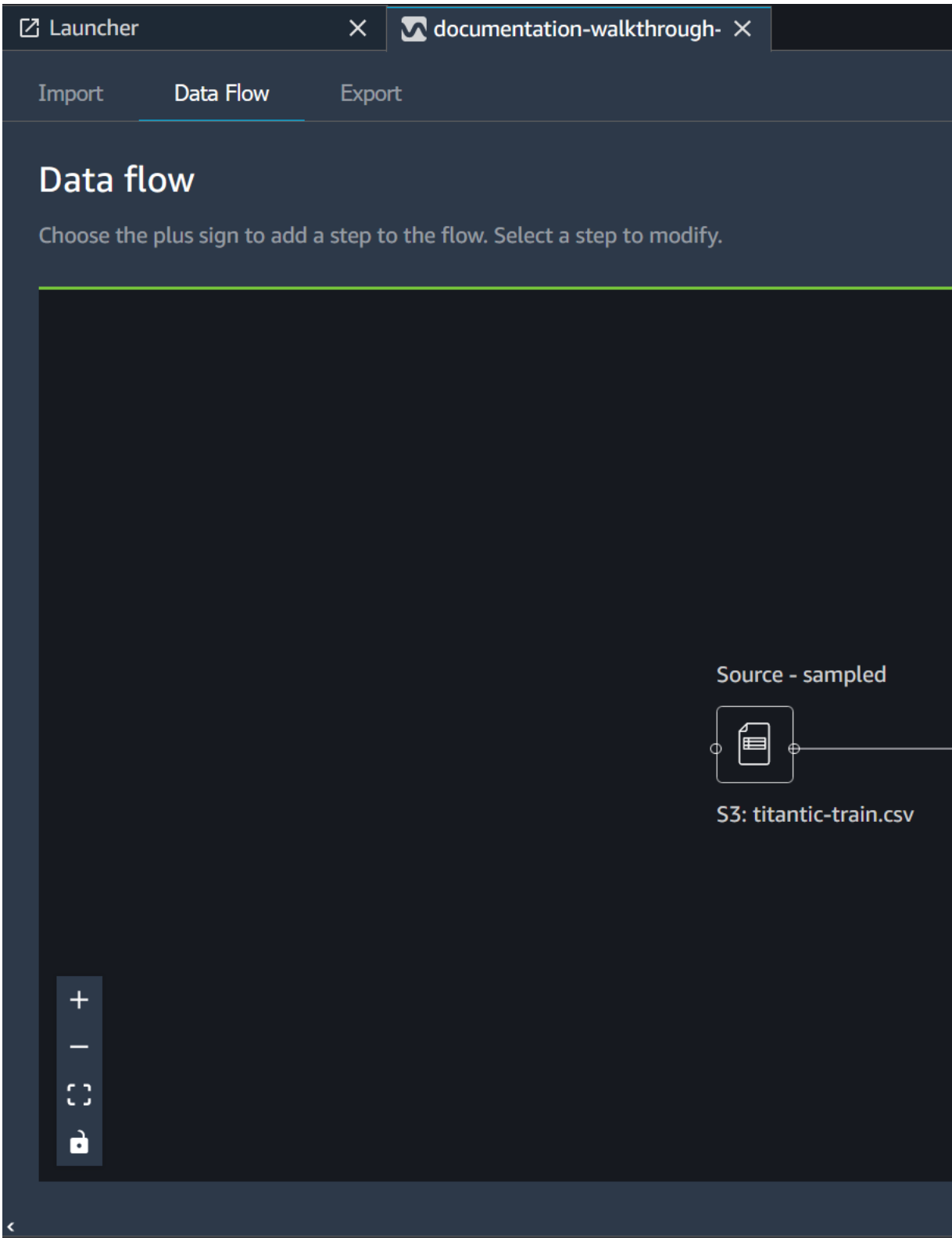
您可以使用以下过程之一删除步骤，而不删除下游步骤。

Delete a step in the Data Wrangler flow

您可以删除数据流中具有单个输入的节点的单个步骤。您无法删除源、加入和连接节点的单个步骤。

可以使用以下过程删除 Data Wrangler 流程中的一个步骤。

1. 选择包含要删除的步骤的步骤组。
2. 选择步骤旁边的图标。
3. 请选择 Delete（删除）。

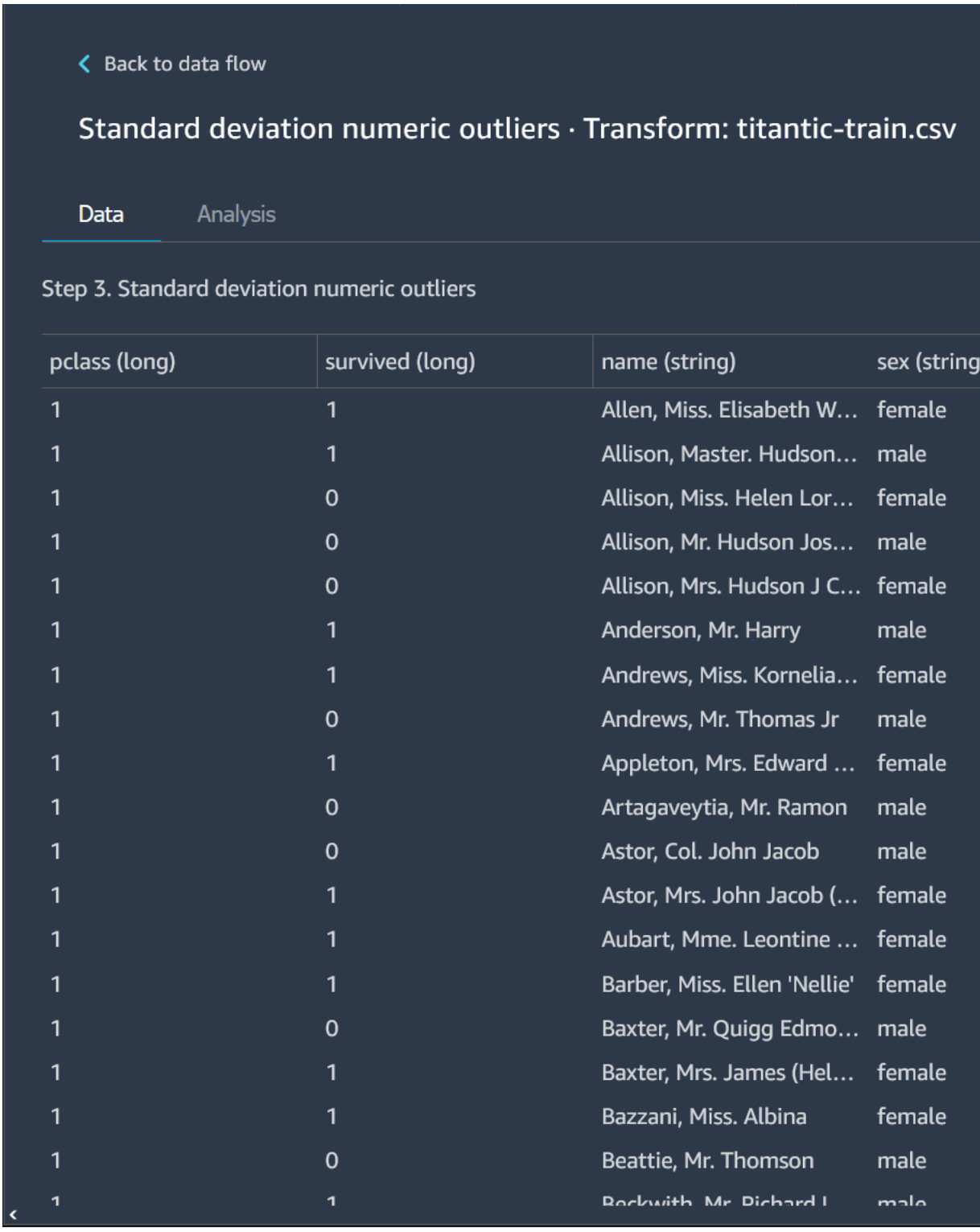

Launcher
documentation-walkthrough-
Import
Data Flow
Export
Data flow
Choose the plus sign to add a step to the flow. Select a step to modify.
Source - sampled
S3: titantic-train.csv

Delete a step in the table view

可以使用以下过程删除表视图中的步骤。

您可以删除数据流中具有单个输入的节点的单个步骤。您无法删除源、加入和连接节点的单个步骤。

1. 选择步骤并打开该步骤的表格视图。
2. 将光标移动到步骤之上。
3. 选择步骤旁边的图标。
4. 请选择 Delete（删除）。


Back to data flow
Standard deviation numeric outliers · Transform: titantic-train.csv
Data
Analysis
Step 3. Standard deviation numeric outliers
pclass (long) survived (long) name (string) sex (string
1 1 Allen, Miss. Elisabeth W... female
1 1 Allison, Master. Hudson... male
1 0 Allison, Miss. Helen Lor... female
1 0 Allison, Mr. Hudson Jos... male
1 0 Allison, Mrs. Hudson J C... female
1 1 Anderson, Mr. Harry male
1 1 Andrews, Miss. Kornelia... female
1 0 Andrews, Mr. Thomas Jr male
1 1 Appleton, Mrs. Edward ... female
1 0 Artagaveytia, Mr. Ramon male
1 0 Astor, Col. John Jacob male
1 1 Astor, Mrs. John Jacob (... female
1 1 Aubart, Mme. Leontine ... female
1 1 Barber, Miss. Ellen 'Nellie' female
1 0 Baxter, Mr. Quigg Edmo... male
1 1 Baxter, Mrs. James (Hel... female
1 1 Bazzani, Miss. Albina female
1 0 Beattie, Mr. Thomson male

编辑数据 Wrangler 流程中的步骤

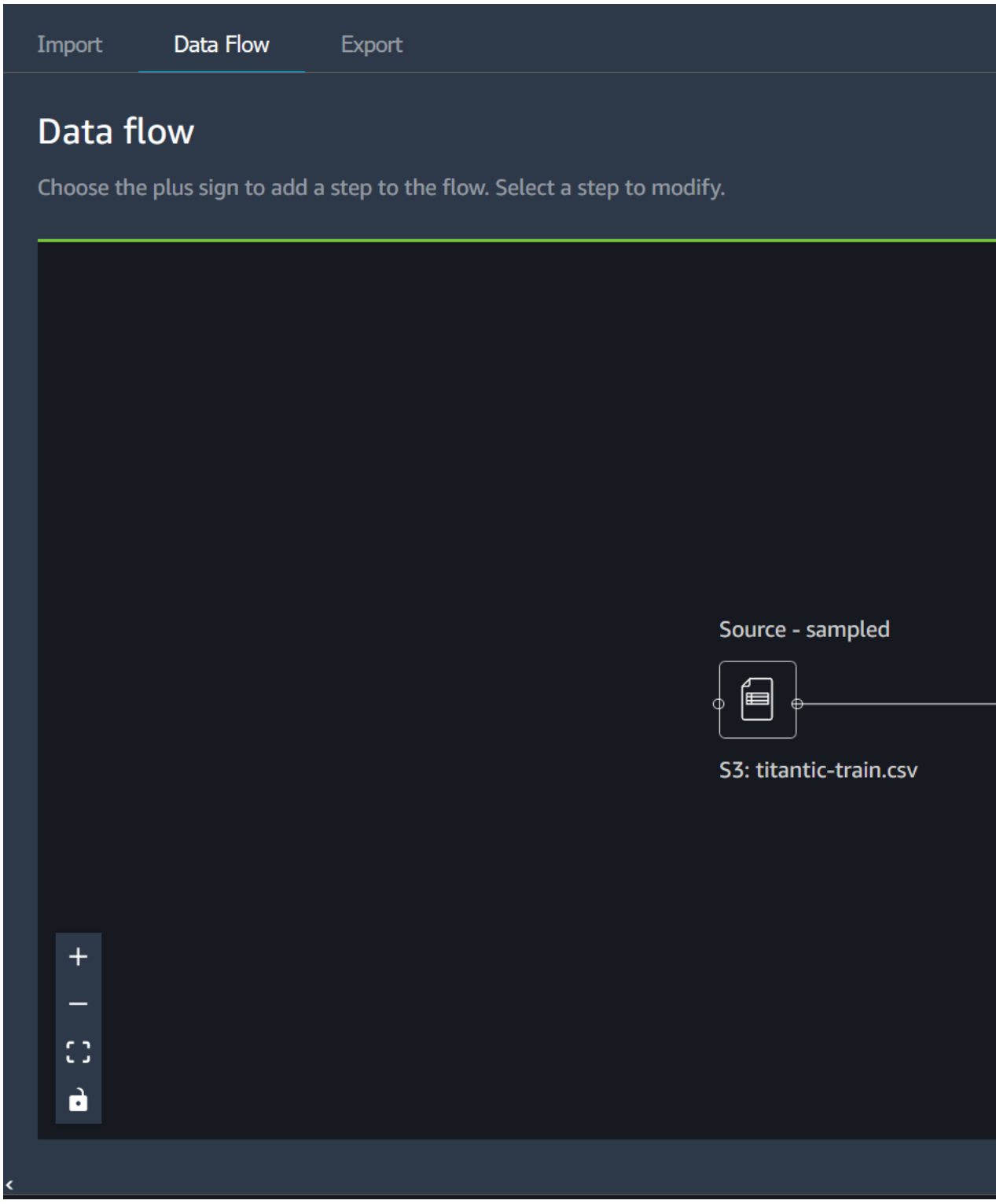
您可以编辑在 Data Wrangler 流程中添加的每个步骤。通过编辑步骤，您可以更改列的转换或数据类型。您可以编辑步骤以进行更改，从而使您能够执行更好的分析。

你可以通过多种方法来编辑步骤。一些例子包括更改估算法或更改将值视为异常值的阈值。

可以使用以下过程编辑步骤。

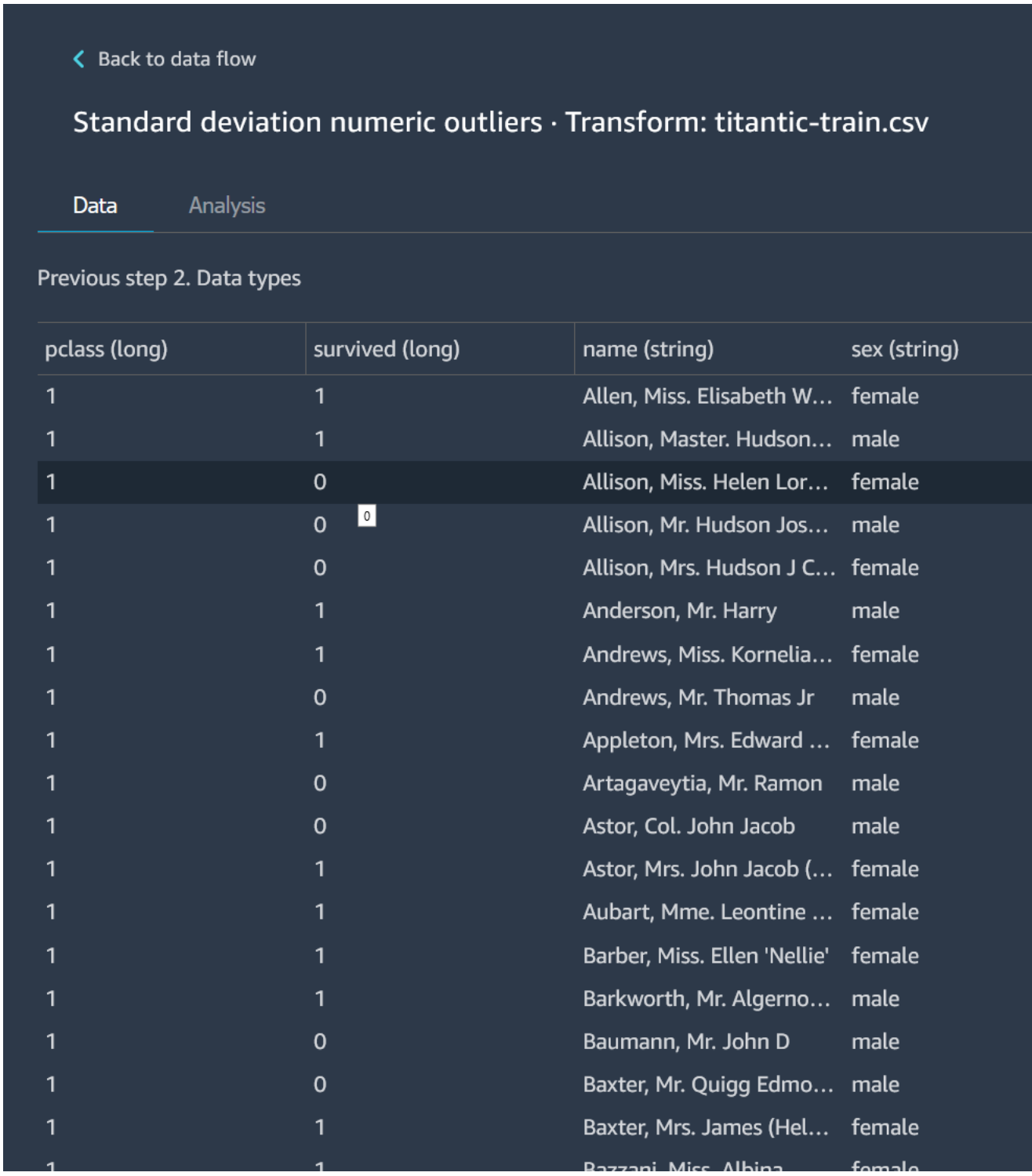
要编辑步骤，请执行以下操作。

1. 在 Data Wrangler 流程中选择一个步骤以打开表格视图。



2. 在数据流中选择一个步骤。
3. 编辑该步骤。

以下是编辑步骤的示例。



获取有关数据和数据质量的见解

使用数据质量和见解报告以便对已导入数据 Wrangler 的数据进行分析。我们建议您在导入数据集后创建报告。您可以使用报告来帮助清理和处理数据。它为您提供了缺失值的数量和异常值数等信息。如果您的数据存在问题，例如目标泄漏或不平衡，见解报告可以引起您注意这些问题。

Note

如果您对导入的数据进行了采样，Data Wrangler 将根据抽样数据创建报告。有关关闭采样的信息，请参阅导入 (p. 566).

以下主题介绍了报告的部分：

主题

- 摘要 (p. 612)
- “目标” 栏 (p. 614)
- 快速模型 (p. 617)
- 功能摘要 (p. 619)
- 示例 (p. 621)
- 定义 (p. 622)

您可以下载报告，也可以在线查看。如需下载报告，请选择屏幕右上角的下载按钮。下图显示了按钮。

摘要

洞察报告对数据进行了简要摘要，其中包括缺少值、无效值、要素类型、异常值计数等一般信息。它还可能包括高严重性警告，这些警告指出数据可能存在问题。我们建议您调查警告。

以下是报告摘要示例。

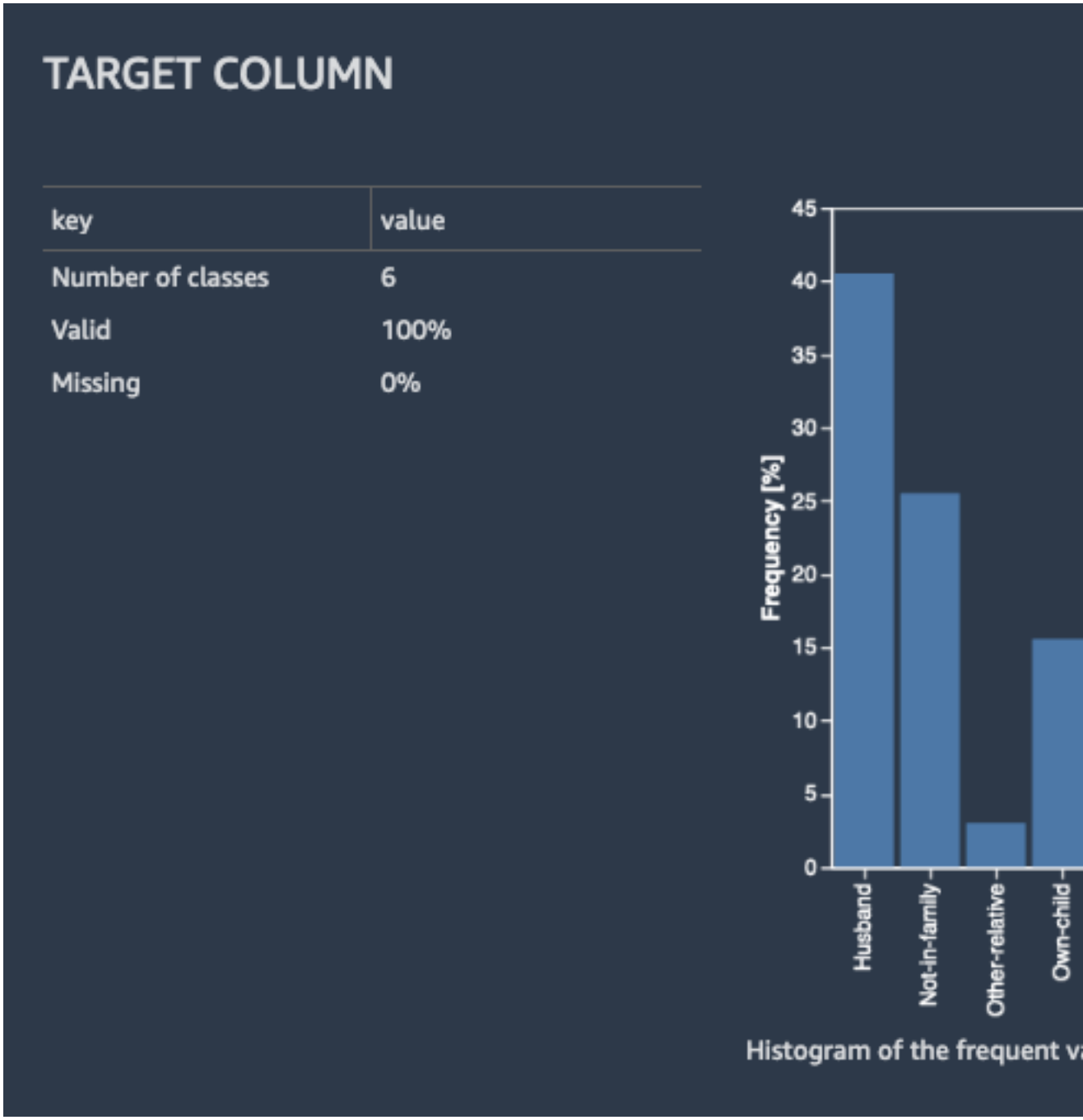
SUMMARY
Dataset statistics
Key
Value
Number of features
13
Number of rows
8553
Missing
0%
Valid
100%
Duplicate rows
4.63%
Feature type
numeric
categorical
text
datetime
binary
vector
None
High Priority Warnings
2 high severity warnings were detected. See the list below.
Skewed target
High
The target column is skewed and contains outliers. Because the outliers induce
prediction quality for the non-outlier samples. In case you are interested in pre
there is no need for further action. However, if extreme values are not the point
under Handle outliers.
Target leakage
High
The feature hoa_(BRL) predicts the target extremely well on it's own. A feature
prediction. For example, a duplicate of the target column in the dataset can res
Alternatively, if the machine learning task is "easy", then a single feature can ha
anything further. However, if you think there's target leakage, we recommende

“目标” 栏

当您创建数据质量和见解报告时，Data Wrangler 为您提供了选择目标列的选项。目标专栏是你试图预测的一列。当您选择目标列时，Data Wrangler 会自动创建目标列分析。它还按照其预测能力的顺序对这些要素进行排名。选择目标列时，必须指定是试图解决回归还是分类问题。

对于分类，Data Wrangler 显示了一个表格和最常见类的直方图。课是一个类别。它还显示目标值缺失或无效的观测值或行。

下图显示了分类问题的目标列分析示例。

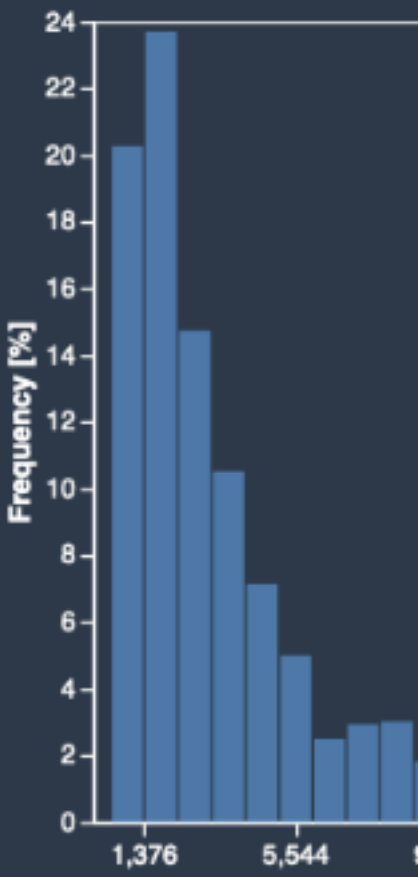


对于回归，Data Wrangler 将显示目标列中所有值的直方图。它还会显示具有缺失、无效或异常值目标值的观测值或行。

下图显示了回归问题的目标列分析示例。

TARGET COLUMN

key	value
Valid	100%
Missing	0%
Outliers	0.103%
Min	450
Max	4.5e+04
Mean	3.9e+03
Median	2.66e+03
Skew	1.84
Kurtosis	4.62
Number of unique	1195



Histogram of the target colum
average.

See below several samples with outlier target values.

city	area	rooms	bathroom	parking spaces	floor	animal	furni
São Paulo	700	4	7	8	-	acept	not f
São Paulo	350	3	3	3	-	acept	not f
São Paulo	486	8	4	6	-	acept	not f
São Paulo	80	2	1	1	1	acept	not f
São Paulo	900	3	4	8	-	acept	not f

快速模型

这些区域有：快速模型提供了对您根据数据训练的模型的预期预测质量的估计值。

Data Wrangler 将您的数据拆分为训练和验证折叠。它使用 80% 的样本进行训练，20% 的值用于验证。对于分类，样品被分层分割。对于分层拆分，每个数据分区的标签比例相同。对于分类问题，训练和分类折叠之间的标签比例相同非常重要。Data Wrangler 使用默认超参数训练 XGBoost 模型。它对验证数据应用提前停止并执行最少的功能预处理。

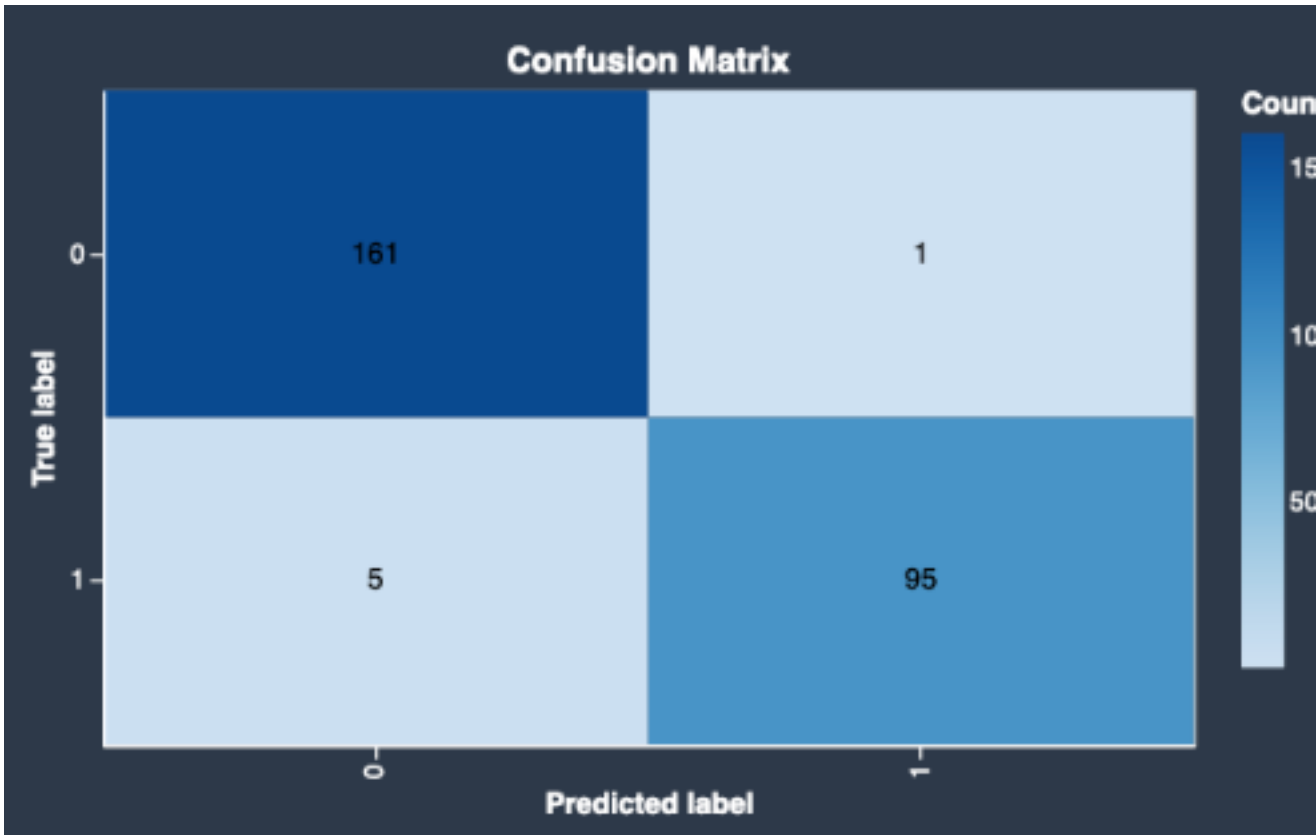
对于分类模型，Data Wrangler 会返回模型摘要和混淆矩阵。

以下是分类模型摘要示例。如需了解有关返回的信息的详细信息，请参阅定义 (p. 622).

Metric	Validation scores	Train scores
Accuracy	0.977	0.992
Balanced accuracy	0.972	0.99
ROC-AUC	0.995	1
F1	0.969	0.99
Precision	0.99	0.997
Recall	0.95	0.983

class	precision	recall	f1-sco
0	0.9698795180722891	0.9938271604938271	0.981
1	0.9895833333333334	0.95	0.969

以下是快速模型返回的混淆矩阵示例。



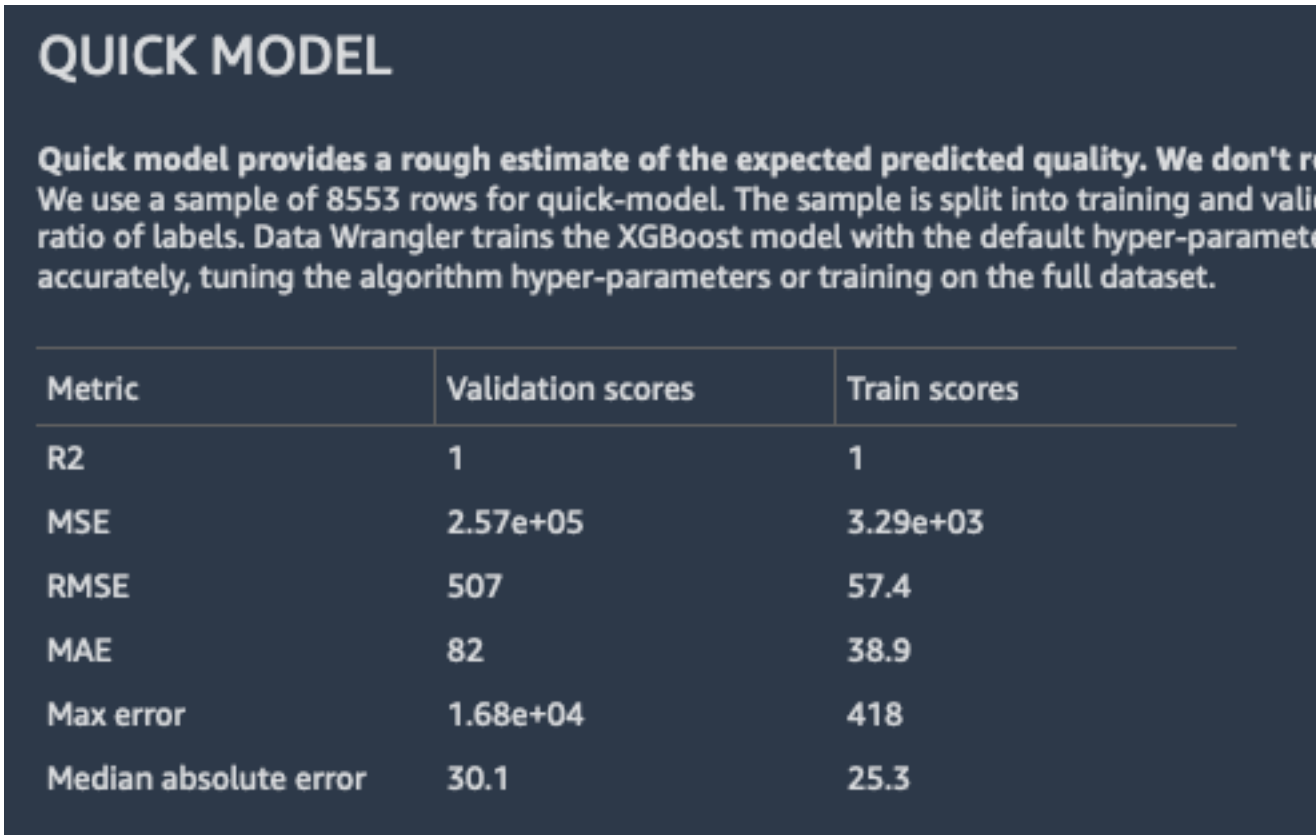
混淆矩阵提供以下信息：

- 预测的标签与真实标签匹配的次数。
- 预测的标签与真实标签不匹配的次数。

真正的标签代表了数据中的实际观察。例如，如果您使用模型来检测欺诈性交易，那么真实的标签代表实际上是欺诈性或非欺诈性的交易。预测的标签表示模型为数据分配的标签。

你可以使用混淆矩阵来查看模型对存在或不存在条件的预测程度如何。如果你预测欺诈性交易，你可以使用混淆矩阵来了解模型的敏感性和特殊性。敏感性是指模型检测欺诈性交易的能力。特殊性是指模型避免将非欺诈性交易检测为欺诈性交易的能力。

下面是回归问题快速模型输出示例。



QUICK MODEL

Quick model provides a rough estimate of the expected predicted quality. We don't r
We use a sample of 8553 rows for quick-model. The sample is split into training and vali
ratio of labels. Data Wrangler trains the XGBoost model with the default hyper-paramet
accurately, tuning the algorithm hyper-parameters or training on the full dataset.

Metric	Validation scores	Train scores
R2	1	1
MSE	2.57e+05	3.29e+03
RMSE	507	57.4
MAE	82	38.9
Max error	1.68e+04	418
Median absolute error	30.1	25.3

功能摘要

当您指定目标列时，Data Wrangler 会按照其预测能力对要素进行排序。预测能力是在以分层方式拆分为 80%/20% 的训练和验证折叠后，是根据数据来衡量的。Data Wrangler 针对训练组上的每个功能单独拟合一个模型。它应用最小的要素预处理并衡量验证数据的预测性能。

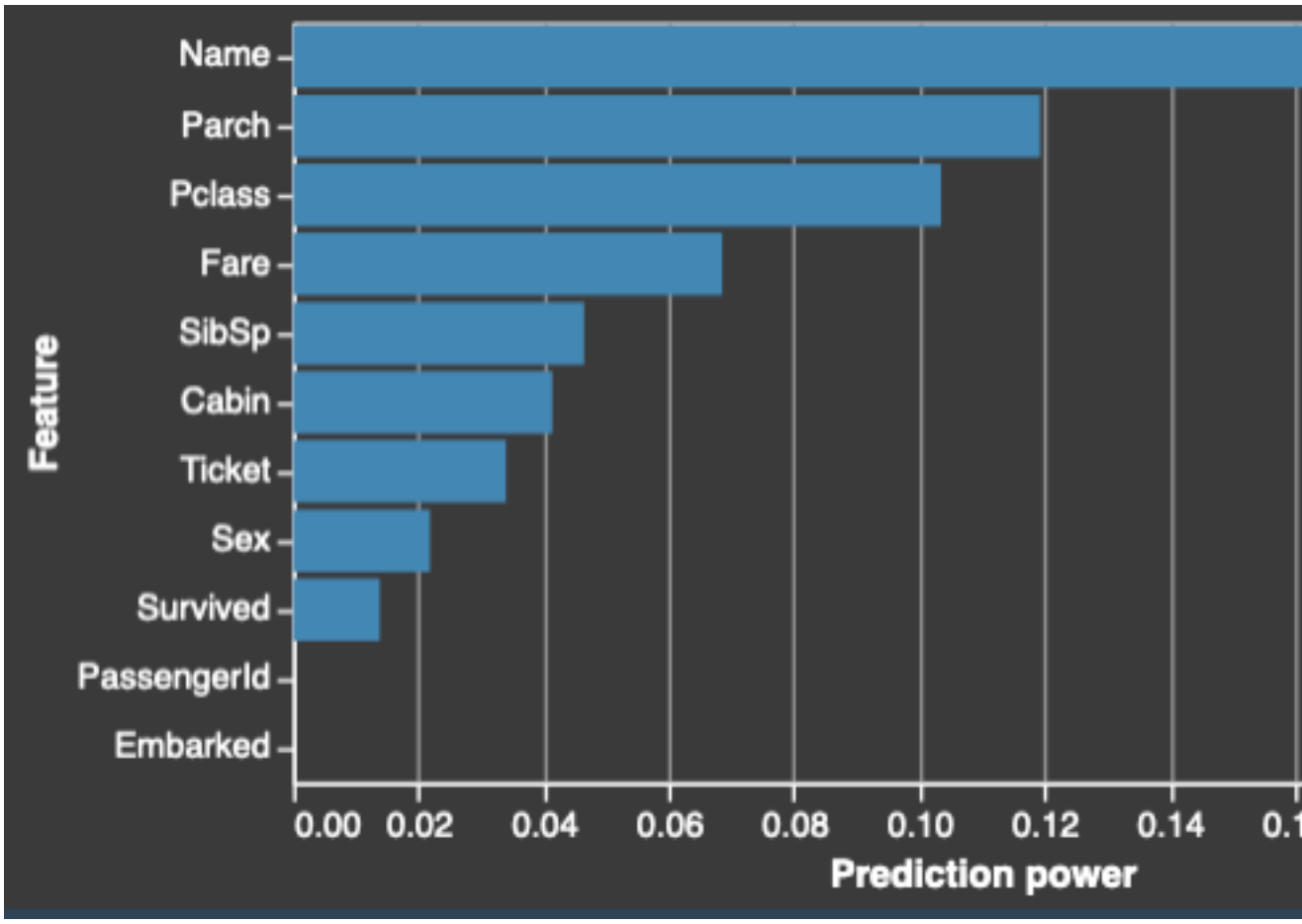
它将分数标准化为范围为 [0,1]。更高的预测分数表示对于自己预测目标更有用的列。分数较低表示不能预测目标列的列。

在与其他列同时使用时，本身并非预测性的列具有预测性，这种情况很少见。您可以放心地使用预测分数来确定数据集中的要素是否具有预测性。

分数较低通常表示该功能是多余的。分数为 1 意味着完美的预测能力，这通常表示目标泄漏。当数据集包含在预测时不可用的列时，通常会发生目标泄漏。例如，它可能是目标列的副本。

以下是显示每个要素的预测值的表格和直方图的示例。

Feature	Prediction power	Type	Valid	Missing	C
Name	0.274276	text	100.0%	0.0%	
Pclass	0.154638	numeric	100.0%	0.0%	
SibSp	0.141675	numeric	100.0%	0.0%	
Parch	0.127353	numeric	100.0%	0.0%	
Cabin	0.112283	text	25.91%	74.09%	
Ticket	0.0869433	numeric	72.97%	0.0%	
Fare	0.0625847	numeric	100.0%	0.0%	
Embarked	0.00600914	categorical	99.72%	0.28%	
Survived	0.00434197	binary	100.0%	0.0%	
PassengerId	0	numeric	100.0%	0.0%	
Sex	0	binary	100.0%	0.0%	



示例

Data Wrangler 会提供有关您的样本是否异常或数据集中是否有重复的信息。

Data Wrangler 使用隔离林算法检测异常样本。隔离林将异常分数与数据集的每个样本（行）关联起来。低异常分数表示样本异常。高分与非异常样本有关。具有负异常分数的样本通常被视为异常，异常分数为正的样本被视为非异常。

当你查看可能异常的样本时，我们建议你注意异常值。例如，您可能有异常值，这些值可能是由于收集和处理数据的错误而导致的。以下是根据 Data Wrangler 实施隔离林算法的最异常样本的示例。我们建议您在检查异常样本时使用领域知识和业务逻辑。

Data Wrangler 会检测重复行并计算数据中重复行的比率。某些数据源可能包含有效的重复项。其他数据源可能存在重复项，指出数据收集方面的问题。由于数据收集错误而导致的重复样本可能会干扰依赖将数据拆分为独立训练和验证折叠的机器学习过程。

以下是见解报告中可能受到重复样本影响的元素：

- 快速模型
- 预测功率估计
- 自动超级参数调整

您可以使用从数据集中删除重复的样本删除重复项转换管理行. Data Wrangler 向您展示最频繁的重复行。

定义

以下是数据见解报告中使用的技术术语的定义。

Feature types

以下是每种要素类型的定义：

- 数字 — 数字值，可以是浮点数或整数。例如：年龄、收入。机器学习模型假定数字值是有序的，并定义了它们的距离。例如，3 接近 4 而不是 10，3 < 4 < 10。
- 类别 — 列条目属于一组唯一值，通常比列中的条目数小得多。例如，长度为 100 的列，其中包含唯一值 “狗”、“猫” 和 “鼠标”。这些值可以是数字，如 “马”、“房子”、8、“爱” 和 3.1 两者的组合的文本都是有效值，可以在同一类别列中找到。与数字要素相反，机器学习模型不假定分类要素值的顺序或距离（即使所有值都是数字）。
- 二进制 — 分类要素的一种特殊情况，其中唯一值集的基数为 2。
- 文本 — 文本列包含许多非数字唯一值。在极端情况下，该列的所有元素都是唯一的。在极端情况下，没有两个条目是相同的。
- 日期时间 — 包含有关日期或时间的信息的列。它可以包含有关于日期和时间的信息。

Feature statistics

以下是每个要素统计信息的定义：

- 预测力量 — 衡量该列在预测目标方面的用处。
- 异常值（在数字列中）— Data Wrangler 使用对异常值强大的两个统计数据检测异常值：中位值和稳健标准差 (RSTD)。RSTD 是通过将要素值裁剪到范围 [5 百分位数，95 百分位数] 并计算剪切向量的标准差来得出的。所有大于中位数 + 5 * RSTD 或小于中位数的所有值-5 * RSTD 被视为异常值。
- 倾斜（在数字列中）— 斜度测量分布的对称性，定义为分布的第三个时刻除以标准差的第三个幂。正态分布或任何其他对称分布的偏度为零。正值表示分布的右尾比左尾部长。负值表示分布的左尾比右尾长。作为经验法则，当倾斜的绝对值大于 3 时，分布被认为是偏斜的。
- Kurtosis（在数字列中）— 皮尔逊的峰度衡量了分布尾部的沉重程度。它被定义为分布的第四个时刻除以第二个时刻的平方。正态分布的峰度为 3。Kurtosis 值低于 3 意味着分布集中在平均值周围，尾部比正态分布的尾部要轻。Kurtosis 值高于 3 意味着尾部或异常值较重。
- 缺少值 — 类 Null 的对象、空字符串和仅由空格组成的字符串被视为缺失。
- 数值要素或回归目标的有效值 — 您可以转换为有限浮点数的所有值均有效。缺少的值无效。
- 分类、二进制、文本要素或分类目标的有效值 — 所有未丢失的值都是有效的。
- 日期时间功能 — 您可以转换为 datetime 对象的所有值都有效。缺少的值无效
- 无效值 — 缺少或无法正确转换的值。例如，在数字列中，不能转换字符串 “六” 或空值。

Quick model metrics for regression

以下是快速模型指标的定义：

- R2 或确定系数）— R2 是模型预测的目标变化的比例。R2 在 [-infty, 1] 的范围内。1 是完美预测目标的模型的分数，0 是总是预测目标均值的微不足道模型的分数。
- MSE 或均方误差 — MSE 处于 [0, inty] 范围内。0 是完美预测目标的模型的分数。
- MAE 或平均绝对误差 — MAE 在 [0, infty] 范围内，其中 0 是完美预测目标的模型的分数。
- RMSE 或均方根误差 — RMSE 处于 [0, inty] 范围内，其中 0 是完美预测目标的模型的分数。
- 最大错误 — 错误与数据集相比的最大绝对值。最大误差在 [0, infty] 范围内。0 是完美预测目标的模型的分数。

- 中位数绝对误差 — 中位数绝对误差在 [0, infty] 范围内。0 是完美预测目标的模型的分数。

Quick model metrics for classification

以下是快速模型指标的定义：

- 准确性 — 准确预测的样本比率。准确度在 [0, 1] 范围内。0 是预测所有样本错误的模型的分数，1 是完美模型的分数。
- 平衡准确度 — 调整班级权重以平衡数据时准确预测的样本比率。所有课程都得到同样的重视，无论其频率如何。平衡精度在 [0, 1] 范围内。0 是预测所有样本错误的模型的分数。1 是完美模型的分数。
- AUC（二进制分类）— 受试者操作特征曲线下的面积。AUC 在 [0, 1] 范围内，随机模型返回 0.5 分，完美模型返回 1 分。
- AUC (OVR) — 对于多类分类，接收器操作特征曲线下的面积使用一个与休息单独计算每个标签。数据 Wrangler 报告了面积的平均值。AUC 在 [0, 1] 范围内，随机模型返回 0.5 分，完美模型返回 1 分。
- 精度 — 精度是针对特定类别定义的。精度是模型归类为该类的所有实例中真正阳性的一部分。精度在 [0, 1] 范围内。1 是模型的分数，对于班级没有误报。对于二进制分类，Data Wrangler 报告了积极类别的精确度。
- 召回 — 召回是针对特定课程定义的。召回是成功检索的相关类实例的一小部分。召回在 [0, 1] 范围内。1 是正确分类该班级的所有实例的模型的分数。对于二进制分类，Data Wrangler 报告了正面阶级的召回情况。
- F1 — F1 是针对特定类定义的。这是精度和调用率的调和平均数。F1 在范围为 [0, 1]。1 是完美模型的分数。对于二进制分类，Data Wrangler 报告具有正值的类的 F1。

转换数据

亚马逊 SageMaker Data Wrangler 提供了许多机器学习数据转换，以简化数据的清理、转换和特色化。添加转换时，它会为数据流添加一个步骤。您添加的每个变换都会修改数据集并生成新的数据框。所有后续变换都适用于生成的数据框。

Data Wrangler 包括内置转换，您可以使用它在无需任何代码的情况下转换列。您还可以使用 PySpark、Python（用户定义函数）、熊猫和添加自定义转换 PySpark SQL。有些转换可以运行，而其他转换则在数据集中创建一个新的输出列。

您可以一次将转换应用于多个列。例如，您可以在一个步骤中删除多个列。

您只能将流程数字应用于单个列并处理缺失的变换。

使用此页可以了解有关这些内置和自定义转换的更多信息。

转换界面

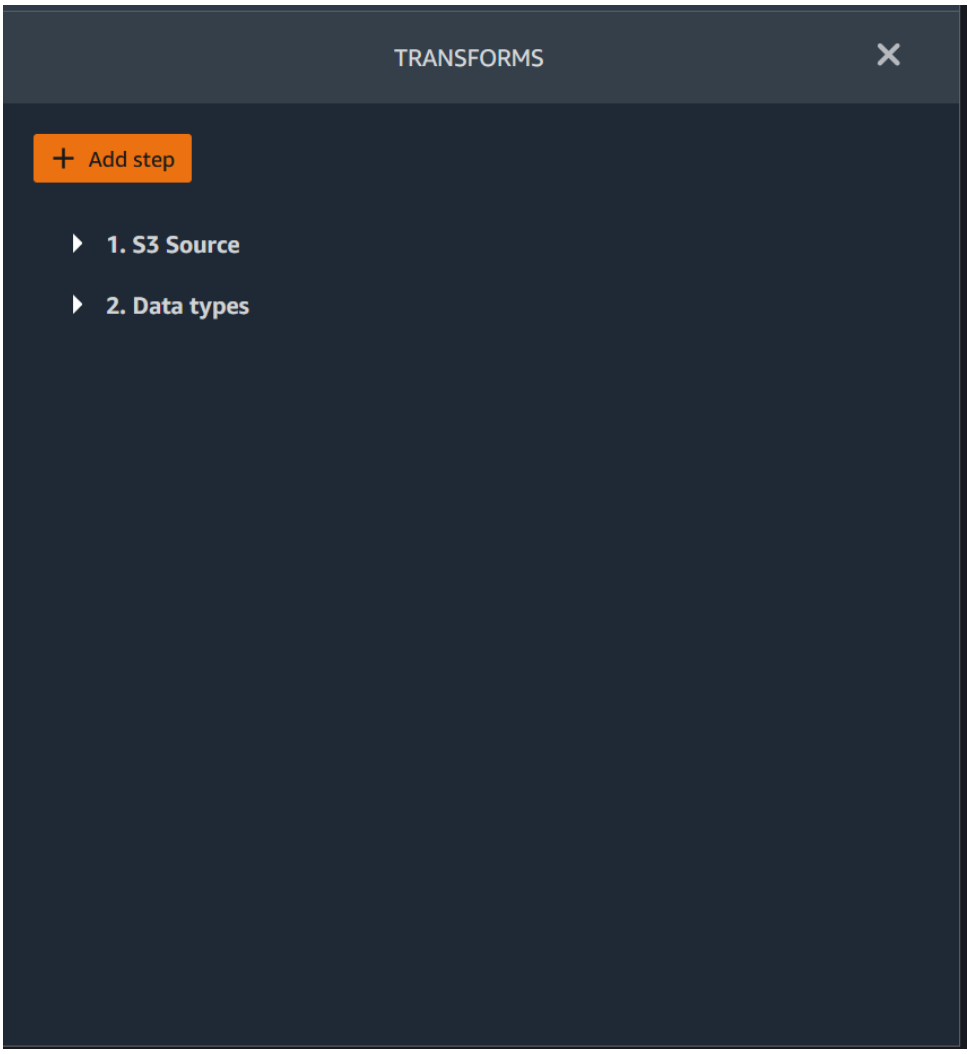
大多数内置转换位于准备Data Wrangler UI 的选项卡。通过数据流视图访问 “连接” 和 “连接” 转换。使用下表预览这两个视图。

Transform

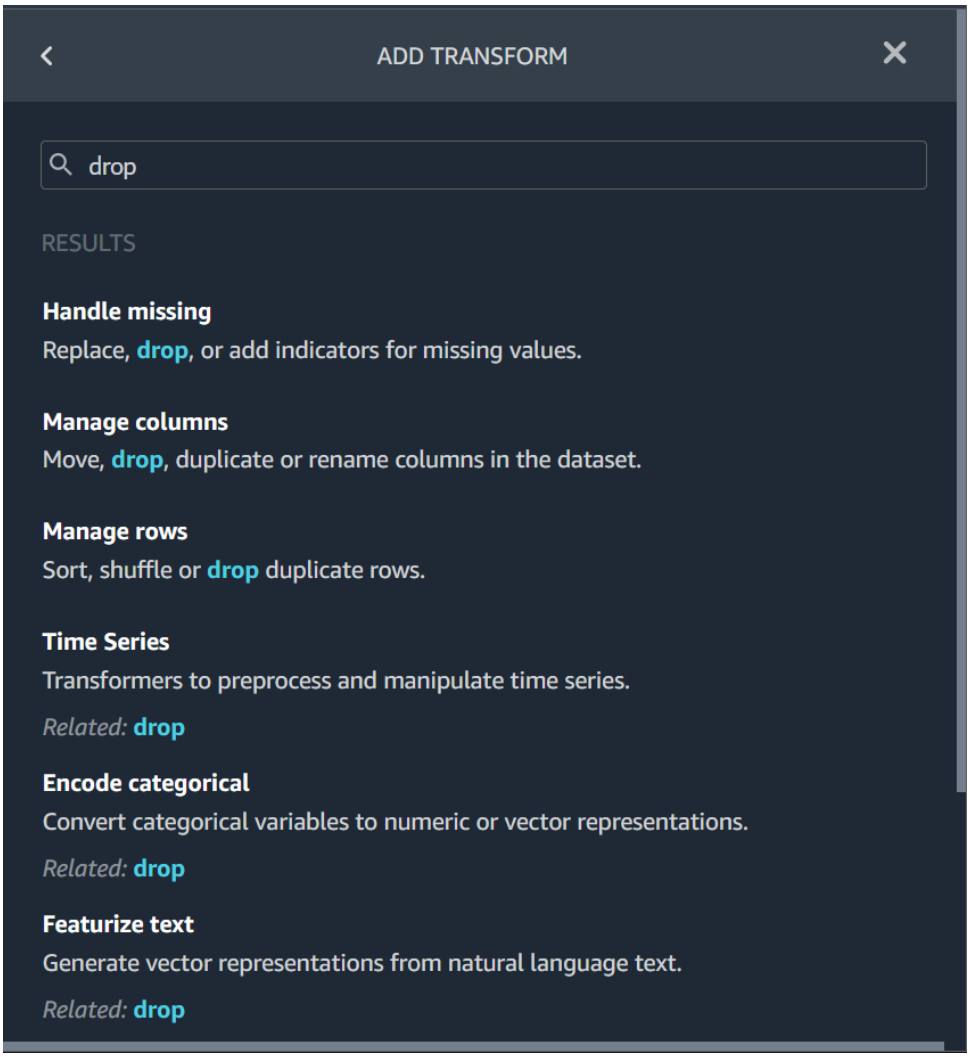
您可以向数据流中的任何步骤添加转换。要向数据流中添加转换，请按照以下过程操作。

要向数据流添加步骤，请执行以下操作。

1. 选择+在数据流中的步骤旁边。
2. 选择添加转换.
3. 选择添加步骤.

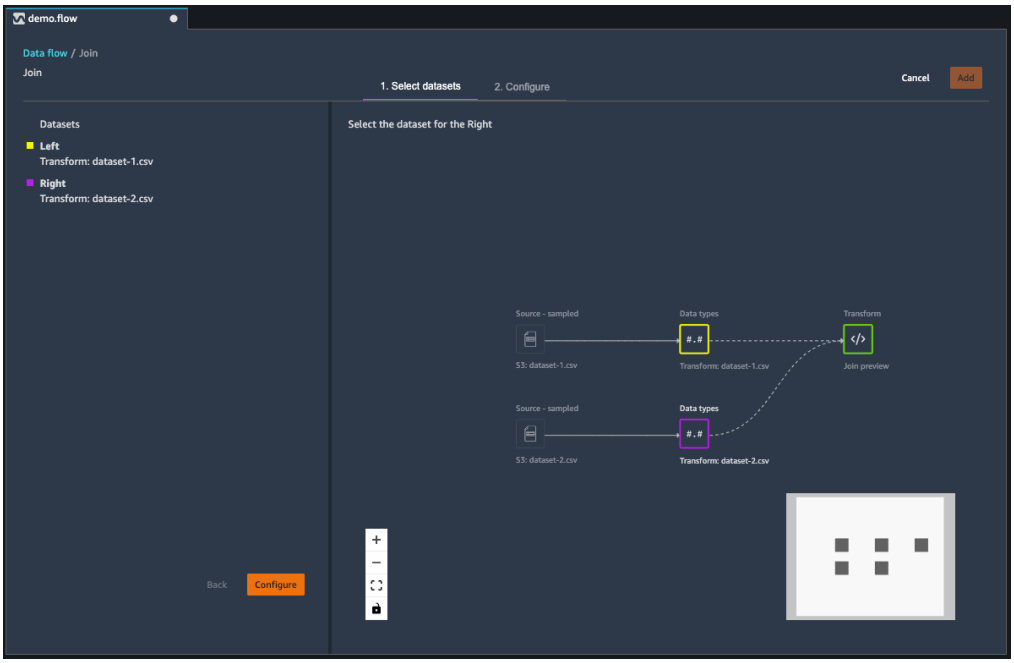


4. 选择一个转换。下图显示了可用转换。
5. （可选）您可以搜索要使用的转换。Data Wrangler 在结果中突出显示查询。

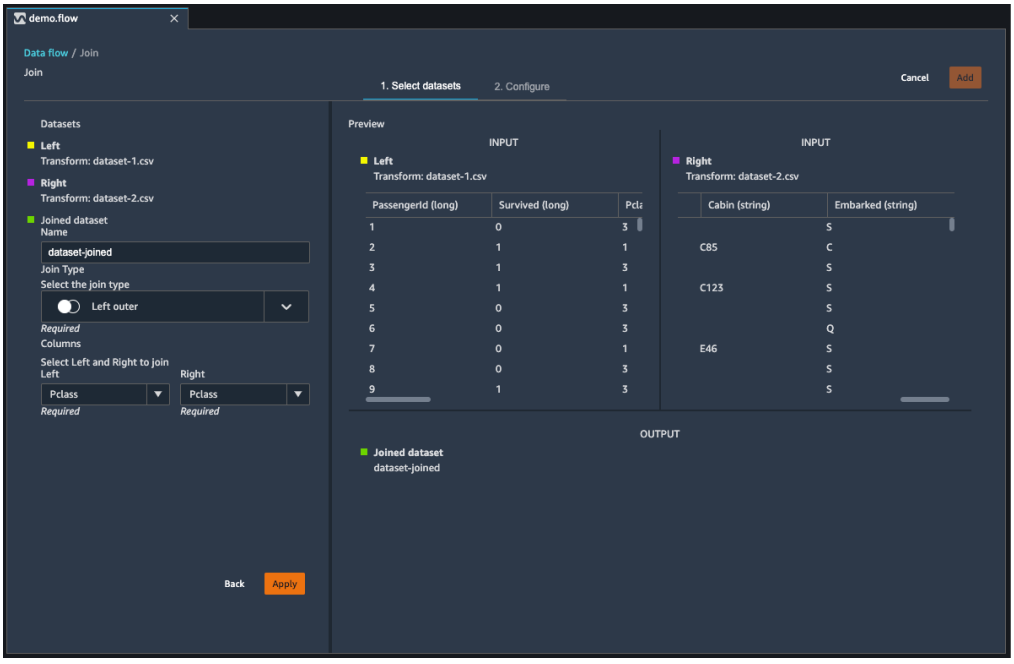


Join View

要连接两个数据集，请选择数据流中的第一个数据集，然后选择加入. 当您选择加入，您会看到类似于下图所示的结果。左侧和右侧的数据集显示在左侧面板中。主面板显示您的数据流，并添加了新加入的数据集。



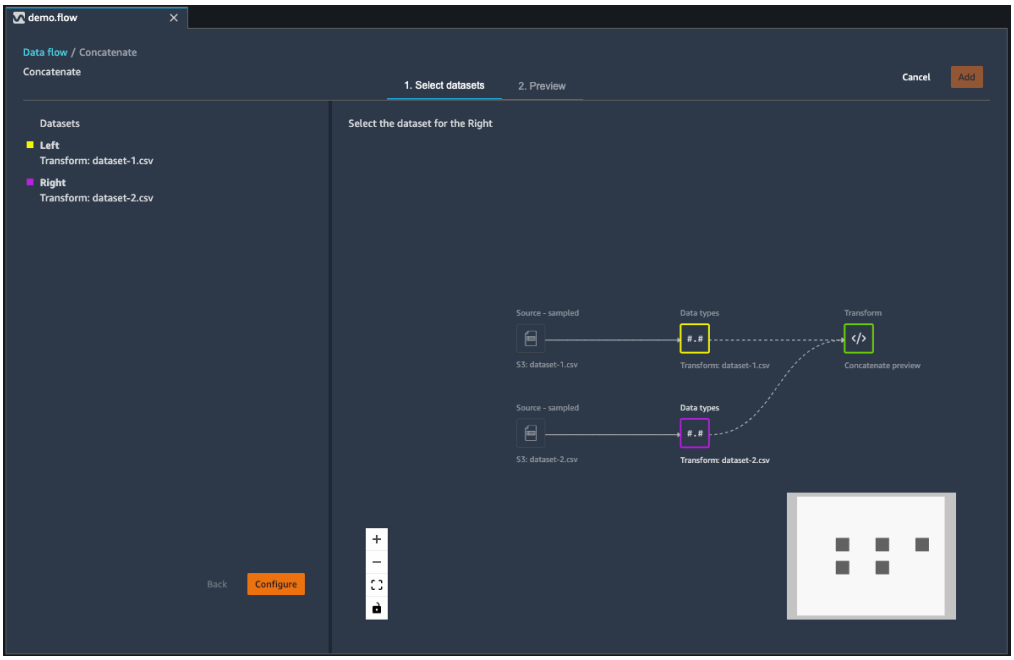
当您选择配置要配置您的加入，您会看到类似于下图所示的结果。您的加入配置显示在左侧面板中。您可以使用此面板选择已连接的数据集名称、联接类型和要联接的列。主面板显示三个表格。前两个表格分别在左侧和右侧显示左侧和右侧数据集。在此表格下，您可以预览已连接的数据集。



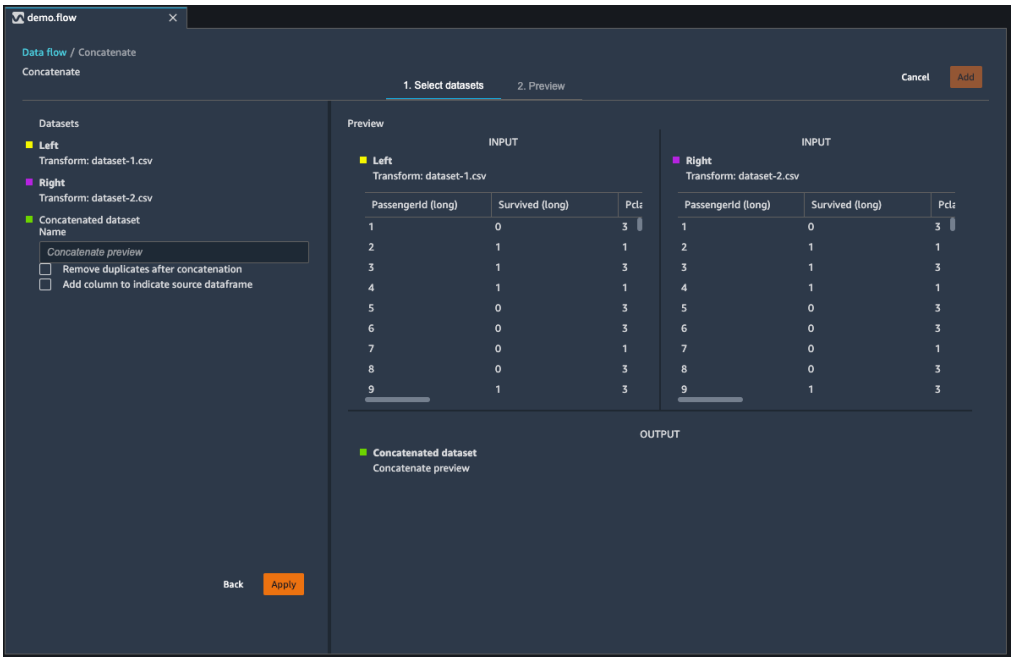
请参阅加入数据集 (p. 627)了解更多信息。

Concatenate View

要连接两个数据集，请选择数据流中的第一个数据集，然后选择联接. 当你选择联接，您会看到类似于下图所示的结果。左侧和右侧的数据集显示在左侧面板中。主面板显示您的数据流，并添加了新连接的数据集。



当您选择配置要配置连接，您会看到类似于下图所示的结果。您的连接配置显示在左侧面板中。您可以使用此面板选择连接数据集的名称，然后选择在连接后删除重复项，然后添加列以指示源数据框。主面板显示三个表格。前两个表格分别在左侧和右侧显示左侧和右侧数据集。在此表格下，您可以预览已连接的数据集。



请参阅连接数据集 (p. 628)了解更多信息。

加入数据集

您可以直接在数据流中加入数据框。连接两个数据集时，生成的联接数据集将显示在流程中。Data Wrangler 支持以下联接类型。

- OUTER Left— 包括左表中的所有行。如果在左表行中联接的列的值与任何右侧表格行值不匹配，则该行包含已连接表中所有右侧表列的空值。
- Left Anti— 包括左表中不包含已连接列的右表中的值的行。
- Left 半— 在左表中为满足 join 语句中标准的所有相同行包括一行。这将从左表中排除符合联接条件的重复行。
- 右外部— 包括右表中的所有行。如果右表行中连接的列的值与任何左表行值不匹配，则该行包含已连接表中所有左表列的空值。
- INNER— 包括左右表中包含连接的列中包含匹配值的行。
- 完全外部— 包括左右表格中的所有行。如果任一表中连接列的行值不匹配，则会在连接的表中创建单独的行。如果行不包含连接表中某列的值，则为该列插入 null。
- 笛卡尔十字— 包括将第一个表中的每一行与第二个表中的每一行组合在一起的行。这是笛卡尔积来自联接中的表中的行。该产品的结果是左桌的大小乘以右桌的大小。因此，我们建议小心在非常大的数据集之间使用此连接。

要加入两个数据框，请使用以下过程。

1. Select+在要加入的左侧数据框旁边。您选择的第一个数据框始终是连接中的左表。
2. 选择加入。
3. 选择正确的数据框。您选择的第二个数据框始终是连接中的正确表格。
4. 选择配置来配置你的加入。
5. 使用为已加入的数据集命名名称字段中返回的子位置类型。
6. 选择联接类型.
7. 从左侧和右侧的表格中选择一列进行加入。
8. 选择Apply以便在右侧预览已连接的数据集。
9. 要将联接的表添加到数据流中，请选择Add.

连接数据集

连接两个数据集：

1. 选择+在要连接的左侧数据框旁边。您选择的第一个数据框始终是连接中的左表。
2. 选择联接.
3. 选择正确的数据框。您选择的第二个数据框始终是连接中的正确表格。
4. 选择配置来配置你的连接。
5. 使用为连接的数据集命名名称字段中返回的子位置类型。
6. （可选）选中旁边的复选框连接后删除重复项删除重复的列。
7. （可选）选中旁边的复选框添加列以指示源数据框如果对于新数据集中的每一列，如果要添加列源的指示符。
8. 选择Apply以预览新数据集。
9. 选择Add将新数据集添加到数据流中。

余额数据

您可以平衡不足类别的数据集的数据集的数据。平衡数据集可以帮助您创建更好的二进制分类模型。

> Note
>
> 您无法平衡包含列矢量的数据集。

您可以使用余额数据使用以下运算符之一来平衡数据的操作：

- 随机过采样— 随机复制少数群体类别中的样本。例如，如果你试图检测欺诈行为，你可能只有 10% 的数据中有欺诈案例。对于同等比例的欺诈和非欺诈案件，该运营商会在数据集中随机重复 8 次欺诈案件。
- 随机采样不足— 大致相当于随机过采样。随机从显示过高的类别中删除样本，以获取您想要的样本比例。
- 合成少数过采样技术 (SMOTE)— 使用代表不足类别的样本来插入新的合成少数群体样本。有关 SMOTE 的更多信息，请参阅以下说明。

您可以对包含数字和非数字要素的数据集使用所有变换。SMOTE 使用相邻样本对值进行插值。Data Wrangler 使用 R 平方距离来确定用于插值额外样本的邻域。Data Wrangler 仅使用数字要素来计算代表不足组中样本之间的距离。

对于代表不足组中的两个实际样本，Data Wrangler 使用加权平均值来插值数字要素。它会随机为 [0, 1] 范围内的样本分配权重。对于数值要素，Data Wrangler 使用样本的加权平均值来插值样本。对于样本 A 和 B，Data Wrangler 可以随机分配 0.7 的权重给 A，0.3 给 B。插值样本的值为 0.7A + 0.3B。

Data Wrangler 通过从任一插值实际样本中复制来插值非数字要素。它复制样本的概率是随机分配给每个样本。对于样本 A 和 B，它可以将概率 0.8 分配给 A，0.2 分给 B。对于它所分配的概率，它会复制 80% 的时间。

自定义转换

这些区域有：自定义转换组允许你使用 Python（用户定义函数）、Pyspark、熊猫或 Pyspark (SQL) 来定义自定义转换。对于所有三个选项，都可以使用变量`df`以访问要将转换应用到的数据框。如果您不使用 Python（用户定义函数），则无需包含返回语句。选择预览以预览自定义转换的结果。选择Add将自定义转换添加到你的列表之前的步骤.

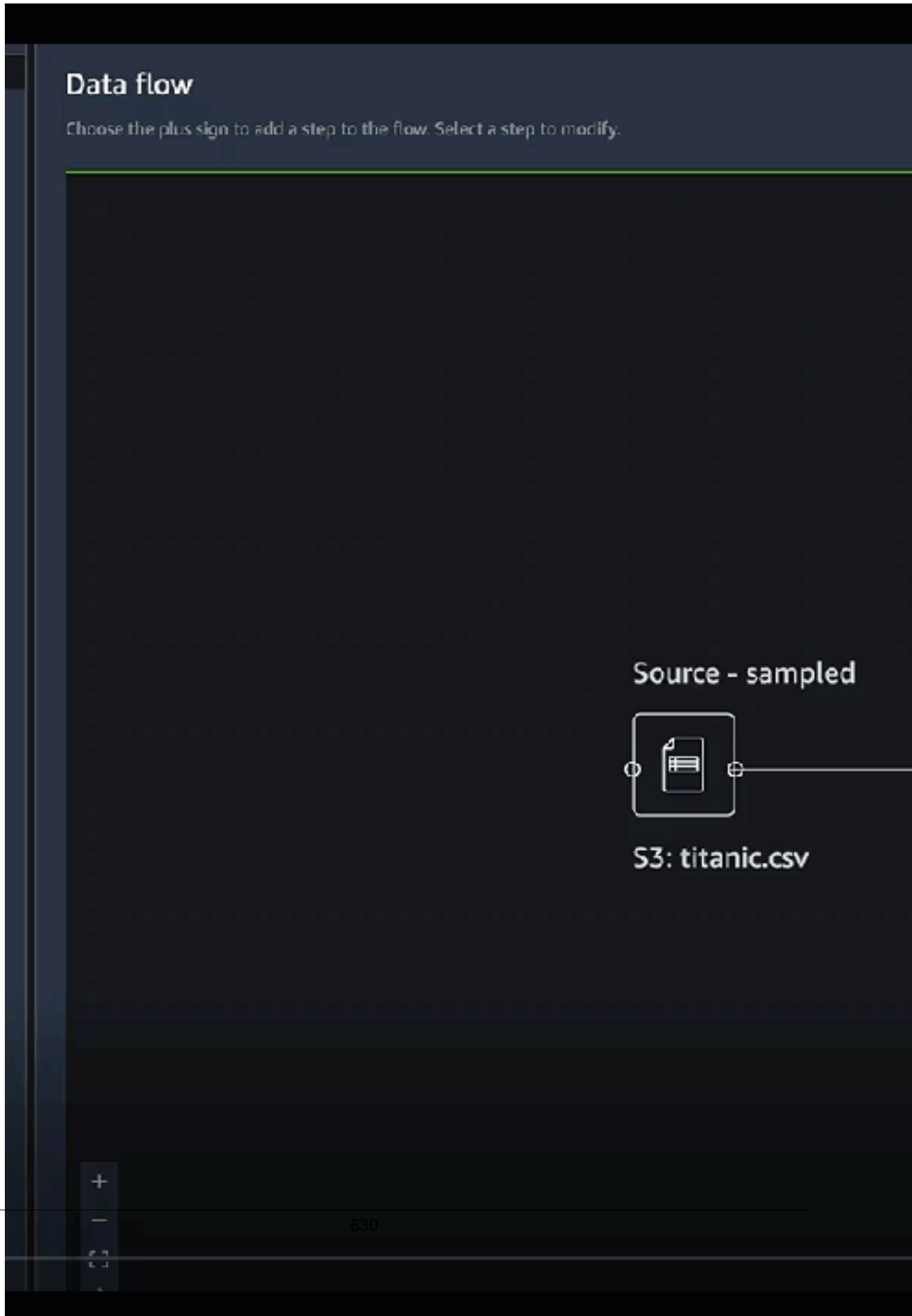
你可以使用`import`自定义转换代码块中的语句，如下所示：

- Numpy 版本 1.19.0
- Scikit-Learn 版本 0.23.2
- SciPy 版本 1.5.4
- 熊猫版本 1.0.3
- Pyspark 版本 3.0.0

> **Important**
>
> 自定义转换不支持名称中包含空格或特殊字符的列。我们建议您指定仅包含字母数字字符和下划线的列名称。您可以使用重命名列转换为管理列转换组以从列名中删除空格。您还可以添加熊猫 自定义转换类似于以下内容，可以在一个步骤中删除多个列中的空格。此示例更改名为`A column`和`B column`到`A_column`和`B_column`分别。

```
df.rename(columns={"A column": "A_column", "B column": "B_column"})
```

如果在代码块中包含打印语句，则当您选择预览. 你可以调整自定义代码变压器面板的大小 调整面板大小可提供更多编写代码的空间。下面显示了调整面板的大小。


Data flow
Choose the plus sign to add a step to the flow. Select a step to modify.
Source - sampled
S3: titanic.csv

下面提供了用于编写自定义转换代码的其他上下文和示例。

Python（用户定义的函数）

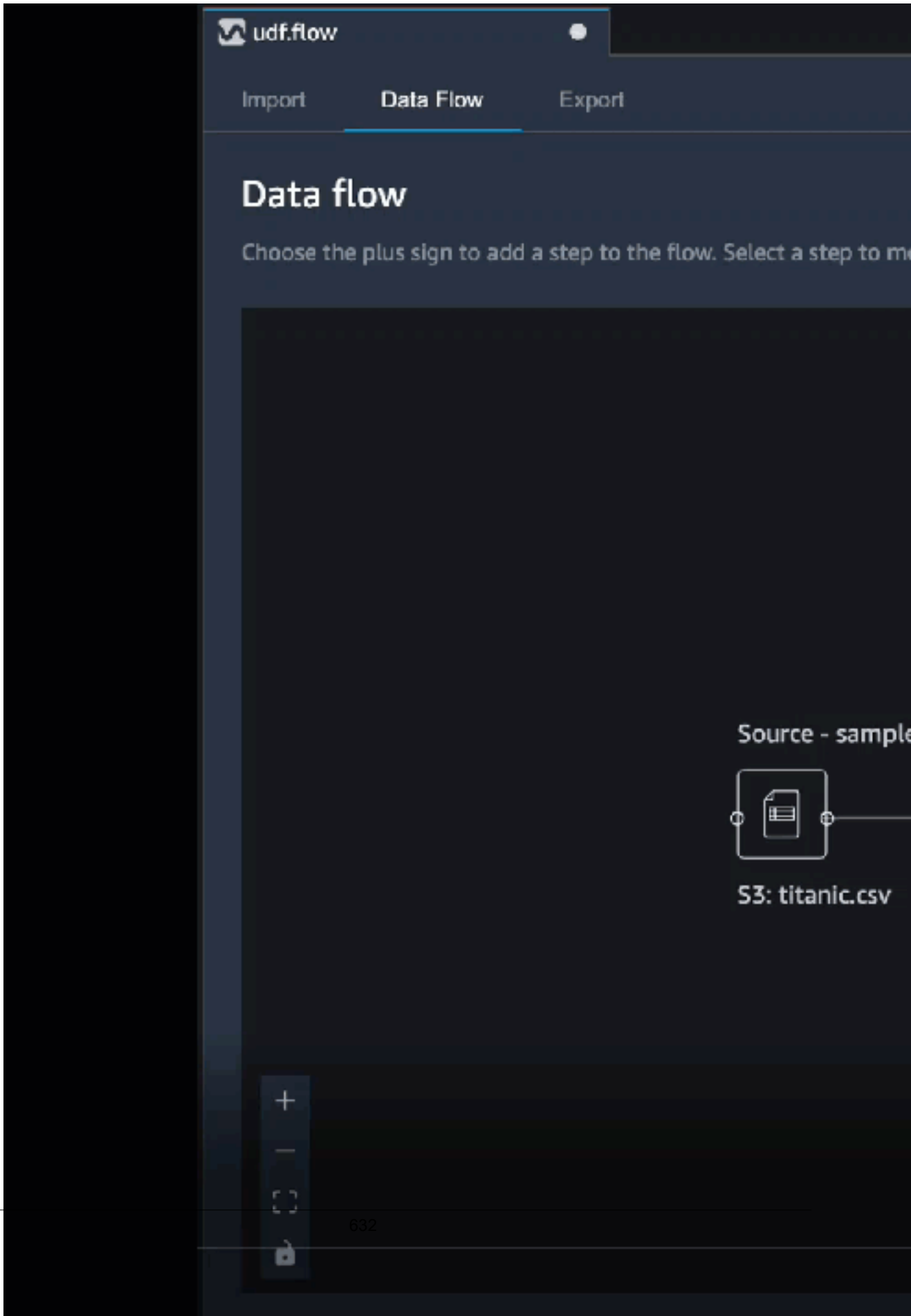
Python 函数使您能够在不知道 Apache Spark 或 Pandas 的情况下编写自定义转换。Data Wrangler 经过优化，可以快速运行自定义代码。在使用自定义 Python 代码和 Apache Spark 插件之间，你会获得类似的性能。

要使用 Python（用户定义函数）代码块，请指定以下内容：

- 输入栏— 要应用转换的输入列。
- Mode (模式)— 脚本模式，无论是熊猫还是 Python。
- 返回类型— 您要返回的值的数据类型。

使用 Pandas 模式可提供更好的性能。Python 模式使您可以通过使用纯 Python 函数更轻松地编写转换。

以下视频显示了如何使用自定义代码创建转换的示例。它使用泰坦尼克号数据集来创建一个包含该人称呼的列。


udf.flow
Import
Data Flow
Export
Data flow
Choose the plus sign to add a step to the flow. Select a step to m
Source - sample
S3: titanic.csv

pSpark

以下示例从时间戳中提取日期和时间。

```
from pyspark.sql.functions import from_unixtime, to_date, date_format
df = df.withColumn('DATE_TIME', from_unixtime('TIMESTAMP'))
df = df.withColumn( 'EVENT_DATE', to_date('DATE_TIME')).withColumn(
'EVENT_TIME', date_format('DATE_TIME', 'HH:mm:ss'))
```

熊猫

以下示例概述了要向其添加转换的数据框。

```
df.info()
```

Pyspark (SQL)

以下内容创建了一个包含五列的新数据框：名称、票价、pclass、幸存下来.

```
SELECT name, fare, pclass, survived FROM df
```

自定义公式

使用自定义公式使用 Spark SQL 表达式来查询当前数据框中的数据来定义新列。查询必须使用 Spark SQL 表达式的约定。

> **Important**
>
> 自定义公式不支持名称中包含空格或特殊字符的列。我们建议您指定仅包含字母数字字符和下划线的列名称。您可以使用重命名列转换为管理列转换组以从列名中删除空格。您还可以添加熊猫 自定义转换类似于以下内容，可以在一个步骤中删除多个列中的空格。此示例更改名为`A column`和`B column`到`A_column`和`B_column`分别。
>
> ```
> df.rename(columns={"A column": "A_column", "B column": "B_column"})
> ```

您可以使用此转换对列执行操作，并按名称引用列。例如，假设当前数据框包含名为的列col_a和col_b，您可以使用以下操作生成输出列这是这两列的产物，其中包含以下代码：

```
col_a * col_b
```

其他常见操作包括以下内容，假设数据框包含`col_a`和`col_b`列:

- 连接两列：`concat(col_a, col_b)`
- 添加两列：`col_a + col_b`
- 减去两列：`col_a - col_b`
- 划分两列：`col_a / col_b`
- 以列的绝对值：`abs(col_a)`

有关更多信息，请参阅 。Spark 文档在选择数据时。

分类编码

分类数据通常由有限数量的类别组成，其中每个类别都用字符串表示。例如，如果您有客户数据表，则表明某人居住的国家/地区的列是分类的。类别将是阿尔巴尼亚、阿尔巴尼亚、阿尔及利亚等。分类数据可以名义

上的要么序数词. 序数类别有固有的顺序，名义类别则没有。获得的最高学位（高中、单身汉、硕士）就是序数类别的一个例子。

对分类数据进行编码是为类别创建数字表示形式的过程。例如，如果你的类别是 Dog 和 Cat，你可以将这些信息编码为两个向量，`[1,0]`代表狗，以及`[0,1]`来表示 Cat。

对序号类别进行编码时，可能需要将类别的自然顺序转换为编码。例如，您可以表示通过以下地图获得的最高学位：`{"High school": 1, "Bachelors": 2, "Masters":3}`.

使用分类编码将字符串格式的分类数据编码为整数数组。

Data Wrangler 分类编码器在定义步骤时为列中存在的所有类别创建编码。启动 Data Wrangler 作业以及时处理数据集时，如果已将新类别添加到列t，此专栏是当时 Data Wrangler 分类编码转换的输入t-1，考虑这些新类别缺失的在数据牧马人工作中。你选择的选项无效的处理策略应用于这些缺失值。什么时候会发生这种情况的例子是：

- 当您使用 .flow 文件创建 Data Wrangler 作业以处理数据流创建后更新的数据集时。例如，您可以使用数据流每月定期处理销售数据。如果每周更新一次销售数据，则可能会将新的类别引入到已定义编码分类步骤的列中。
- 当你选择采样导入数据集时，某些类别可能会被排除在样本之外。

在这些情况下，这些新类别被视为 Data Wrangler 工作中缺少的值。

您可以选择和配置序数词和一热编码. 可以使用以下部分了解这些选项的更多信息。

两个转换都创建一个名为输出列名称. 您可以使用指定此列的输出格式输出样式：

- SelectVector来生成带有稀疏矢量的单列。
- Select列为每个类别创建一个列，其中包含一个指标变量，用于指示原始列中的文本是否包含等于该类别的值。

序数编码

Select序数编码将类别编码为介于 0 和中类别总数之间的整数输入栏您选择。

无效的交付策略：选择一种方法来处理无效或缺失值。

- 选择跳过如果你想省略缺少值的行。
- 选择保持将缺失值保留为最后一个类别。
- 选择错误如果你希望 Data Wrangler 抛出错误，如果在输入栏.
- 选择替换为 NaN用 NaN 替换缺失。如果 ML 算法可以处理缺失值，则建议使用此选项。否则，此列表中的前三个选项可能会产生更好的结果。

一热编码

Select一热编码为了转换使用一热编码。使用以下命令配置此转换：

- 删除最后一类别：如果`True`，最后一个类别在单热编码中没有相应的索引。当可能缺少值时，丢失的类别始终是最后一个类别，并将其设置为`True`意味着缺少值会导致全零向量。
- 无效的交付策略：选择一种方法来处理无效或缺失值。
  - 选择跳过如果你想省略缺少值的行。
  - 选择保持将缺失值保留为最后一个类别。
  - 选择错误如果你希望 Data Wrangler 抛出错误，如果在输入栏.

- 输入序数是否已编码：如果输入向量包含序号编码数据，请选择此选项。此选项要求输入数据包含非负整数。如果True，输入i被编码为矢量中的非零i位置。

相似性编码

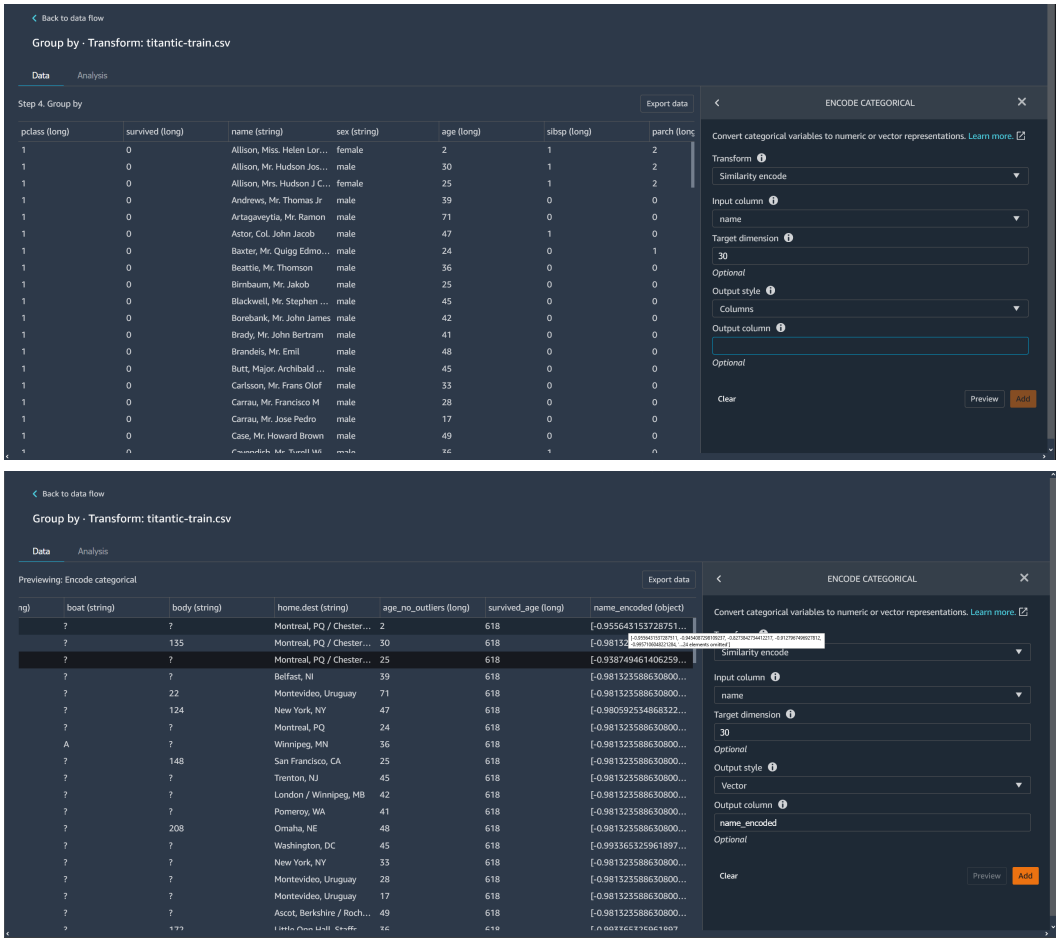
如果有以下条件，请使用相似性编码：

- 大量的类别变量
- 嘈杂的数据

相似度编码器为包含分类数据的列创建嵌入。嵌入是将离散对象（例如单词）与实数向量的映射。它将类似的字符串编码为包含相似值的矢量。例如，它为“加利福尼亚州”和“加州”创建了非常类似的编码。

Data Wrangler 使用 3 克标记器将数据集中的每个类别转换为一组令牌。它使用 min-hash 编码将令牌转换为嵌入。

以下示例说明相似度编码器如何从字符串中创建向量。



Data Wrangler 创建的相似性编码：

- 维度较小
- 可扩展至大量类别
- 坚固耐用且耐噪音

出于上述原因，相似性编码比单热编码更通用。

要向数据集中添加相似性编码转换，请按照以下过程操作。

要使用相似性编码，请执行以下操作。

1. 登录到Amazon SageMaker 控制台.
2. 选择打开 Studio。
3. 选择启动应用.
4. 选择工作室.
5. 指定数据流。
6. 选择转换的步骤。
7. 选择添加步骤.
8. 选择分类编码.
9. 指定以下内容：

   - 转换–编码相似性
   - 输入栏— 包含正在编码的分类数据的列。
   - 目标维度—（可选）分类嵌入向量的维度。默认值是 30。如果您的大型数据集包含许多类别，我们建议使用较大的目标维度。
   - 输出样式— 选择Vector对于包含所有编码值的单个向量。选择列将编码的值放在单独的列中。
   - 输出列—（可选）矢量编码输出的输出列的名称。对于列编码的输出，列名的前缀后跟列出的数字。

特色文本

使用功能文本转换组来检查字符串类型的列，并使用文本嵌入来显示这些列。

此功能组包含两个功能，字符统计数和Vectorize. 可以使用以下部分了解这些转换的更多信息。对于这两种选项，输入栏必须包含文本数据（字符串类型）。

角色统计数

使用字符统计数为包含文本数据的列中的每一行生成统计信息。

这将转换为每行计算以下比率和计数，并创建一个新列来报告结果。使用输入列名称作为前缀和特定于比率或计数的后缀来命名新列。

- 字数：该行中的单词总数。此输出列的后缀为`-stats_word_count`.
- 字符数：该行中的字符总数。此输出列的后缀为`-stats_char_count`.
- 上限的比例：从 A 到 Z 的大写字符数除以列中的所有字符。此输出列的后缀为`-stats_capital_ratio`.
- 较低的比率：从 a 到 z 的小写字符数除以列中的所有字符。此输出列的后缀为`-stats_lower_ratio`.
- 数字比率：单行中的位数与输入列中数字总和的比率。此输出列的后缀为`-stats_digit_ratio`.
- 特殊字符比例：非字母数字（如 #$&%: @）字符与输入列中所有字符之和的比率。此输出列的后缀为`-stats_special_ratio`.

Vectorize

文本嵌入涉及将词汇中的单词或短语映射到实数矢量。使用 Data Wrangler 文本嵌入变换将文本数据标记化并将其矢量化为术语频率 — 反向文档频率 (TF-IDF) 矢量。

当为一列文本数据计算 TF-IDF 时，每句话中的每个单词都将转换为代表其语义重要性的实数。数字越高与不太频繁的单词有关，这往往更有意义。

当你定义Vectorize在定义此步骤时，使用 Data Wrangler 中的可用数据来定义转换步骤、计数矢量器和 TF-IDF 方法。运行 Data Wrangler 作业时也使用这些相同的方法。

您可以使用以下命令配置此转换：

- 输出列名称：此转换将创建一个带有文本嵌入的新列。使用此字段可以指定此输出列的名称。
- 令牌化：分词器将句子转换为单词列表，或者象征.

  选择标准使用分割空格并将每个单词转换为小写字母的分词器。例如，`"Good dog"`被标记化为`["good","dog"]`.

  选择Custom (自定义)使用自定义的标记器。如果选择Custom (自定义)，您可以使用以下字段来配置标记器：
  - 最短令牌长度：令牌有效的最小长度（以字符为单位）。默认值为 `1`。例如，如果指定`3`对于最短令牌长度，例如`a, at, in`从标记化的句子中删除。
  - 正则表达式应该在差距上拆分：如果选择此项，regex分裂差距。否则，它与代币匹配。默认值为 `True`。
  - 正则表达式模式：定义标记化过程的正则表达式模式。默认值为 `' \\ s+'`。
  - 小写形式：如果选择此选项，则在标记化之前将所有字符转换为小写字母。默认值为 `True`。

  要了解更多信息，请参阅上的 Spark 文档令牌化.
- Vectorizer：矢量器将标记列表转换为稀疏的数字矢量。每个令牌对应于向量中的索引，非零表示输入句子中存在该令牌。您可以从两个矢量器选项中进行选择，计数和HASHING.
  - 计数向量允许过滤不频繁或太常见的令牌的自定义项。计数矢量化参数中支持的脚本编写选项如下：
    - 最小学期频率：在每一行中，都会过滤频率较小的术语（令牌）。如果指定整数，则这是绝对阈值（包括在内）。如果指定介于 0（含）和 1 之间的分数，则阈值是相对于总期限计数的相对值。默认值为 `1`。
    - 最小文档频率：必须包含术语（令牌）的最小行数。如果指定整数，则这是绝对阈值（包括在内）。如果指定介于 0（含）和 1 之间的分数，则阈值是相对于总期限计数的相对值。默认值为 `1`。
    - 最大文档频率：可以包含术语（令牌）的最大文档（行）数。如果指定整数，则这是绝对阈值（包括在内）。如果指定介于 0（含）和 1 之间的分数，则阈值是相对于总期限计数的相对值。默认值为 `0.999`。
    - 最大词汇量：词汇量的最大大小。词汇表由列中所有行中的所有术语（令牌）组成。默认值为 `262144`。
    - 二进制输出：如果选中此项，矢量输出不包括文档中某个术语的出现次数，而是其外观的二进制指标。默认值为 `False`。

    要了解有关此选项的更多信息，请参阅上的 Spark 文档。CountVectorizer.
  - HASHING计算速度更快。哈希矢量化参数包括以下内容：
    - 散列过程中的功能数量：哈希矢量器根据令牌的哈希值将令牌映射到矢量索引。此功能决定了可能的哈希值的数量。较大的值会减少哈希值之间的冲突，但维度较高的输出矢量。

    要了解有关此选项的更多信息，请参阅上的 Spark 文档。功能 Hasher
- 应用 IDF：选择后，将应用 IDF 转换，该转换将术语频率与用于 TF-IDF 嵌入的标准逆文档频率乘以。IDF 参数中支持的脚本编写选项如下：
  - 最小文档频率：必须包含术语（令牌）的文档（行）的最小数量。如果count_vectorize是选择的矢量器，我们建议保留默认值，并且只修改min_doc_freq字段计数矢量化参数. 默认值为 `5`。
- 输出格式: 每行的输出格式。
  - SelectVector来生成带有稀疏矢量的单列。
  - Select扁平为每个类别创建一个列，其中包含一个指标变量，用于指示原始列中的文本是否包含等于该类别的值。您只能选择扁平的时候Vectorizer设置为计数向量.

转换时间序列

在 Data Wrangler 中，您可以转换时间序列数据。时间序列数据集中的值将索引至特定时间。例如，显示一天中每小时商店中每个小时的客户数量的数据集就是时间序列数据集。下表显示了时间序列数据集示例。

商店中的每小时客户数

Customer Number	时间（小时）
4	09:00
10	10:0:00
14	11:0:00
25	12:00
20	13:00
18	14:00

对于上表，Customer Number列包含时间序列数据。时间序列数据根据“时间（小时）”列中的每小时数据编制索引。

您可能需要对数据执行一系列转换，才能以可用于分析的格式获取数据。使用时间序列转换组以转换你的时间序列数据。有关您可以执行的转换的更多信息，请参阅以下部分。

主题

- 按时间序列分组 (p. 638)
- 重新采样时间序列数据 (p. 639)
- 处理缺失时间序列数据 (p. 640)
- 验证时间序列数据的时间戳 (p. 642)
- 标准化时间序列的长度 (p. 643)
- 从时序数据中提取要素 (p. 644)
- 使用时序数据中的滞后功能 (p. 644)
- 在时间序列中创建日期时间范围 (p. 645)
- 在你的时间序列中使用滚动窗口 (p. 646)

按时间序列分组

您可以使用按操作分组对列中特定值的时间序列数据进行分组。

例如，您有下表跟踪了家庭的平均每日用电量。

平均每日家庭用电

家庭 ID	日常时间戳	用电量 (kWh)	住户数量
家庭 _0	2020 年 1 月 1 日	30	2
家庭 _0	1/2/2020	40	2
家庭 _0	1/4/2020	35	3
家庭 _1	1/2/2020	45	3

家庭 ID	日常时间戳	用电量 (kWh)	住户数量
家庭 _1	1/3/2020	55	4

如果你选择按 ID 分组，你会得到下表。

按家庭 ID 分组的电力使用情况

家庭 ID	用电系列 (kWh)	家庭居住者数量系列
家庭 _0	[30、40、35]	[2、2、3、2、3]
家庭 _1	[45, 55]	[3, 4]

时间序列序列中的每个条目都按相应的时间戳排序。序列的第一个元素对应于该系列的第一个时间戳。适用于`household_0`、`30`是的第一个值电力使用系列. 的价值`30`对应于的第一次时间戳`1/1/2020`.

您可以包括开始时间戳和结束时间戳。下表显示了如何显示的示例。

按家庭 ID 分组的电力使用情况

家庭 ID	用电系列 (kWh)	家庭居住者数量系列	Start_time	end_time
家庭 _0	[30、40、35]	[2、2、3、2、3]	2020 年 1 月 1 日	1/4/2020
家庭 _1	[45, 55]	[3, 4]	1/2/2020	1/3/2020

您可以使用以下过程按照时间序列来批量。

1. 打开数据 Wrangler 数据流。
2. 如果尚未导入数据集，请将其导入到导入数据选项卡。
3. 在你的数据流中，在数据类型，选择+Select, 然后选择添加转换.
4. 选择添加步骤.
5. 选择时间序列.
6. UNDER转换，选择Group by (分组依据).
7. 在中指定一列按此专栏进行分组.
8. 适用于应用于列，指定一个值。
9. 选择预览以生成转换的预览。
10. 选择Add将转换添加到 Data Wrangler 数据流中。

重新采样时间序列数据

时间序列数据通常有不定期进行的观测值。例如，数据集可以有一些观测值每小时记录一次，另一些观测值每两小时记录一次。

许多分析，例如预测算法，都需要定期进行观测。通过重新采样，您可以为数据集中的观测值建立定期间隔。

您可以对时间序列进行上采样或降样。缩减采样会增加数据集中观测值之间的间隔。例如，如果对每小时或每两小时采取一次的观测值进行降样，则数据集中的每个观测值将每两小时进行一次。使用平均值或中位数之类的聚合方法将每小时观测值聚合为单个值。

上采样可缩短数据集中观测值之间的间隔。例如，如果您将每两小时取得的观测值上采样为小时观测值，则可以使用插值方法从每两小时获取的观测值中推断每小时观测值。有关插值方法的信息，请参阅Pandas.dataframe.插入插值.

您可以对数字和非数字数据重新采样。

使用重新采样对时间序列数据进行重新采样的操作。如果数据集中有多个时间序列，Data Wrangler 会标准化每个时间序列的时间间隔。

以下是使用均值作为聚合方法缩减时间序列数据的示例。数据从每两小时减少到每小时一次。

缩减采样前一天的每小时温度读数

时间戳	温度（摄氏度）
12:00	30
1:01:00	32
2:00	35
3:0:00	32
4:00	30

温度读数降至每两小时

时间戳	温度（摄氏度）
12:00	30
2:00	33.5
2:00	35
4:00	32.5

您可以使用以下过程对时间序列数据重新采样。

1. 打开数据 Wrangler 数据流。
2. 如果尚未导入数据集，请将其导入到导入数据选项卡。
3. 在你的数据流中，在数据类型，选择+Select, 然后选择添加转换.
4. 选择添加步骤.
5. 选择重新采样.
6. 适用于时间戳中，选择时间戳列。
7. 适用于Frequency中，指定要重新采样的频率。
8. (可选) 为频率数.
9. 要配置其余字段，则可以配置转换。
10. 选择预览以生成转换的预览。
11. 选择Add将转换添加到 Data Wrangler 数据流中。

处理缺失时间序列数据

如果数据集中缺少值，则可以执行以下操作之一：

- 对于具有多个时间序列的数据集，删除缺失值大于指定阈值的时间序列。
- 使用时间序列中的其他值来推断时间序列中的缺失值。

计算缺失值涉及通过指定值或使用推理方法来替换数据。以下是你可以用来计算的方法：

- 常量值 — 将数据集中所有缺失的数据替换为指定的值。
- 最常见的值 — 将所有丢失的数据替换为数据集中频率最高的值。
- 向前填充 — 使用向前填充将缺失的值替换为缺失值之前的不缺失值。对于序列：[2、4、7、NaN、NaN、NaN、8]，所有缺失的值都被 7 替换。使用正向填充产生的顺序是 [2、4、7、7、7、7、7、8]。
- 向后填充 — 使用向后填充将缺失值替换为缺失值后面的不缺失值。对于序列：[2、4、7、NaN、NaN、NaN、8]，所有缺失的值都被 8 替换。使用向后填充所产生的顺序是 [2、4、7、8、8、8、8]。
- 插值 — 使用插值函数来计算缺失的值。有关可用于插值的函数的更多信息，请参阅Pandas.dataframe.插入插值.

某些估算方法可能无法计算数据集中的所有缺失值。例如，向前填充不能推断出现在时间序列开头的缺失值。您可以通过使用向前填充或向后填充来计算值。

您可以归因为单元格内或列中的缺失值。

以下示例说明如何在单元格中估算值。

缺少值的电力使用

家庭 ID	用电系列 (kWh)
家庭 _0	[30、40、35、NaN、NaN]
家庭 _1	[45、NaN、55]

使用正向填充计算值的电力使用情况

家庭 ID	用电系列 (kWh)
家庭 _0	[30、40、35、35、35]
家庭 _1	[45、45、55]

以下示例说明如何在列中估算值。

缺少值的家庭平均每日用电量

家庭 ID	用电量 (kWh)
家庭 _0	30
家庭 _0	40
家庭 _0	NaN
家庭 _1	NaN
家庭 _1	NaN

平均每日家庭用电量，使用前期填充估算值

家庭 ID	用电量 (kWh)
家庭 _0	30
家庭 _0	40
家庭 _0	40
家庭 _1	40
家庭 _1	40

您可以使用以下过程处理缺少的值。

1. 打开数据 Wrangler 数据流。
2. 如果尚未导入数据集，请将其导入到导入数据选项卡。
3. 在你的数据流中，在数据类型，选择+Select, 然后选择添加转换.
4. 选择添加步骤.
5. 选择缺少处理.
6. 适用于时间序列输入类型中，选择是要处理单元格内部还是沿列处理缺少的值。
7. 适用于归因为此列的缺失值中，指定包含缺少值的列。
8. 适用于计算值的方法中，选择一种方法。
9. 要配置其余字段，则可以配置转换。
10. 选择预览以生成转换的预览。
11. 如果缺少值，可以在下面指定一种方法来计算它们计算值的方法.
12. 选择Add将转换添加到 Data Wrangler 数据流中。

验证时间序列数据的时间戳

您可能有无效的时间戳数据。您可以使用验证时间戳函数来确定数据集中的时间戳是否有效。由于以下一个或多个原因，您的时间戳可能无效：

- 您的时间戳列缺少值。
- 时间戳列中的值格式不正确。

如果数据集中有无效的时间戳，则无法成功执行分析。您可以使用 Data Wrangler 来识别无效的时间戳，并了解需要在哪里清理数据。

时间序列验证有两种方式之一：

如果 Data Wrangler 在数据集中遇到缺少值时，您可以将 Data Wrangler 配置为执行以下操作之一：

- 删除缺少值或无效值的行。
- 识别具有缺少值或无效值的行。
- 如果它在数据集中发现任何缺失或无效值，则抛出错误。

您可以在具有`timestamp`键入或`string`。如果该列有`string`类型，Data Wrangler 将列的类型转换为`timestamp`然后执行验证。

您可以使用以下过程验证数据集中的时间戳。

1. 打开数据 Wrangler 数据流。
2. 如果尚未导入数据集，请将其导入到导入数据选项卡。
3. 在你的数据流中，在数据类型，选择+Select, 然后选择添加转换.
4. 选择添加步骤.
5. 选择验证时间戳.
6. 适用于时间戳列中，选择时间戳列。
7. 适用于策略中，选择是否要处理缺少的时间戳。
8. （可选）对于输出列中，指定输出列的名称。
9. 如果为字符串类型设置了日期时间列的格式，请选择强制转换为日期时间.
10. 选择预览以生成转换的预览。
11. 选择Add将转换添加到 Data Wrangler 数据流中。

标准化时间序列的长度

如果您将时间序列数据存储为数组，则可以将每个时间序列标准化为相同的长度。标准化时间序列数组的长度可能会使您更轻松地对数据进行分析。

对于需要固定数据长度的数据转换，您可以对时间序列进行标准化。

许多 ML 算法都要求在使用时序数据之前对时间序列数据进行拼合。拼合时间序列数据是将时间序列的每个值分成数据集中自己的列。数据集中的列数不能更改，因此需要在将每个数组拼合为一组要素之间标准化时间序列的长度。

每个时间序列都设置为您指定为时间序列集的分位数或百分位数的长度。例如，您可以有三个具有以下长度的序列：

- 3
- 4
- 5

您可以将所有序列的长度设置为具有 50 个百分位数长度的序列的长度。

小于指定长度的时间序列数组会添加缺失值。以下是将时间序列标准化为更长长度的示例格式：
[2、4、5、NaN、NaN、NaN]。

您可以使用不同的方法来处理缺失的值。有关这些方法的信息，请参阅处理缺失时间序列数据 (p. 640).

长于指定长度的时间序列数组将被截断。

您可以使用以下过程来标准化时间序列的长度。

1. 打开数据 Wrangler 数据流。
2. 如果尚未导入数据集，请将其导入到导入数据选项卡。
3. 在你的数据流中，在数据类型，选择+Select, 然后选择添加转换.
4. 选择添加步骤.
5. 选择标准化长度.
6. 适用于标准化列的时间序列长度中，选择一列。
7. （可选）对于输出列中，指定输出列的名称。如果您没有指定名称，转换就已到位。
8. 如果为字符串类型设置了日期时间列的格式，请选择强制转换为日期时间.
9. 选择截止分位数中，指定一个分位数来设置序列的长度。
10. 选择展平输出将时间序列的值输出到单独的列中。

11. 选择预览以生成转换的预览。
12. 选择Add将转换添加到 Data Wrangler 数据流中。

从时序数据中提取要素

如果您对时间序列数据运行分类或回归算法，我们建议在运行算法之前从时间序列中提取要素。提取功能可能会提高算法的性能。

使用以下选项选择要从数据中提取要素的方式：

- 使用最小子集以指定提取您知道在下游分析中有用的 8 个要素。当你需要快速执行计算时，你可以使用最小的子集。当 ML 算法有过度拟合的高风险并且希望为其提供的功能较少时，也可以使用它。
- 使用高效的子集指定在不提取分析中计算密集型要素的情况下尽可能提取最多的要素。
- 使用所有功能以指定从 Tune 系列中提取所有要素。
- 使用手动子集以选择一个你认为很好地解释数据变化的功能列表。

使用以下步骤从时间序列数据中提取要素。

1. 打开数据 Wrangler 数据流。
2. 如果尚未导入数据集，请将其导入到导入数据选项卡。
3. 在你的数据流中，在数据类型，选择+Select, 然后选择添加转换.
4. 选择添加步骤.
5. 选择提取要素.
6. 适用于提取此列的功能中，选择一列。
7. （可选）选择Flatten将要素输出到单独的列中。
8. 适用于Strategy中，选择一个策略来提取功能。
9. 选择预览以生成转换的预览。
10. 选择Add将转换添加到 Data Wrangler 数据流中。

使用时序数据中的滞后功能

对于许多用例来说，预测时间序列 future 行为的最佳方法是使用其最近的行为。

滞后功能的最常见用途如下：

- 收集一些过去的价值观。例如，就时间而言，t + 1，你收集 t、t-1、t-2 和 t-3。
- 收集与数据中的季节性行为对应的值。例如，要预测下午 1:00 餐厅的入住率，您可能希望从前一天下午 1:00 开始使用这些功能。从当天中午 12:00 或上午 11 点开始使用这些功能可能不像使用前几天的功能那样具有预测性。

1. 打开数据 Wrangler 数据流。
2. 如果尚未导入数据集，请将其导入导入数据选项卡。
3. 在你的数据流中，在数据类型，选择+Select, 然后选择添加转换.
4. 选择添加步骤.
5. 选择滞后功能.
6. 适用于为此列生成滞后功能中，选择一列。
7. 适用于时间戳列中，选择包含时间戳的列。
8. 适用于LAG中，指定滞后的持续时间。
9. （可选）使用以下选项之一配置输出：

- 包括整个滞后窗口
- 展平输出
- 删除没有历史记录

10. 选择预览以生成转换的预览。
11. 选择Add将转换添加到 Data Wrangler 数据流中。

在时间序列中创建日期时间范围

你可能有没有时间戳的时间序列数据。如果您知道观测值是定期拍摄的，则可以在单独的列中为时间序列生成时间戳。要生成时间戳，请指定开始时间戳的值和时间戳的频率。

例如，您可能具有餐厅客户数量的以下时序数据。

餐厅客户数量的时间序列数据

Customer Number
10
14
24
40
30
20

如果您知道餐厅在下午 5 点开业且观察每小时进行，则可以添加与时间序列数据对应的时间戳列。您可以在下表中看到时间戳列。

餐厅客户数量的时间序列数据

Customer Number	时间戳
10	1:01:00 PM
14	2:00 PM
24	3:0:00 PM
40	4:00 PM
30	5:00 PM
20	6:00 PM

要向数据中添加日期时间范围，请按照以下过程操作。

1. 打开数据 Wrangler 数据流。
2. 如果尚未导入数据集，请将其导入导入数据选项卡。
3. 在你的数据流中，在数据类型，选择+Select, 然后选择添加转换.
4. 选择添加步骤.
5. 选择日期时间范围.

6. 适用于Frequency中，选择用于测量时间戳频率的单位。
7. 适用于开始时间戳中，指定开始时间戳。
8. 适用于输出列中，指定输出列的名称。
9. （可选）使用其余字段配置输出。
10. 选择预览以生成转换的预览。
11. 选择Add将转换添加到 Data Wrangler 数据流中。

在你的时间序列中使用滚动窗口

您可以在一段时间内提取要素。例如，对于时间、t 和时间窗长度为 3，对于表示第 t 个时间戳的行，我们追加从时间序列中提取的要素时间为 t-3、t -2 和 t-1。有关提取要素的信息，请参阅从时序数据中提取要素 (p. 644).

您可以使用以下过程在一段时间内提取要素。

1. 打开数据 Wrangler 数据流。
2. 如果尚未导入数据集，请将其导入导入数据选项卡。
3. 在数据流中，下数据类型，选择+Select, 然后选择添加转换.
4. 选择添加步骤.
5. 选择滚动窗口功能.
6. 适用于为此列生成滚动窗口功能中，选择一列。
7. 适用于时间戳列中，选择包含时间戳的列。
8. （可选）对于输出列中，指定输出列的名称。
9. 适用于窗口大小中，指定窗口大小。
10. 适用于Strategy，选择提取策略。
11. 选择预览以生成转换的预览。
12. 选择Add将转换添加到 Data Wrangler 数据流中。

展平日期/时间

使用特征日期/时间以创建表示日期/时间字段的矢量嵌入。要使用此转换，您的日期/时间数据必须采用以下格式之一：

- 描述日期/时间的串：例如，`"January 1st, 2020, 12:44pm"`。
- Unix 时间戳：Unix 时间戳描述了 1970 年 1 月 1 日起的秒数、毫秒、微秒或纳秒数。

您可以选择推断日期时间格式并提供日期时间格式. 如果您提供了日期/时间格式，则必须使用Python 文档. 为这两种配置选择的选项会影响操作速度和最终结果。

- 最手动和计算最快的选择是指定日期时间格式然后选择否为了推断日期时间格式.
- 为了减少体力劳动，你可以选择推断日期时间格式而不是指定日期/时间格式。这也是一种计算速度快的操作；但是，假定输入列中遇到的第一个日期/时间格式是整个列的格式。如果列中还有其他格式，则最终输出中这些值为 NaN。推断日期/时间格式可以给你未解析的字符串。
- 如果您没有指定格式，然后选择否为了推断日期时间格式，你得到了最强大的结果。解析所有有效的日期/时间字符串。但是，此操作可能比此列表中的前两个选项慢一个数量级。

使用此转换时，您可以指定输入栏其中包含以上列出的格式之一的日期/时间数据。转换会创建一个名为的输出列输出列名称. 输出列的格式取决于使用以下内容的配置：

- Vector：输出单列作为矢量。

- 列：为每个功能创建一个新列。例如，如果输出包含年、月和日，则会为年、月和日创建三个单独的列。

此外，您必须选择嵌入模式. 对于线性模型和深度网络，我们建议选择循环的. 对于基于树的算法，我们建议选择序数词.

格式字符串

这些区域有：格式字符串转换包含标准的字符串格式化操作。例如，您可以使用这些操作删除特殊字符、规范化字符串长度以及更新字符串大小写。

此功能组包含以下变换。所有转换都会返回字符串的副本输入栏然后将结果添加到新的输出列中。

名称	函数
左侧垫	用给定的字符串向左填充填充字符到给定的width. 如果字符串长度width，返回值缩短为widthcharacters.
右侧垫	用给定的向右键填充字符串填充字符到给定的width. 如果字符串长度width，返回值缩短为widthcharacters.
居中（两侧有垫）	Centrer-Pad 字符串（在字符串的两侧添加填充）填充字符到给定的width. 如果字符串长度width，返回值缩短为widthcharacters.
前缀零	用零向左填充数字字符串，最多给定width. 如果字符串长度width，返回值缩短为widthcharacters.
向左和向右分片	返回删除前导和尾随字符的字符串的副本。
从左侧删除字符	返回删除前导字符的字符串的副本。
从右边删除角色	返回删除了尾随字符的字符串的副本。
小写	将文本中的所有字母转换为小写。
大写	将文本中的所有字母转换为大写。
资本化	将每句话中的第一个字母大写。
交换案	将给定字符串的所有大写字符转换为小写字符，将所有小写字符转换为大写字符，然后返回它。
添加前缀或后缀	在字符串列中添加前缀和后缀。您必须指定至少一个前缀和后缀.
移除符号	从字符串中删除给定符号。所有列出的字符都将被删除。默认为空格。

处理异常值

机器学习模型对功能值的分布和范围很敏感。异常值或稀有值可能会对模型准确性产生负面影响，并导致更长的训练时间。使用此要素组可以检测和更新数据集中的异常值。

当你定义处理异常值转换步骤，用于检测异常值的统计信息是在定义此步骤时根据 Data Wrangler 中的可用数据生成的。运行 Data Wrangler 作业时，使用这些相同的统计信息。

有关此组包含的转换的更多信息，请使用以下部分了解以下部分。你指定输出名称这些变换中的每一个都会生成一个包含结果数据的输出列。

稳健的标准差数字异常值

此转换使用对异常值强大的统计数据来检测并修复数字要素中的异常值。

你必须定义上分位数和下分位数，它们用于计算异常值的统计数据。您还需要指定标准偏差值必须与平均值不同才能被视为异常值。例如，如果您将 3 指定为标准偏差，一个值必须比平均值下降超过 3 个标准差才能被视为异常值。

这些区域有：修复方法是在检测到异常值时处理异常值的方法。可从以下选项中进行选择：

- 剪辑：使用此选项可将异常值剪裁到相应的异常值检测边界。
- Remove：使用它可以从数据框中删除包含异常值的行。
- 失效：使用此选项可以用无效值替换异常值。

标准差数字异常值

此变换使用均值和标准差检测并修复数字要素中的异常值。

您可以指定的数量标准偏差值必须与平均值不同才能被视为异常值。例如，如果您将 3 指定为标准偏差，一个值必须比平均值下降超过 3 个标准差才能被视为异常值。

这些区域有：修复方法是在检测到异常值时处理异常值的方法。可从以下选项中进行选择：

- 剪辑：使用此选项可将异常值剪裁到相应的异常值检测边界。
- Remove：使用它可以从数据框中删除包含异常值的行。
- 失效：使用此选项可以用无效值替换异常值。

分位数数值异常值

使用此变换可以使用分位数检测和修复数字要素中的异常值。您可以定义上分位数和下分位数，分别高于或低于这些分位数值的所有值都被视为异常值。

这些区域有：修复方法是在检测到异常值时处理异常值的方法。可从以下选项中进行选择：

- 剪辑：使用此选项可将异常值剪裁到相应的异常值检测边界。
- Remove：使用它可以从数据框中删除包含异常值的行。
- 失效：使用此选项可以用无效值替换异常值。

最小-最大数字异常值

此转换使用上限和下限阈值检测并修复数字要素中的异常值。如果您知道标记异常值的阈值，请使用此方法。

你指定阈值上限和阈值降低，如果值分别高于或低于这些阈值，则将其视为异常值。

这些区域有：修复方法是在检测到异常值时处理异常值的方法。可从以下选项中进行选择：

- 剪辑：使用此选项可将异常值剪裁到相应的异常值检测边界。
- Remove：使用它可以从数据框中删除包含异常值的行。
- 失效：使用此选项可以用无效值替换异常值。

替换稀有

当您使用替换罕见的转换时，您可以指定阈值，Data Wrangler 会查找满足该阈值的所有值，然后用您指定的字符串替换它们。例如，您可能希望使用此转换将列中的所有异常值分类为 “其他” 类别。

- 替换字符串：用来替换异常值的字符串。
- 绝对阈值：如果实例数量小于或等于此绝对阈值，则类别是罕见的。
- 分数阈值：如果实例数小于或等于此分数阈值乘以行数，则类别是罕见的。
- 最大常见类别：手术后保留的最大不罕见类别。如果阈值没有过滤足够的类别，那么出场次数最多的类别将被归类为不罕见的类别。如果设置为 0（默认值），则对类别数量没有硬限制。

处理缺失值

缺少值是机器学习数据集中常见的情况。在某些情况下，使用计算值（例如平均值或绝对普通值）来计算缺失的数据是合适的。您可以使用处理缺失值转换组。此组包含以下变换。

缺少填充

使用缺少填充转换为替换缺少的值填充值你定义。

归因缺少

使用归因缺少转换以创建一个包含在输入分类和数值数据中找到缺少值的估算值的新列。配置取决于您的数据类型。

对于数字数据，请选择一种估算策略，该策略用于确定要估算的新价值。您可以选择将平均值或中位数与数据集中存在的值相比进行计算。Data Wrangler 使用它计算的值来计算缺失的值。

对于分类数据，Data Wrangler 使用列中最常见的值来计算缺失值。要归因自定义字符串，请使用缺少填充相反，转换。

添加缺失的指示器

使用添加缺失的指示器转换为创建一个包含布尔值的新指标列`"false"`如果行包含值，以及`"true"`如果一行包含缺少值。

丢失

使用丢失选项用于删除包含缺失值的行输入栏.

管理列

您可以使用以下转换快速更新和管理数据集中的列：

名称	函数
删除列	删除列。
复制列	复制列。
重命名列	重命名列。
移至列	移动数据集中列的位置。选择将列移动到数据集的起始或结尾、参考列之前或之后，或移动到特定索引。

管理行

使用此转换组可以快速对行执行排序和随机播放操作。此组包含以下内容：

- 排序：按给定列对整个数据框进行排序。选中 旁边的复选框升序对于此选项；否则，取消选中该复选框并使用降序进行排序。
- 随机排序：随机洗牌数据集中的所有行。

管理向量

使用此变换组可以组合或拼合矢量列。此组包含以下变换。

- 组装：使用此转换可将 Spark 向量和数字数据合并为单个列。例如，您可以合并三列：两列包含数字数据，另一列包含矢量。添加你想要合并的所有列输入列然后指定输出列名称对于合并的数据。
- Flatten：使用此变换可以拼合包含矢量数据的单个列。输入列必须包含 PySpark 向量或类似数组的对象。您可以通过指定一个来控制创建的列数。检测输出数量的方法. 例如，如果选择第一个矢量的长度，列中找到的第一个有效矢量或数组中的元素数量决定了创建的输出列的数量。包含太多项目的所有其他输入向量都将被截断。项目太少的输入充满了 NaNs。

  您还可以指定输出前缀，它用作每个输出列的前缀。

处理数值

使用处理数值要素组来处理数字数据。此组中的每个标量都是使用 Spark 库定义的。支持以下标量：

- 标准缩放器：通过从每个值中减去均值并缩放至单位方差来标准化输入列。要了解更多信息，请参阅 Spark 文档标准缩放器.
- 强大的缩放器：使用对异常值强大的统计信息来缩放输入列。要了解更多信息，请参阅 Spark 文档robustScaler.
- Min Max Scaler：通过将每个要素缩放至给定范围来转换输入列。要了解更多信息，请参阅 Spark 文档minmaxScaler.
- 最大绝对比例：通过将每个值除以最大绝对值来缩放输入列。要了解更多信息，请参阅 Spark 文档maxabScaler.

采样

导入数据后，可以使用采样变压器获取一个或多个样品。当您使用采样变压器时，Data Wrangler 会对原始数据集进行采样。

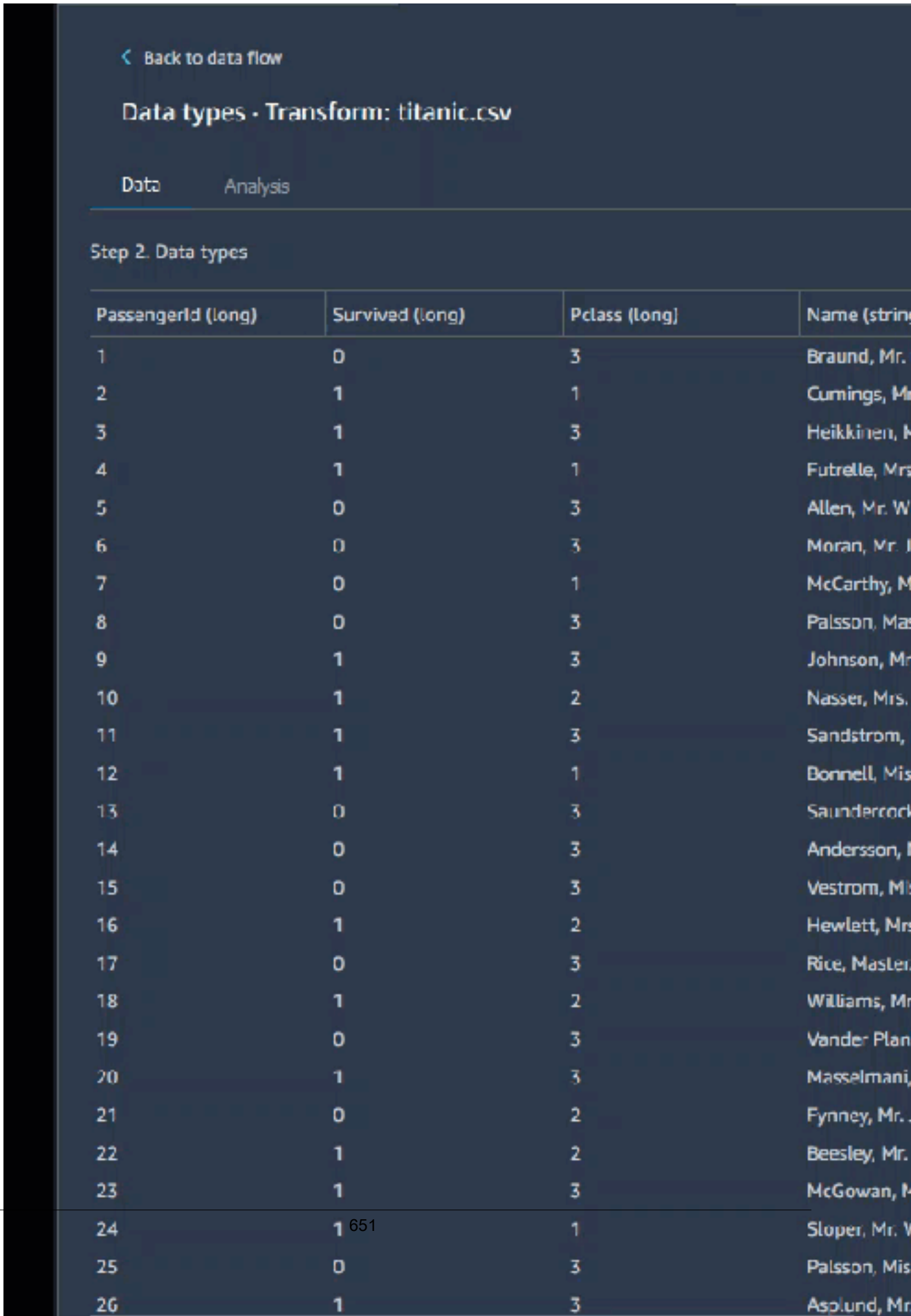
您可以选择以下示例方法之一：

- 限制— 从第一行开始到指定的限制对数据集进行样本。
- 随机化— 获取您指定大小的随机样本。
- 分层— 获取分层随机样本。

您可以对随机样本进行分层，以确保它代表数据集的原始分布。

您可能正在为多个使用案例执行数据准备。对于每个用例，您可以采取不同的样本并应用一组不同的转换。

下面 GIF 显示了一个采取随机化A. 示例和分层数据集中的样本。


Back to data flow
Data types · Transform: titanic.csv
Data
Analysis
Step 2. Data types
PassengerId (long)
Survived (long)
Pclass (long)
Name (strin
1 0 3 Braund, Mr.
2 1 1 Cumings, M
3 1 3 Heikkinen, M
4 1 1 Futrelle, Mr
5 0 3 Allen, Mr. W
6 0 3 Moran, Mr.
7 0 1 McCarthy, M
8 0 3 Palsson, Ma
9 1 3 Johnson, M
10 1 2 Nasser, Mrs.
11 1 3 Sandstrom,
12 1 1 Bonnell, Mis
13 0 3 Saundercock
14 0 3 Andersson,
15 0 3 Vestrom, Mi
16 1 2 Hewlett, Mr
17 0 3 Rice, Master
18 1 2 Williams, Mr
19 0 3 Vander Plan
20 1 3 Masselmani
21 0 2 Fynney, Mr.
22 1 2 Beesley, Mr.
23 1 3 McGowan, M
24 1 1 Sloper, Mr.
25 0 3 Palsson, Mis
26 1 3 Asplund, Mr

以下过程描述了创建随机样本的过程。前面的 GIF 显示了过程中使用的步骤。

从你的数据中随机抽取样本。

1. 选择+位于您导入的数据集的右侧。数据集的名称位于+.
2. 选择添加转换.
3. 选择 Sampling (采样)。
4. 适用于采样方法中，选择采样方法。
5. 适用于大致样本数量中，选择您想要在样本中的观测值的大致数量。
6. （可选）指定一个整数Random seed (随机种子)以创建一个可重复的样品。

以下过程描述了创建分层样本的过程。

从数据中获取分层样本。

1. 选择+位于您导入的数据集的右侧。数据集的名称位于+.
2. 选择添加转换.
3. 选择 Sampling (采样)。
4. 适用于采样方法中，选择采样方法。
5. 适用于大致样本数量中，选择您想要在样本中的观测值的大致数量。
6. 适用于分层列，请指定要分层的列名称。
7. （可选）指定一个整数Random seed (随机种子)以创建一个可重复的样品。

搜索和编辑

使用此部分可以搜索和编辑字符串中的特定模式。例如，您可以查找和更新句子或文档中的字符串、按分隔符拆分字符串以及查找特定字符串的出现次数。

下面支持以下转换搜索和编辑. 所有转换都会返回字符串的副本输入栏然后将结果添加到新的输出列中。

名称	函数
查找子字符串	返回首次出现的索引SUBSTRING你可以选择搜索，开始和结束搜索启动和End分别。
查找子字符串（从右起）	返回最后一次出现的索引SUBSTRING您可以选择地搜索到的开始和结束搜索启动和End分别。
匹配前缀	如果字符串包含给定的，则返回布尔值模式. 模式可以是字符序列或正则表达式。或者，您可以使模式区分大小写。
查找所有事件	返回一个包含给定模式的所有出现情况的数组。模式可以是字符序列或正则表达式。
使用正则表达式提取	返回与给定 Regex 模式匹配的字符串。
在分隔符之间提取	返回一个包含所有字符的字符串左分隔符和右分隔符.
从位置中提取	返回一个字符串，从起始位置在输入字符串中，其中包含直到起始位置的所有字符加长度.

名称	函数
查找并替换子字符串	返回一个包含给定的所有匹配项的字符串模式（正则表达式）替换为替换字符串.
在分隔符之间替换	返回一个字符串，其中包含在第一次出现之间找到的子字符串左分隔符和最后一次出现的右分隔符替换为替换字符串. 如果未找到匹配项，则没有替换任何内容。
从位置替换	返回一个子字符串介于之间的字符串起始位置和起始位置加长度替换为替换字符串. 如果起始位置加长度大于替换字符串的长度，输出包含....
将正则表达式转换为缺少	将字符串转换为`None`如果无效，则返回结果。有效性是用中的正则表达式定义的模式.
按分隔符拆分字符串	从输入字符串中返回一个字符串数组，拆分方式分隔符，最多最大拆分数（可选）。分隔符默认为空格。

拆分数据

使用拆分数据转换以将数据集拆分为两三个数据集。例如，您可以将数据集拆分为用于训练模型的数据集和用于测试模型的数据集。您可以确定进入每个拆分的数据集的比例。例如，如果您将一个数据集拆分为两个数据集，则训练数据集可以有 80% 的数据，而测试数据集具有 20%。

将数据拆分为三个数据集使您能够创建训练、验证和测试数据集。您可以通过删除目标列来查看模型在测试数据集上的表现情况。

您的使用案例决定了每个数据集获取的原始数据集的数量以及用于拆分数据的方法。例如，您可能希望使用分层拆分来确保目标列中的观测值在数据集之间的分布相同。你可以使用以下拆分变换：

- 随机分割 — 每个拆分都是原始数据集的随机、不重叠的样本。对于较大的数据集，使用随机拆分可能会昂贵计算，并且比有序拆分所花费的时间更长。
- 有序拆分 — 根据观测值的顺序拆分数据集。例如，对于 80/20 次训练测试分割，构成数据集 80% 的第一个观测值将转到训练数据集。最后 20% 的观测值用于测试数据集。有序拆分可以有效地保持拆分之间的数据的现有顺序。
- 分层拆分 — 拆分数据集以确保输入列中的观测值数具有比例表示。对于包含观测值 1、1、1、1、1、1、2、2、2、2、2、2、2、2、2、3、3、3、3、3、3、3、3、3、3、3、3、3 的输入列中的 80/20 个分割意味着大约 80% 的 1s、2S 的 80% 和 80% 的 3 进入训练集合。每种观察的大约 20% 进入测试套装。
- 按键拆分 — 避免在多个拆分中发生同一个密钥的数据。例如，如果您有一个包含 “ustomer_id” 列的数据集，并且您将其用作密钥，则没有客户 ID 在多个拆分中。

拆分数据后，可以对每个数据集应用其他转换。对于大多数使用案例，它们没有必要。

Data Wrangler 计算绩效的拆分比例。您可以选择错误阈值来设置拆分的准确性。较低的误差阈值更准确地反映您为拆分指定的比例。如果设置了较高的错误阈值，则可以获得更好的性能，但准确性会降低。

要完全拆分数据，请将错误阈值设置为 0。为了获得更好的性能，您可以指定 0 到 1 之间的阈值。如果您指定的值大于 1，Data Wrangler 会将该值解释为 1。

如果数据集中有 10000 行，并且指定了 80/20 拆分，错误为 0.001，则您将获得大约以下结果之一的观测值：

- 训练集中有 8010 个观测结果和测试集中的 1990 年

- 培训集中有 7990 次观察结果和测试集中的 2010 年观察

上一个示例中测试集的观测值数在 8010 到 7990 之间的间隔内。

默认情况下，Data Wrangler 使用随机种子来使拆分可重现。您可以为种子指定不同的值来创建不同的可重复分割。

Randomized split

要对数据集执行随机拆分，请按照以下过程操作。

要随机拆分数据集，请执行以下操作：

1. 选择+在包含要拆分的数据集的节点旁边。
2. 选择添加转换.
3. 选择拆分数据.
4. （可选）对于Splet中，指定每个拆分的名称和比例。比例必须总和为 1。
5. (可选) 选择+以创建额外的拆分。
   - 指定所有拆分的名称和比例。比例必须总和为 1。
6. (可选) 为错误阈值除了默认值。
7. (可选) 为Random seed (随机种子).
8. 选择预览.
9. 选择 Add（添加）。

Ordered split

要对数据集执行有序拆分，请按照以下过程操作。

要在数据集中进行有序拆分，请执行以下操作。

1. 选择+在包含要拆分的数据集的节点旁边。
2. 选择添加转换.
3. 适用于转换，选择下令拆分.
4. 选择拆分数据.
5. （可选）对于Splet中，指定每个拆分的名称和比例。比例必须总和为 1。
6. (可选) 选择+以创建额外的拆分。
   - 指定所有拆分的名称和比例。比例必须总和为 1。
7. (可选) 为错误阈值除了默认值。
8. （可选）对于输入栏中，指定一个包含数字值的列。使用列的值推断每个拆分中有哪些记录。较小的值在一个拆分中，另一个拆分中的值越大。
9. （可选）选择处理重复项为重复值添加噪声并创建一个包含完全唯一值的数据集。
10. (可选) 为Random seed (随机种子).
11. 选择预览.
12. 选择 Add（添加）。

Stratified split

要对数据集执行分层拆分，请按照以下过程操作。

要在数据集中进行分层拆分，请执行以下操作。

1. 选择+在包含要拆分的数据集的节点旁边。
2. 选择添加转换.
3. 选择拆分数据.
4. 适用于转换，选择分层拆.
5. （可选）对于Splet中，指定每个拆分的名称和比例。比例必须总和为 1。
6. (可选) 选择+以创建额外的拆分。
   - 指定所有拆分的名称和比例。比例必须总和为 1。
7. 适用于输入栏中，指定最多包含 100 个唯一值的列。Data Wrangler 不能对具有超过 100 个唯一值的列进行分层。
8. (可选) 为错误阈值除了默认值。
9. (可选) 为Random seed (随机种子)来指定不同的种子。
10. 选择预览.
11. 选择 Add（添加）。

Split by column keys

可以使用以下过程按照数据集中的列键来拆分。

要按数据集中的列键拆分，请执行以下操作。

1. 选择+在包含要拆分的数据集的节点旁边。
2. 选择添加转换.
3. 选择拆分数据.
4. 适用于转换，选择按键拆分.
5. （可选）对于Splet中，指定每个拆分的名称和比例。比例必须总和为 1。
6. (可选) 选择+以创建额外的拆分。
   - 指定所有拆分的名称和比例。比例必须总和为 1。
7. 适用于键列中，请使用您不希望在两个数据集中显示的值来指定列。
8. (可选) 为错误阈值除了默认值。
9. 选择预览.
10. 选择 Add（添加）。

将值解析为类型

使用此转换可将列转换为新类型。支持的 Data Wrangler 数据类型包括：

- 长整型
- Float
- 布尔值
- 日期，格式为 dd-mm-yyyy，分别表示日、月和年。
- 字符串

验证字符串

使用验证字符串转换以创建一个指示一行文本数据满足指定条件的新列。例如，您可以使用验证字符串转换以验证字符串是否只包含小写字符。下面支持以下转换验证字符串.

此转换组中包括以下转换。如果转换输出布尔值，`True`用一个表示`1`和`False`用一个表示`0`.

名称	函数
字符串长度	返回值`True`如果字符串长度等于指定的长度。否则返回 `False`。
从 开始	返回值`True`如果字符串开始，将有一个指定的前缀。否则返回 `False`。
Ends with	返回值`True`如果字符串长度等于指定的长度。否则返回 `False`。
字母数字吗	返回值`True`如果字符串只包含数字和字母。否则返回 `False`。
是 Alpha（字母）	返回值`True`如果字符串只包含字母。否则返回 `False`。
数字	返回值`True`如果字符串只包含数字。否则返回 `False`。
是空间	返回值`True`如果字符串只包含数字和字母。否则返回 `False`。
是标题	返回值`True`如果字符串包含任何空格。否则返回 `False`。
小写形式	返回值`True`如果字符串只包含小写字母。否则返回 `False`。
大写形式	返回值`True`如果字符串只包含大写字母。否则返回 `False`。
数值	返回值`True`如果字符串只包含数字。否则返回 `False`。
小数	返回值`True`如果字符串只包含十进制数字。否则返回 `False`。

取消嵌套 JSON 数据

如果你有 .csv 文件，你的数据集中可能有 JSON 字符串的值。同样，您可能会将数据嵌套在 Pasic 文件或 JSON 文档的列中。

使用展平结构运算符将第一级键分成单独的列。第一级键是不嵌套在值中的键。

例如，您可能有一个具有人列中每个人的人口统计信息存储为 JSON 字符串。JSON 字符串可能与以下内容类似。

```
"{"seq": 1,"name": {"first": "Nathaniel","last": "Ferguson"},"age": 59,"city":
"Posbotno","state": "WV"}"
```

这些区域有：展平结构运算符会将以下第一级键转换为数据集中的其他列：

- seq

- name
- 年龄
- city
- state

Data Wrangler 将键的值作为值放在列之下。下面显示了 JSON 的列名和值。

```
seq, name,                                    age, city, state
1, {"first": "Nathaniel","last": "Ferguson"}, 59, Posbotno, WV
```

对于数据集中包含 JSON 的每个值，展平结构运算符为第一级键创建列。要为嵌套键创建列，请再次调用运算符。对于前面的示例，调用运算符将创建列：

- name_irst
- name_last

以下示例显示了再次调用操作后产生的数据集。

```
seq, name,                                    age, city, state, name_first, name_last
1, {"first": "Nathaniel","last": "Ferguson"}, 59, Posbotno, WV, Nathaniel, Ferguson
```

选择要拼合的钥匙以指定要作为单独列提取的第一级键。如果您未指定任何密钥，则默认情况下，Data Wrangler 会提取所有密钥。

分解数组

使用分解数组将数组的值扩展为单独的输出行。例如，该操作可以获取数组中的每个值 [[1、2、3、]、[4、5、6]、[7、8、9]] 并创建一个包含以下行的新列：

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

数据 Wrangler 将新列命名为 input_column_name_flatten。

您可以拨打分解数组多次操作以将数组的嵌套值放入单独的输出列中。以下示例显示了对具有嵌套数组的数据集多次调用该操作的结果。

将嵌套数组的值放到单独的列中

id	数组	id	array_items	id	array_items_项目
1	[[猫、狗]、[蝙蝠、青蛙]]	1	[猫、狗]	1	cat
2	[[玫瑰，矮牵牛]，[莉莉，雏菊]]	1	[蝙蝠，青蛙]	1	狗

id	数组	id	array_items	id	array_items_项目
		2	[玫瑰，矮牵牛]	1	bat
		2	[莉莉，雏菊]	1	青蛙
			2	2	玫瑰
			2	2	矮牵牛
			2	2	莉莉
			2	2	雏菊

可视化

亚马逊 SageMaker Data Wrangler 包括内置分析，可帮助您点击几下鼠标即可生成可视化和数据分析。您还可以使用自己的代码创建自定义分析。

通过在数据流中选择一个步骤然后选择，可以将分析添加到数据框中添加分析. 要访问您创建的分析，请选择包含该分析的步骤，然后选择分析。

所有分析都是使用 10 万行数据集生成的。

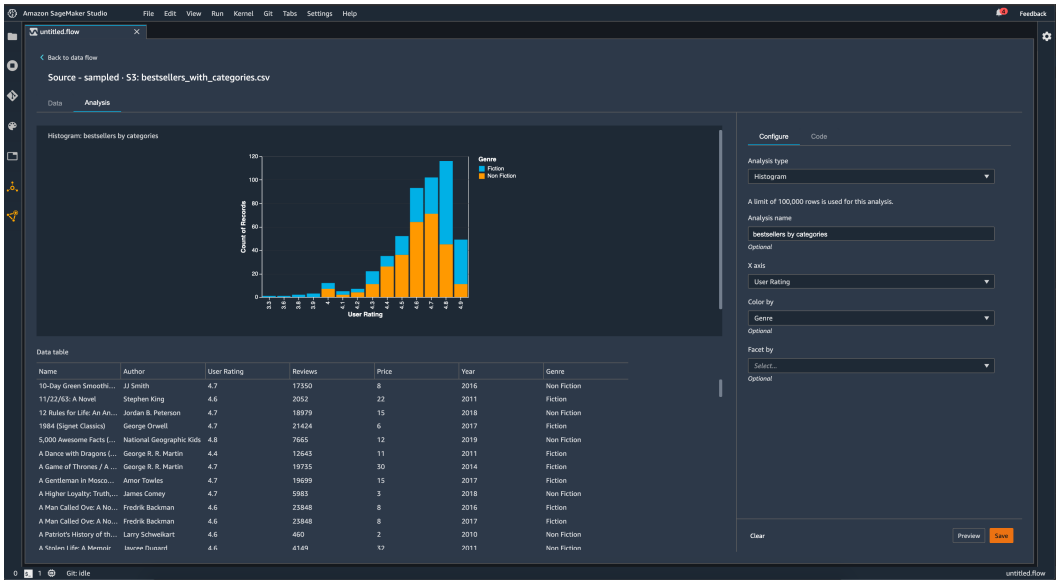
您可以将以下分析添加到数据框中：

- 数据可视化，包括直方图和散点图。
- 数据集的快速摘要，包括条目数、最小值和最大值（对于数字数据）以及最频繁和最少的类别（对于分类数据）。
- 数据集的快速模型，可用于为每个要素生成重要性评分。
- 目标泄漏报告，您可以使用该报告来确定一个或多个要素是否与目标要素密切关联。
- 使用自己的代码进行自定义可视化。

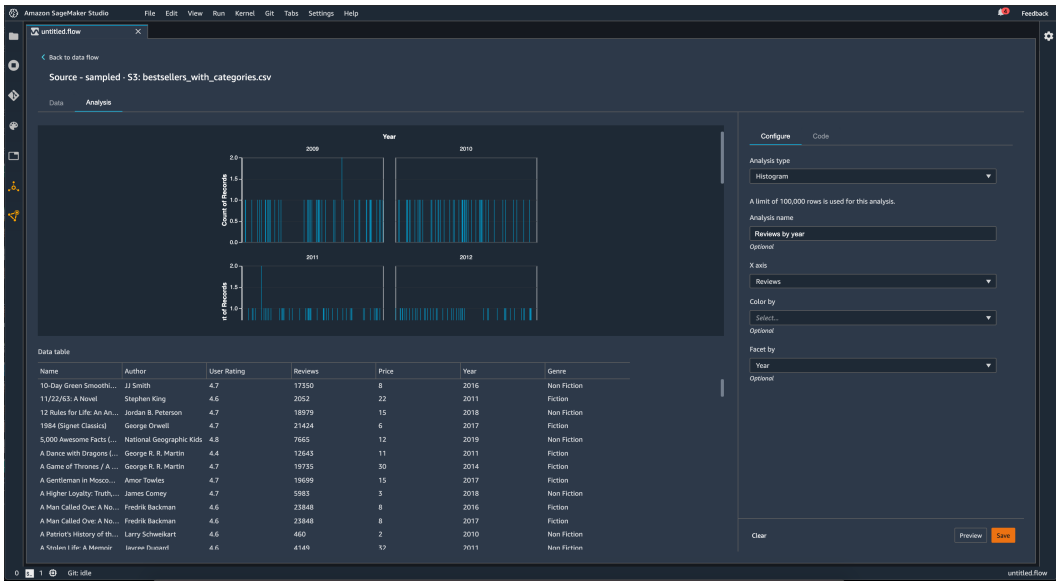
有关这些选项的更多信息，请参阅以下部分。

直方图

使用直方图查看特定要素的要素值计数。您可以使用按颜色选项。例如，以下直方图显示了 2009-2019 年亚马逊最畅销书的用户评分分布情况，按类型着色。



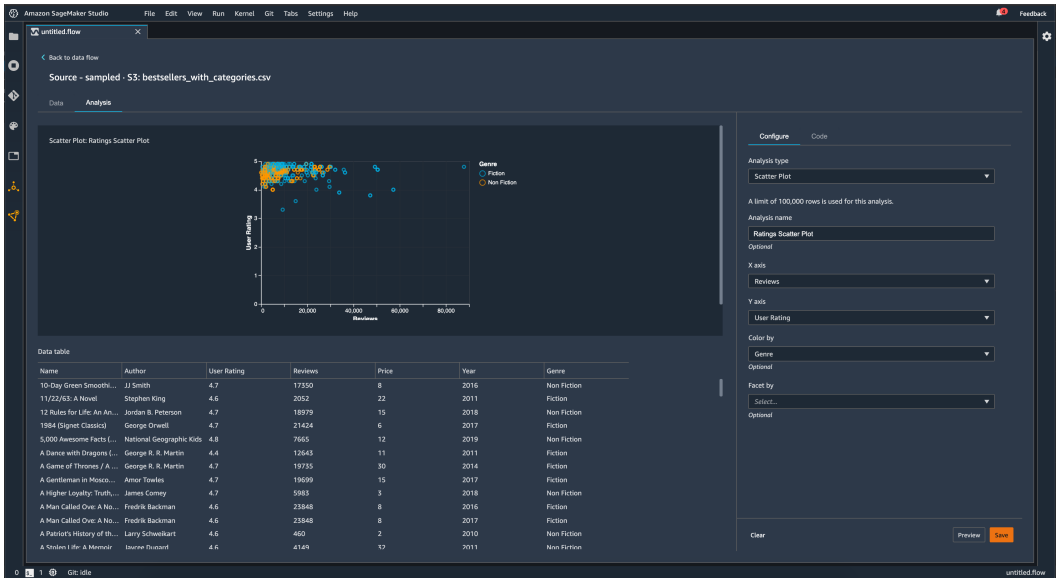
您可以使用分面由功能，用于为另一列中的每个值创建一列的直方图。例如，下图显示了亚马逊上最畅销图书的用户评论的直方图（如果按年分面）。



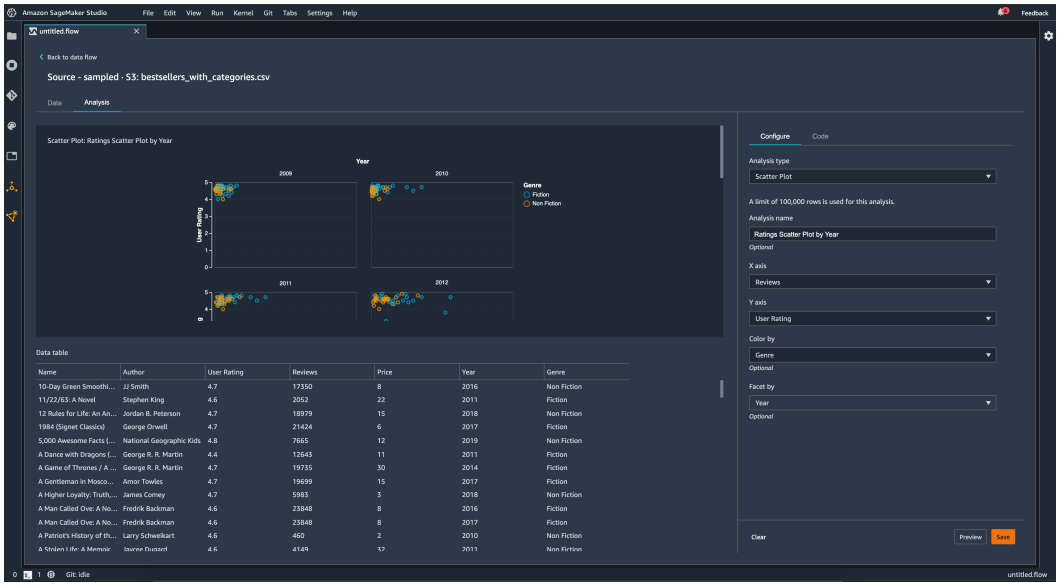
散点图

使用散点图功能来检查要素之间的关系。要创建散点图，请选择要在X 轴和Y 轴. 这两列都必须是数字类型的列。

您可以通过额外的列为散点图着色。例如，以下示例显示了一个散点图，将 2009 年至 2019 年期间亚马逊上畅销书籍的用户评分对比评论数量。散点图是按照书籍类型着色的。



此外，您可以按要素分面散点图。例如，下图显示了按年分面的同一评论与用户评级散点图的示例。



表摘要

使用表摘要进行分析以快速总结数据。

对于包含数值数据（包括日志和浮点数据）的列，表摘要会报告每列的条目数（计数）、最小值（最小值）、最大值（最大值）、均值和标准差（stddev）。

对于包含非数字数据的列，包括具有字符串、布尔值或日期/时间数据的列，表摘要将报告条目数（计数）、最低频率值（最小值）和最常见值（最大值）。

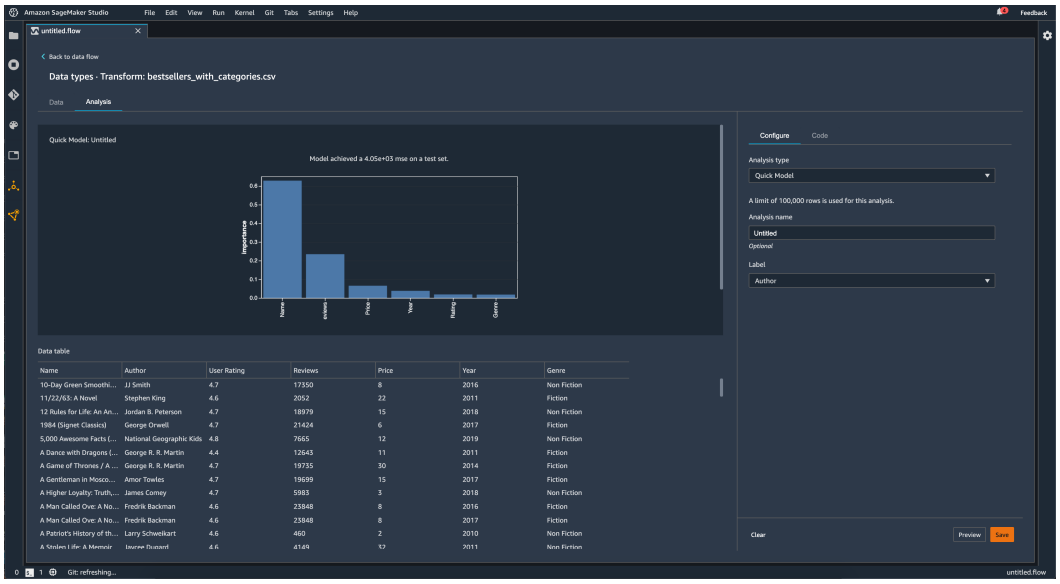
快速模型

使用快速模型可视化以快速评估数据并为每个功能生成重要性分数。一个功能重要性分数分数表示功能在预测目标标签方面的有用性。要素重要性分数介于 [0, 1] 之间，较高的数字表示要素对整个数据集更重要。在快速模型图表的顶部，有一个模型分数。分类问题显示 F1 分数。回归问题有平方误差 (MSE) 分数。

创建快速模型图表时，您可以选择要评估的数据集以及要比较要素重要性的目标标签。数据 Wrangler 将执行以下操作：

- 推断目标标签的数据类型以及所选数据集中的每个要素的数据类型。
- 确定问题类型。根据标签列中不同值的数量，Data Wrangler 确定这是回归还是分类问题类型。数据 Wrangler 将分类阈值设置为 100。如果标签列中有 100 多个不同的值，Data Wrangler 将其归类为回归问题；否则，它被归类为分类问题。
- 预处理要素和标签数据以进行训练。使用的算法要求将要素编码为矢量类型，并将标签编码为双精类型。
- 使用 70% 的数据训练随机森林算法。Spark 的随机森林回归用于训练回归问题的模型。这些区域有：随机森林分类器用于训练分类问题的模型。
- 评估其余 30% 数据的随机森林模型。Data Wrangler 使用 F1 分数评估分类模型，并使用 MSE 分数评估回归模型。
- 使用 Gini 重要性方法计算每个要素的要素重要性。

下图显示了快速模型功能的用户界面。



目标泄漏

当机器学习训练数据集中存在与目标标签密切相关但在真实世界数据中不可用的数据时，就会发生目标泄漏。例如，您的数据集中可能有一列用作要使用模型预测的列的代理。

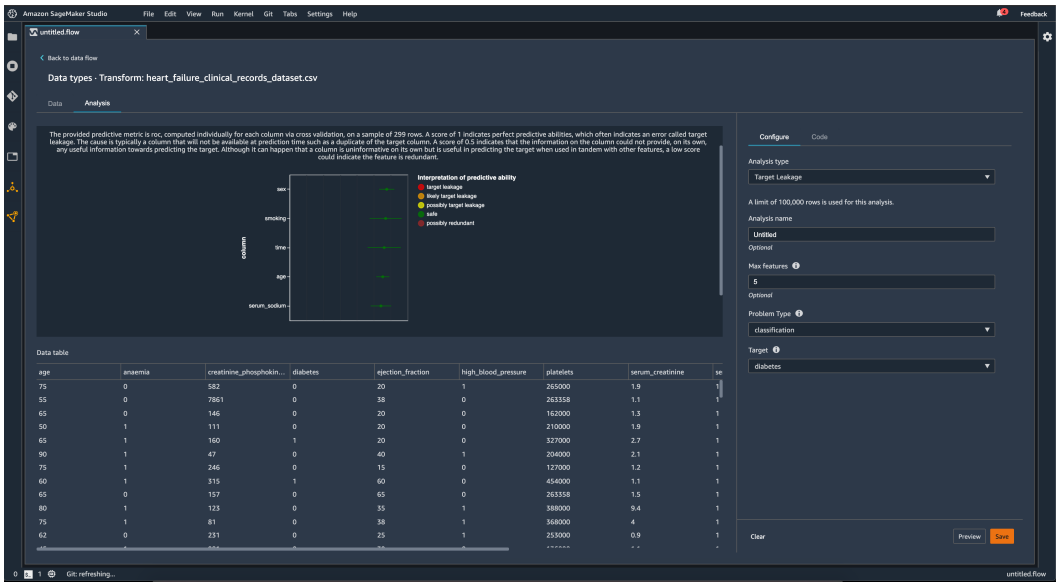
当您使用目标泄漏分析时，您可以指定以下内容：

- 目标：这是您希望机器学习模型能够对其进行预测的功能。
- 问题类型：这是你正在处理的 ML 问题类型。问题类型可以是分类要么回归.
- （可选）最大功能：这是可视化中要显示的最大要素数量，它显示了按目标泄漏风险排序的要素。

对于分类，目标泄漏分析使用接收机操作特征下的面积，或每列 AUC-ROC 曲线，最多可达最大功能. 对于回归，它使用确定系数或 R2 指标。

AUC-ROC 曲线针对多达 1000 行的样本提供了一个预测指标，使用交叉验证为每列单独计算。分数为 1 表示完美的预测能力，这通常表示目标泄漏。分数为 0.5 或更低，表明该栏中的信息本身无法提供预测目标的任何有用信息。尽管可能发生一列本身没有信息性，但在与其他功能同时使用时对预测目标很有用，但较低的分数可能表明该功能是多余的。

例如，下图显示了糖尿病分类问题的目标泄漏报告，即预测一个人是否患有糖尿病。AUC-ROC 曲线用于计算五个要素的预测能力，并且所有要素都被确定为免受目标泄漏的影响。



检测时间序列数据中的异常

您可以使用异常检测可视化来查看时间序列数据中的异常值。要了解什么决定了异常情况，你需要明白我们将时间序列分解为一个预测的术语和错误术语。我们将时间序列的季节性和趋势视为预测期限。我们将残差视为错误术语。

对于误差项，您可以指定阈值，作为残差可以远离平均值的偏差标准数，以便将其视为异常。例如，您可以将阈值指定为 3 个标准差。任何离平均值大于 3 个标准差的残差都是异常现象。

您可以使用以下过程执行异常检测分析。

1. 打开数据 Wrangler 数据流。
2. 在你的数据流中，在数据类型，选择+，然后选择添加分析.
3. 适用于分析类型，选择时间序列.
4. 适用于可视化，选择异常检测.
5. 适用于异常阈值中，选择一个值被视为异常的阈值。
6. 选择预览以生成分析的预览。
7. 选择Add将转换添加到 Data Wrangler 数据流中。

时间序列数据中的季节性趋势分解

您可以使用季节性趋势分解可视化来确定时间序列数据中是否存在季节性。我们使用 STL（使用黄土进行季节性趋势分解）方法来执行分解。我们将时间序列分解为季节性、趋势和残余组成部分。这一趋势反映了该系列系列的长期发展。季节性组成部分是一个在一段时间内再次出现的信号。在从时间序列中删除趋势和季节性成分之后，您就有剩余值。

您可以使用以下过程执行季节趋势分解分析。

1. 打开数据 Wrangler 数据流。
2. 在你的数据流中，在数据类型，选择+，然后选择添加分析.
3. 适用于分析类型，选择时间序列.

4. 适用于可视化，选择季节趋势分解.
5. 适用于异常阈值中，选择一个值被视为异常的阈值。
6. 选择预览以生成分析的预览。
7. 选择Add将转换添加到 Data Wrangler 数据流中。

偏置报告

您可以使用 Data Wrangler 中的偏差报告来发现数据中的潜在偏见。要生成偏见报告，必须指定目标列，或标签，您要预测的以及分面或者您要检查偏置的列。

标签：您希望模型对其进行预测的功能。例如，如果您预测客户转化率，则可以选择一个列，其中包含客户是否已下订单的数据。还必须指定此功能是标签还是阈值。如果指定标签，则必须指定积极结果看起来像你的数据中。在买家转化示例中，积极结果可能是订单列中的 1，表示买家在过去三个月内下订单的积极结果。如果指定阈值，则必须指定定定义正面结果的下限。例如，如果您的客户订单列包含去年下达的订单数量，则可能需要指定 1。

分面：您要检查偏置的列。例如，如果您试图预测客户转化率，您的方面可能是客户的年龄。你可能选择这个方面是因为你认为自己的数据偏向某个年龄组。您必须确定小平面是以值还是阈值来衡量。例如，如果你想检查一个或多个特定的年龄，你可以选择值并指定那些年龄。如果您要查看年龄组，则选择Threshold并指定要检查的年龄阈值。

选择要素和标签后，您可以选择要计算的偏差指标类型。

要了解更多信息，请参阅为训练前数据中的偏差生成报告.

创建自定义可视化

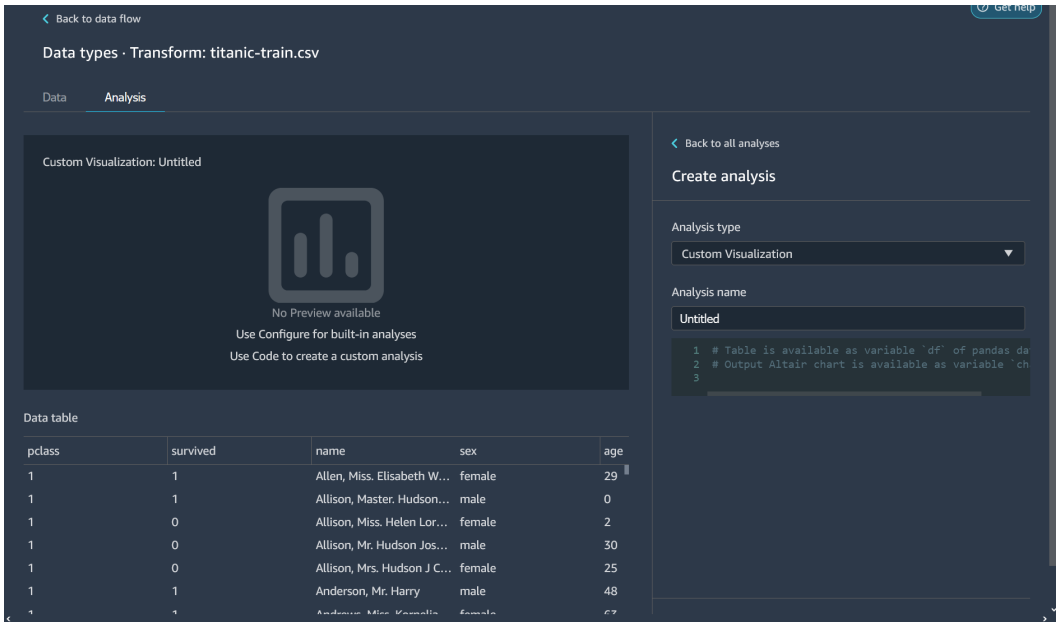
您可以将分析添加到 Data Wrangler 流程以创建自定义可视化。您的数据集以及已应用的所有转换，都可以作为熊猫 DataFrame. 数据 Wrangler 使用`df`变量来存储数据框。您可以通过调用变量来访问数据框。

您必须提供输出变量，`chart`，存储Altair输出图表。例如，您可以使用以下代码块使用泰坦尼克号数据集创建自定义直方图。

```
import altair as alt
df = df.iloc[:30]
df = df.rename(columns={"Age": "value"})
df = df.assign(count=df.groupby('value').value.transform('count'))
df = df[["value", "count"]]
base = alt.Chart(df)
bar = base.mark_bar().encode(x=alt.X('value', bin=True, axis=None), y=alt.Y('count'))
rule = base.mark_rule(color='red').encode(
    x='mean(value):Q',
    size=alt.value(5))
chart = bar + rule
```

要创建自定义可视化：

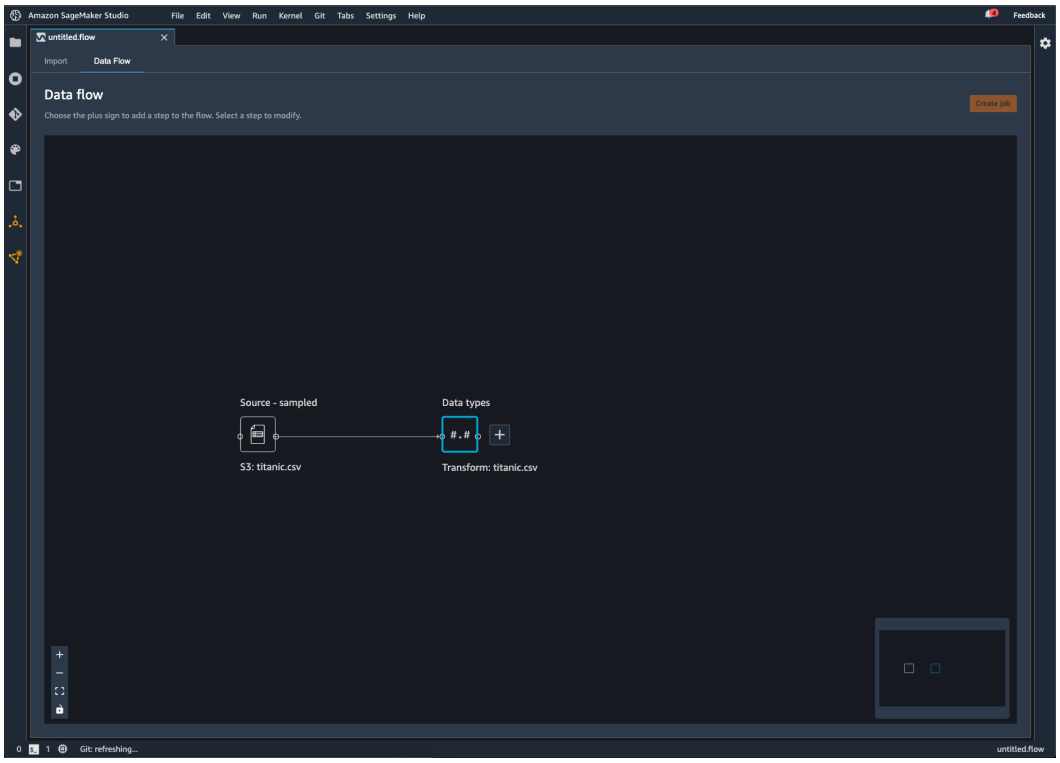
1. 在包含您想要可视化的转换的节点旁边，选择+.
2. 选择添加分析.
3. 适用于分析类型，选择定制可视化.
4. 适用于分析名称中指定名称。
5. 在代码框中输入您的代码。
6. 选择预览以预览可视化。
7. 选择Save（保存）来添加可视化。



Export

在 Data Wrangler 流程中，您可以导出对数据处理管道所做的部分或全部转换。

一个数据 Wangler 流是您对数据执行的一系列数据准备步骤。在数据准备过程中，您可以对数据执行一次或多个转换。每个转换都是使用转换步骤完成的。该流程有一系列节点，用于表示数据的导入和您执行的转换。有关节点示例，请参阅下图。



上图显示了具有两个节点的 Data Wrangler 流。这些区域有：源-取样节点显示了从中导入数据的数据源。这些区域有：数据类型节点表示 Data Wrangler 已执行转换以将数据集转换为可用格式。

添加到 Data Wrangler 流程的每个转换都显示为附加节点。有关您可以添加的转换的信息，请参阅转换数据 (p. 623). 下图显示了数据 Wrangler 流程，该流程具有重命名列节点来更改数据集中列的名称。

您可以将数据转换导出为以下内容：

- Amazon S3
- SageMaker 管道
- 亚马逊 SageMaker Feature Store
- Python 代码

> **Important**
>
> 建议使用 IAM`AmazonSageMakerFullAccess`要授予的托管策略Amazon使用数据 Wrangler 的权限。如果您不使用托管策略，您可以使用 IAM 策略，该策略授予 Data Wrangler 访问 Amazon S3 存储桶的访问权限。有关策略的更多信息，请参阅安全和权限 (p. 676).

导出数据流时，您需要为Amazon您使用的资源。您可以使用成本分配标签组织和管理这些资源的成本。您可以为用户配置文件创建这些标签，Data Wrangler 会自动将它们应用于导出数据流的资源。有关更多信息，请参阅使用成本分配标签。

导出数据转换的选项取决于将数据转换导出到的位置。有关更多信息，请参阅以下部分。

导出到 Amazon S3。

使用 Data Wrangler，您可以将数据导出到 Amazon S3 存储桶中的某个位置。您可以使用以下方法之一指定位置：

- 目标节点 — Data Wrangler 在处理完数据后存储数据的位置。
- 导出到 — 导出转换为 Amazon S3 所产生的数据。
- 导出数据 — 对于小型数据集，可以快速导出已转换的数据。

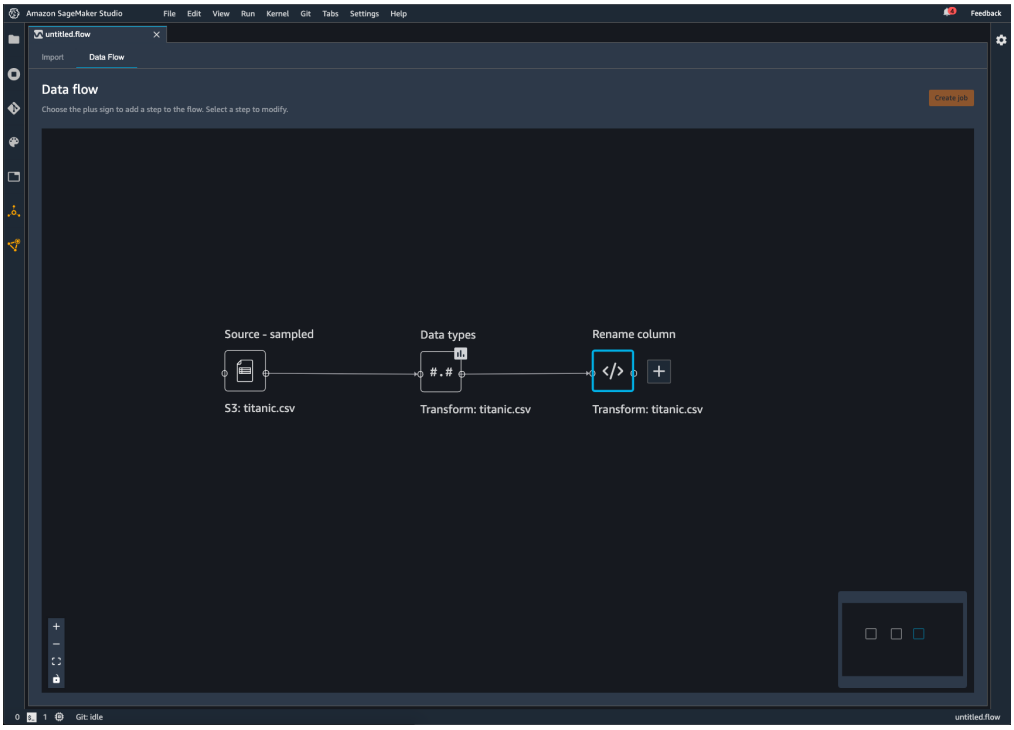
使用以下部分了解有关每种方法的更多信息。

Destination Node

如果要输出向 Amazon S3 执行的一系列数据处理步骤，请创建一个目标节点。处理完数据后，目标节点会告诉 Data Wrangler 在哪里存储数据。创建目标节点后，您可以创建处理作业以输出数据。处理任务是亚马逊 SageMaker 处理作业。当您使用目标节点时，它会运行输出您转换为 Amazon S3 的数据所需的计算资源。

您可以使用目标节点导出部分转换或您在 Data Wrangler 流程中所做的所有转换。

您可以使用多个目标节点来导出不同的转换或转换集。以下示例显示了单个 Data Wrangler 流中的两个目标节点。

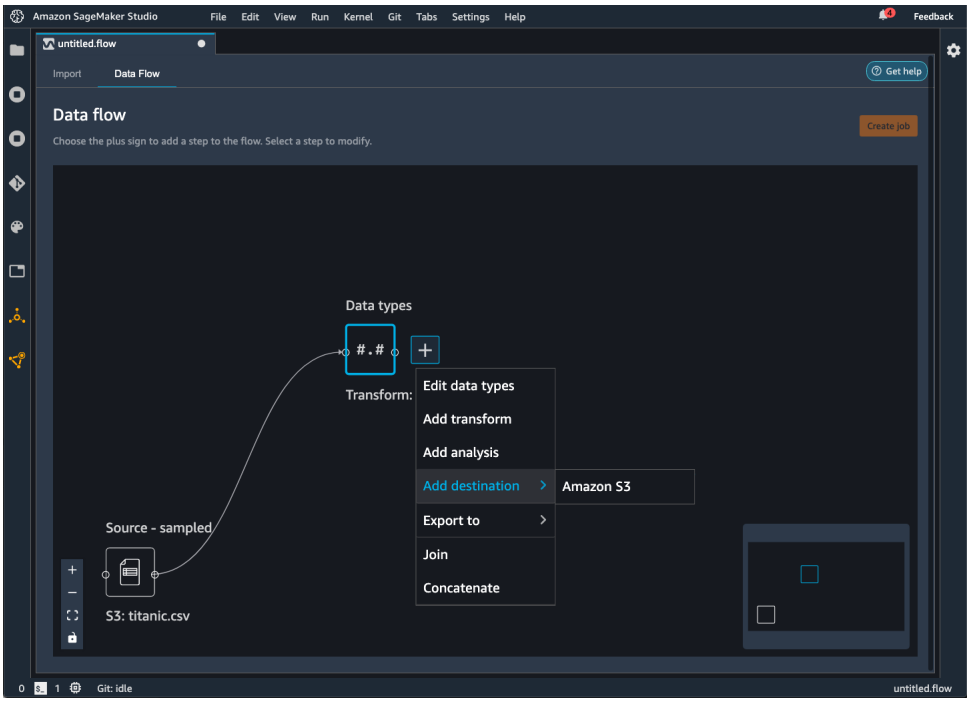


您可以使用以下过程创建目标节点并将其导出到 Amazon S3 存储桶。

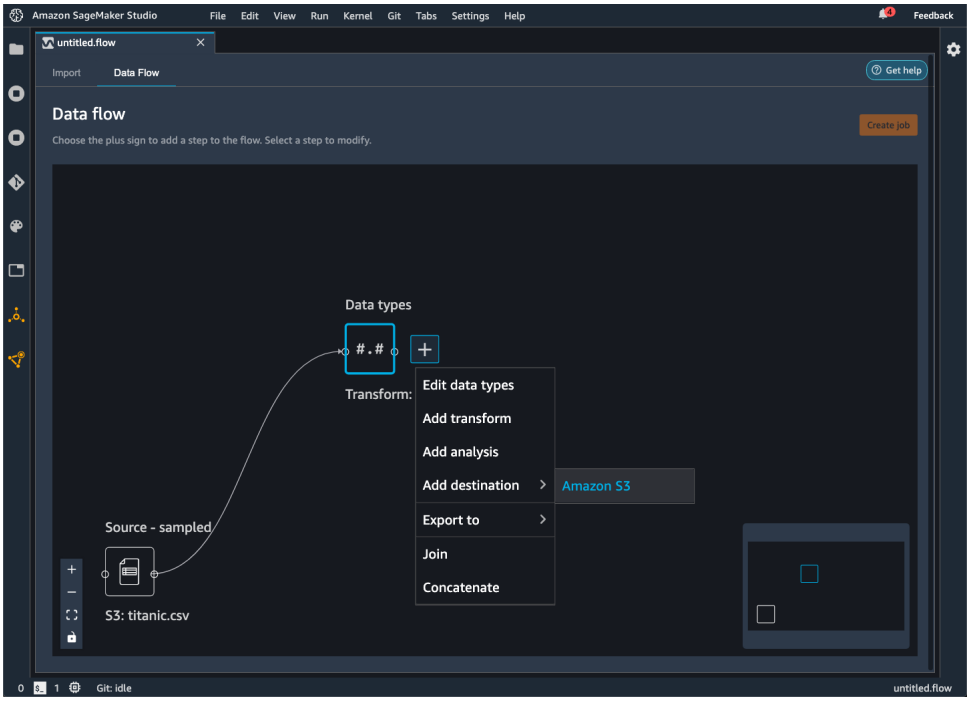
要导出数据流，您需要创建目标节点和 Data Wrangler 作业来导出数据。创建数据 Wrangler 作业开始 SageMaker 处理作业以导出流程。创建目标节点后，您可以选择要导出的目标节点。

要创建目标节点，请按照以下过程操作。

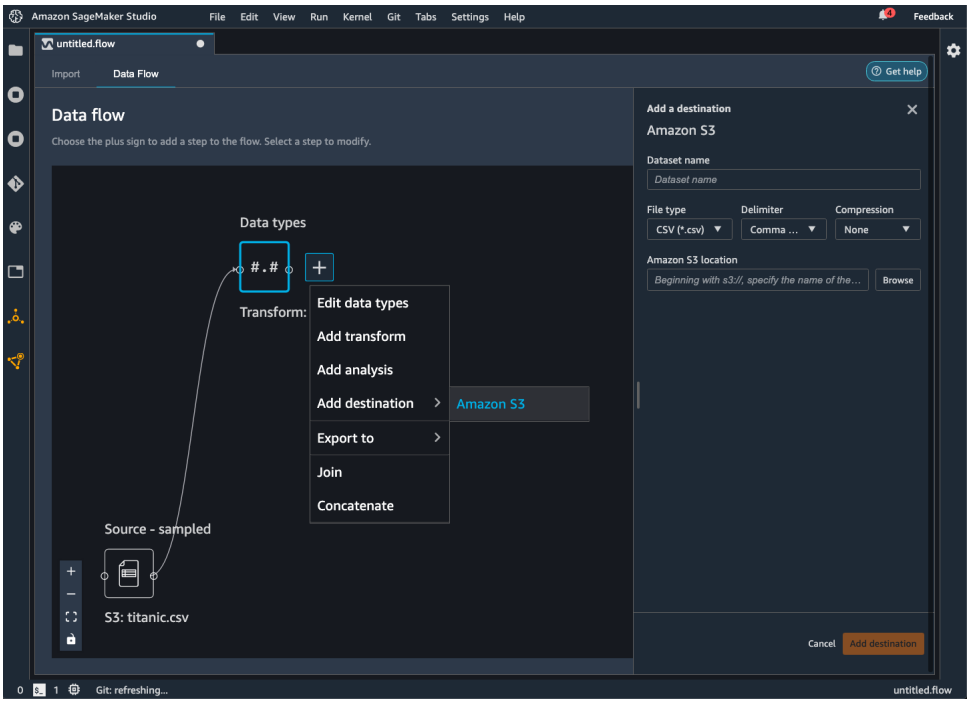
1. 选择+在表示要导出的转换的节点旁边。
2. 选择 Add destination。



3. 选择 Amazon S3。



4. 指定下图中显示的字段。

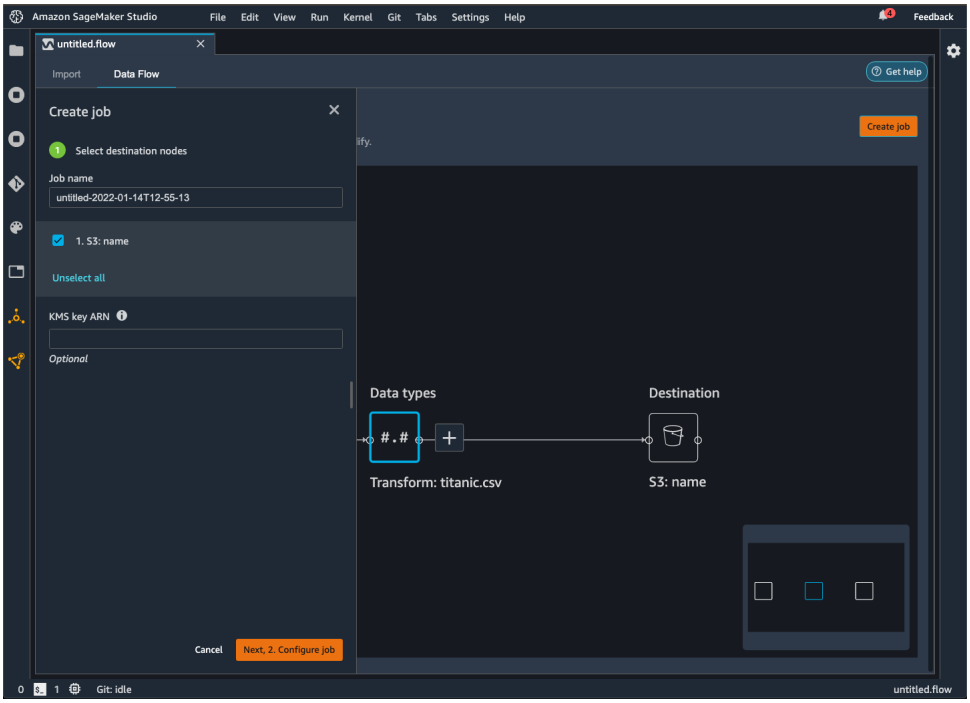


5. 选择 Add destination。

使用以下过程创建 Data Wrangler 作业。

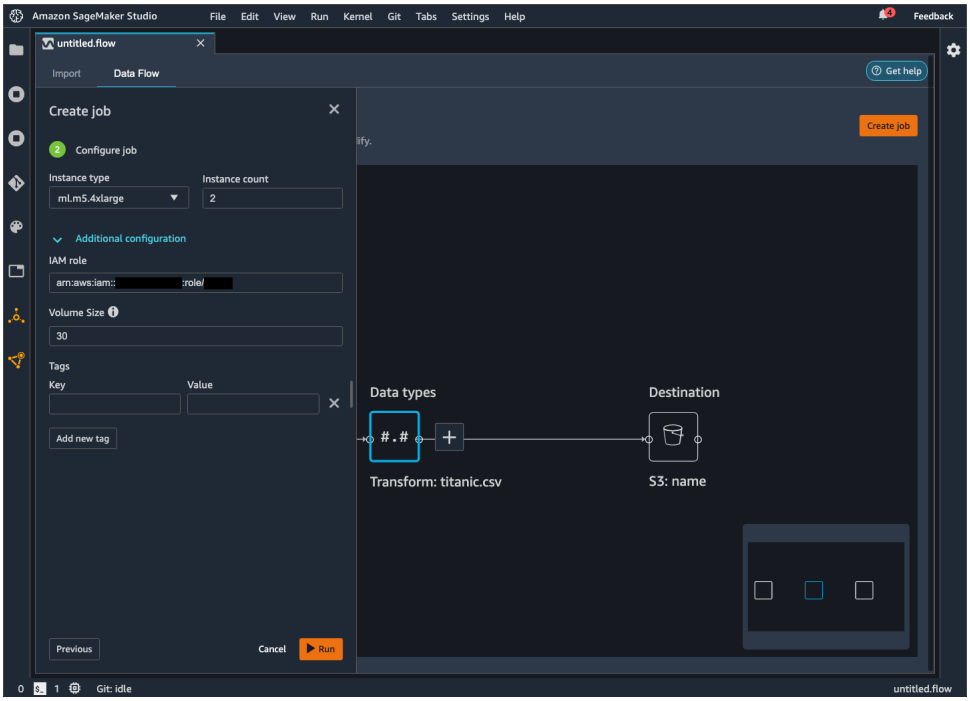
从中创建作业数据流页面，然后选择要导出的目标节点。

1. 请选择 Create job (创建任务)。下图显示了选择后显示的窗格。创建作业.



2. 适用于作业名称，请指定导出作业的名称。

3. 选择要导出的目标节点。
4. （可选）指定Amazon KMS密钥 ARN。一个Amazon KMS密钥是您可以用于保护数据的加密密钥。有关 的更多信息Amazon KMS钥匙，请参阅Amazon Key Management Service
5. 选择配置作业. 下图显示了配置作业页.



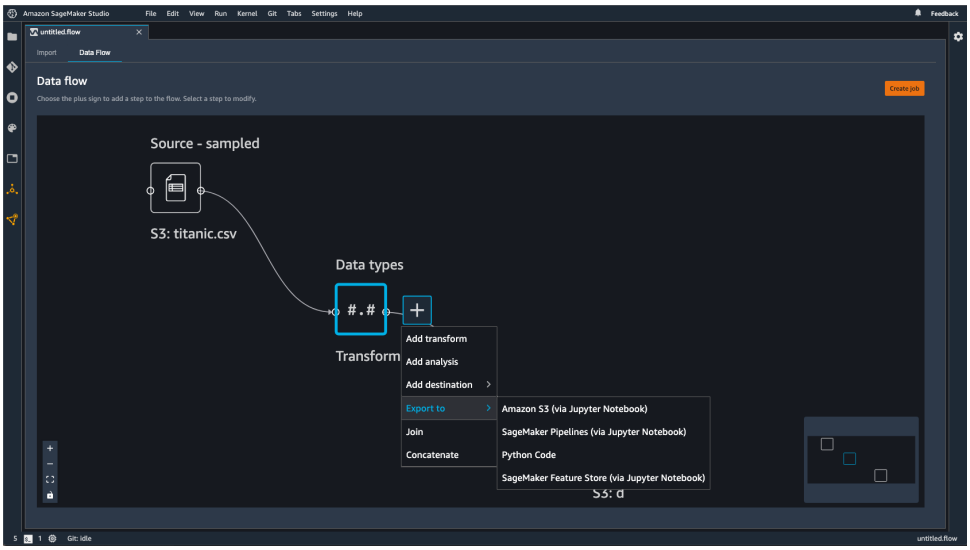
6. （可选）配置 Data Wrangler 作业。
7. 选择 Run（运行）。

Export to

作为使用目标节点的替代方案，您可以使用导出到选项可以使用 Jupyter 笔记本将您的数据 Wrangler 流导出到 Amazon S3。您可以在 Data Wrangler 流程中选择任何数据节点并将其导出。导出数据节点将导出节点所表示的转换以及之前的转换。

使用以下过程生成一个 Jupyter 笔记本并运行它以将您的 Data Wrangler 流导出到 Amazon S3。

1. 选择+在您要导出的节点旁边。
2. 选择导出到.
3. 选择Amazon S3（通过 Jupyter 笔记本）.
4. 运行 Jupyter 笔记本。



当你运行笔记本时，它会将数据流（.flow 文件）导出在同一个Amazon区域作为数据 Wrangler 流。

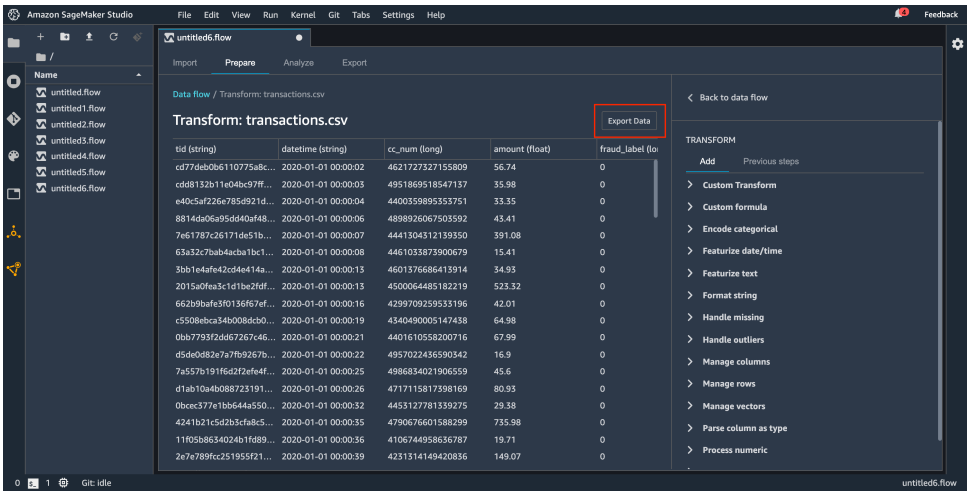
Export data

如果对要快速导出的小数据集进行了转换，则可以使用导出数据方法。在您开始时选择导出数据，Data Wrangler 同步工作以将您转换为 Amazon S3 的数据导出。在数据 Wrangler 完成导出数据或取消操作之前，您无法使用 Data Wrangler。

有关使用导出数据在 Data Wrangler 流程中，请参阅以下过程。

使用导出数据方法。

1. 通过双击 Data Wrangler 流程中选择一个节点。



2. 配置要导出数据的方式。
3. 选择导出数据.

当您将数据流导出到 Amazon S3 存储桶时，Data Wrangler 会在 S3 存储桶中存储流文件的副本。它将流程文件存储在data_wrangler_flowprefix. 如果您使用默认的 Amazon S3 存储桶存储流文件，则它们将使用以下命名约定：`sagemaker-region-account number`. 例如，如果您的帐号为 111122223333，并且您在 us-east-1 中使用 Studio，则导入的数据集将存储在`sagemaker-us-east-1-111122223333`.

在此示例中，在 us-east-1 中创建的 .flow 文件存储在`s3://sagemaker-region-account number/data_wrangler_flows/`.

导出到 SageMaker 管道

当你想构建和部署大规模机器学习 (ML) 工作流时，可以使用 SageMaker 要创建的管道 end-to-end 管理和部署的工作流 SageMaker 个作业。 SageMaker Pipeline 使您能够构建管理您的工作流程 SageMaker 数据准备、模型训练和模型部署作业。你可以使用第一方算法 SageMaker 使用优惠 SageMaker 管道。有关 SageMaker 管道，请参阅SageMaker 管道.

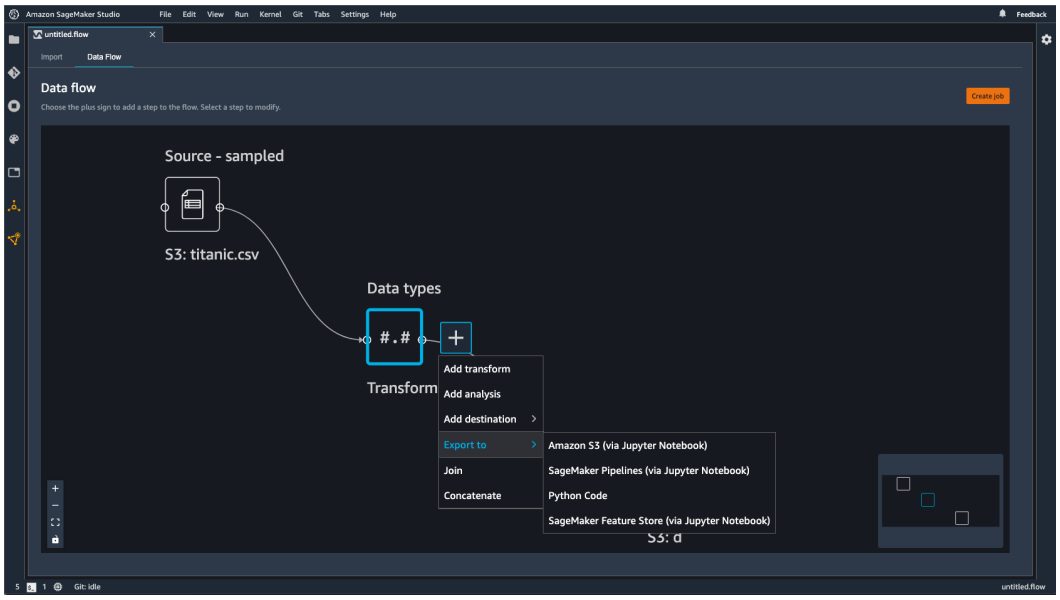
将数据流中的一个或多个步骤导出到 SageMaker 管道，Data Wrangler 创建了一个 Jupyter 笔记本，您可以用它来定义、实例化、运行和管理管道。

使用 Jupyter 笔记本创建管道

使用以下过程创建 Jupyter 笔记本，以将数据 Wrangler 流程导出到 SageMaker 管道。

使用以下过程生成一个 Jupyter 笔记本并运行它以将您的 Data Wrangler 流程导出到 SageMaker 管道。

1. 选择+在您要导出的节点旁边。
2. 选择导出到.
3. 选择SageMaker 管道（通过 Jupyter 笔记本）.
4. 运行 Jupyter 笔记本。



Data Wrangler 生产的木星笔记本可以用来定义管道。管道包括数据 Wrangler 流程定义的数据处理步骤。

您可以通过将步骤添加到管道中添加其他步骤`steps`在笔记本中的以下代码中列出：

```
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[instance_type, instance_count],
    steps=[step_process], #Add more steps to this list to run in your Pipeline
)
```

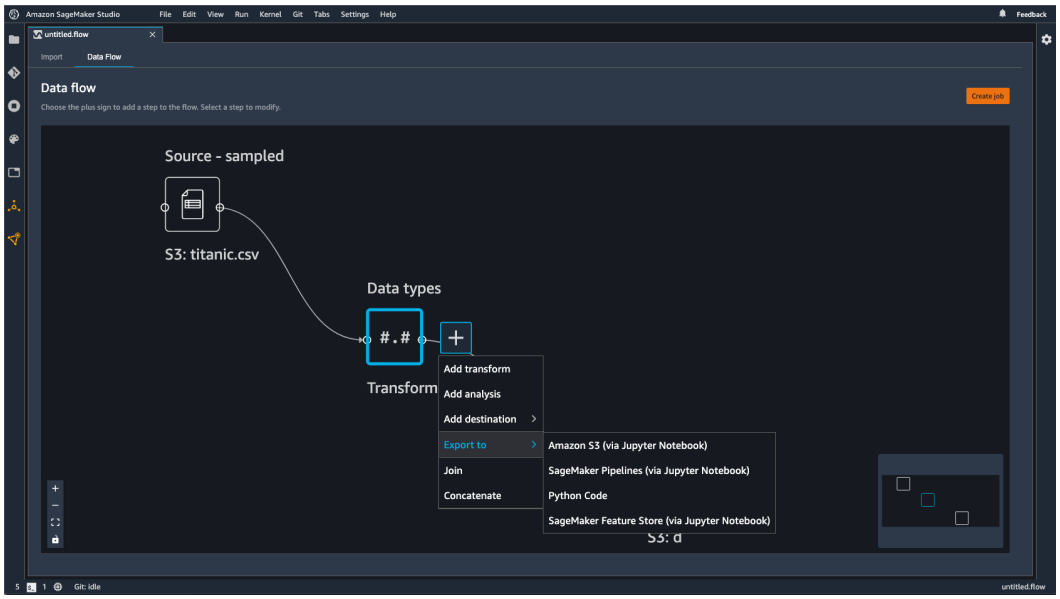
有关定义管道的更多信息，请参阅定义 SageMaker 管道.

导出到 Python 代码

要将数据流中的所有步骤导出到可以手动集成到任何数据处理工作流程中的 Python 文件，请使用以下步骤。

使用以下过程生成一个 Jupyter 笔记本并运行它以将数据 Wrangler 流程导出为 Python 代码。

1. 选择+在您要导出的节点旁边。
2. 选择导出到.
3. 选择Python 代码.
4. 运行 Jupyter 笔记本。



您可能需要配置 Python 脚本以使其在管道中运行。例如，如果您运行的是 Spark 环境，请确保从有权访问的环境中运行脚本Amazon资源的费用。

导出到亚马逊 SageMaker Feature Store

您可以使用 Data Wrangler 将您创建的功能导出到亚马逊 SageMaker Feature Store。要素是数据集中的一列。要素商店是要素及其关联元数据的集中式商店。您可以使用功能商店创建、共享和管理用于机器学习 (ML) 开发的精选数据。集中式存储使您的数据更易于发现和可重复使用。有关功能商店的更多信息，请参阅亚马逊 SageMaker Feature Store.

功能商店的核心概念是功能组。要素组是要素、其记录（观测值）和相关元数据的集合。它与数据库中的表类似。

您可以使用 Data Wrangler 执行以下操作之一：

- 使用新记录更新现有要素组。记录是数据集中的观测值。
- 从 Data Wrangler 流程中的节点创建新功能组。Data Wrangler 会将数据集中的观测值添加为要素组中的记录。

如果要更新现有要素组，则数据集的方案必须与要素组的方案匹配。要素组中的所有记录都将替换为数据集中的观测值。

您可以使用 Jupyter Notebook 或目标节点来使用数据集中的观测值更新要素组。

Destination Node

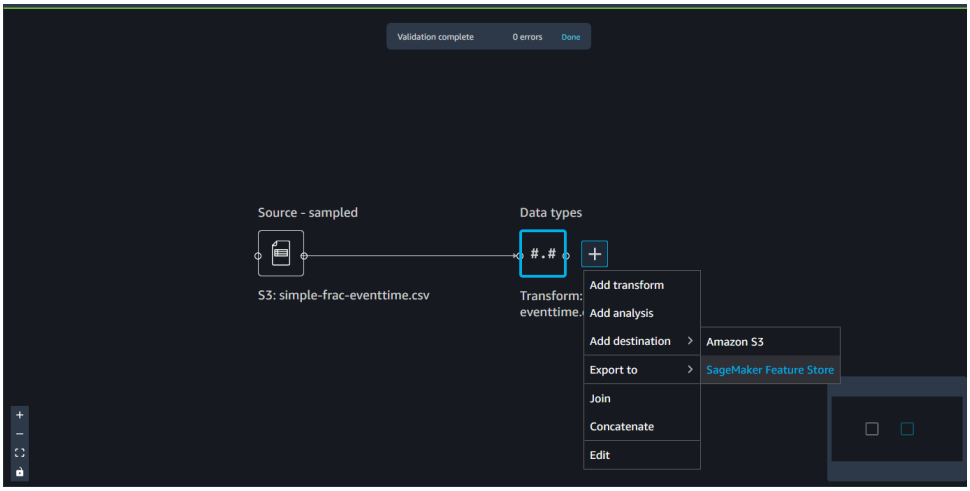
如果要将已执行的一系列数据处理步骤输出到要素组，则可以创建目标节点。当您创建和运行目标节点时，Data Wrangler 会使用您的数据更新要素组。您还可以从目标节点 UI 创建新功能组。创建目标节点后，您可以创建处理作业以输出数据。处理任务是亚马逊 SageMaker 处理作业。当您使用目标节点时，它会运行输出已转换到要素组的数据所需的计算资源。

您可以使用目标节点导出部分转换或您在 Data Wrangler 流程中所做的所有转换。

使用以下过程创建目标节点，以使用数据集中的观测值更新要素组。

要使用目标节点更新要素组，请执行以下操作。

1. 选择+包含要导出的数据集的节点旁边的符号。
2. 在添加目标，选择SageMaker Feature Store.



3. 双击功能组。Data Wrangler 检查要素组的架构是否与您用于更新要素组的数据架构匹配。
4. （可选）选择仅导出到离线商店适用于同时拥有在线商店和离线商店的功能组。此选项仅使用来自数据集的观测值更新离线商店。
5. Data Wrangler 验证数据集的架构后，选择Add.

使用以下过程创建新的要素组，其中包含数据集中的数据。

对于要存储要素组的方式，您可以有以下选项：

- 在线 — 支持实时查找记录的功能组的低延迟、高可用性缓存。在线商店允许快速访问功能组中记录的最新值。
- 离线 — 将功能组的数据存储在 Amazon S3 存储桶中。当您不需要低延迟（次秒）读取时，您可以离线存储数据。您可以使用离线商店来获取用于数据探索、模型训练和批量推断的功能。
- 在线和离线 — 将数据存储在在线商店和离线商店中。

要使用目标节点创建要素组，请执行以下操作。

1. 选择+包含要导出的数据集的节点旁边的符号。
2. 在添加目标，选择SageMaker Feature Store.
3. 选择创建功能组.
4. 在以下对话框中，如果数据集没有事件时间列，请选择创建 “EventTime” 专栏.
5. 选择 Next（下一步）。

6. 选择复制 JSON 架构. 创建要素组时，可以将架构粘贴到功能定义中。
7. 选择 Create（创建）。
8. 适用于功能组名称，请为功能组指定名称。
9. 适用于Description (可选)中，指定描述以使您的要素组更容易被发现。
10. 要为在线商店创建功能组，请执行以下操作。
    a. Select在线启用存储.
    b. 适用于在线商店加密密钥中，指定Amazon托管加密密钥或自己的加密密钥。
11. 要为离线商店创建功能组，请执行以下操作。
    a. Select脱机启用存储.
    b. 指定以下字段的值：
       - S3 存储桶名称— 存储功能组的存储桶的名称。
       - （可选）数据集目录名称— 您用来存储功能组的 Amazon S3 前缀。
       - IAM 角色 ARN— 有权访问 feature store 的 IAM 角色。
       - 离线商店加密密钥— 默认情况下，要素商店使用Amazon托管密钥，但是您可以使用该字段指定自己的密钥。
12. 选择 Continue（继续）。
13. 选择 JSON。
14. 删除窗口中的占位符括号。
15. 粘贴步骤 6 中的 JSON 文本。
16. 选择 Continue（继续）。
17. 适用于记录标识符功能名称中，选择数据集中具有数据集中每条记录的唯一标识符的列。
18. 适用于事件时间功能名称中，选择包含时间戳值的列。
19. 选择 Continue（继续）。
20. （可选）添加标签以使功能组更易于发现。
21. 选择 Continue（继续）。
22. 选择创建功能组.
23. 返回到 Data Wrangler 流程，然后选择功能组搜索栏。

    Note

    如果您已经为流程中的要素组创建了目标节点，则无法为同一要素组创建另一个目标节点。如果要为同一要素组创建另一个目标节点，则必须创建另一个流文件。

使用以下过程创建 Data Wrangler 作业。

从中创建作业数据流页面，然后选择要导出的目标节点。

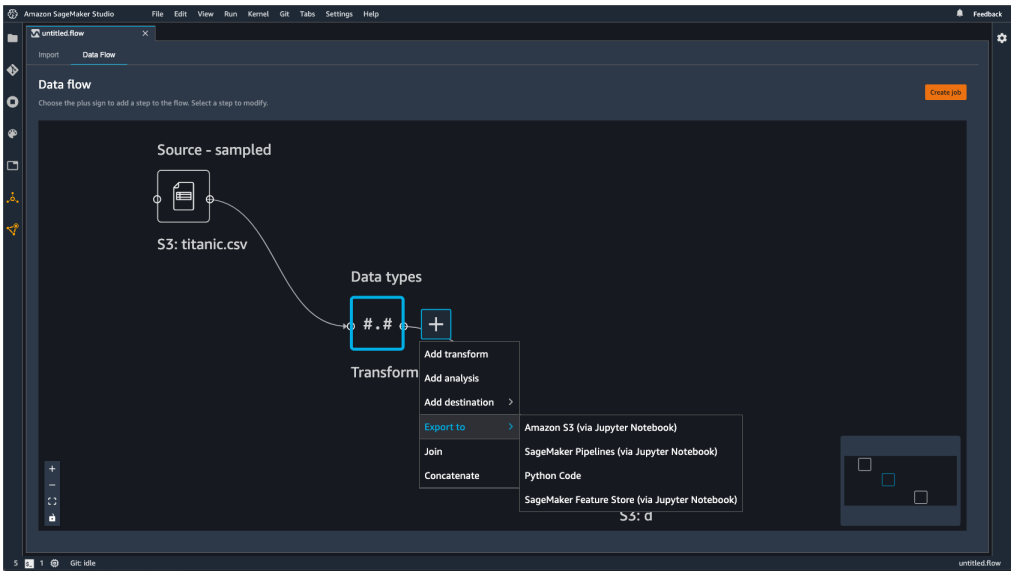
1. 请选择 Create job (创建任务)。下图显示了选择后显示的窗格。创建作业.
2. 适用于作业名称，请指定导出作业的名称。
3. 选择要导出的目标节点。
4. （可选）指定Amazon KMS密钥 ARN。一个Amazon KMS密钥是您可以用于保护数据的加密密钥。有关 的更多信息Amazon KMS钥匙，请参阅Amazon Key Management Service
5. 选择配置作业. 下图显示了配置作业页.
6. （可选）配置 Data Wrangler 作业。
7. 选择 Run（运行）。

Jupyter Notebook

使用以下过程将 Jupyter 笔记本导出到亚马逊。 SageMaker Feature Store。

使用以下过程生成一个 Jupyter 笔记本并运行它以将 Data Wrangler 流程导出到功能商店。

1. 选择+在您要导出的节点旁边。
2. 选择导出到.
3. 选择亚马逊 SageMaker 功能商店（通过 Jupyter 笔记本）.
4. 运行 Jupyter 笔记本。



运行木星笔记本会运行 Data Wrangler 作业。运行 Data Wrangler 作业会启动 SageMaker 处理作业。处理作业将流程提取到在线和离线 feature store 中。

> **Important**
>
> 用于运行此笔记本的 IAM 角色必须具有以下内容Amazon附加的托管策略：`AmazonSageMakerFullAccess`和`AmazonSageMakerFeatureStoreAccess`.

创建功能组时，您只需启用一个在线或离线 feature store。您还可以启用这两个功能。要禁用在线商店创建，请设置`EnableOnlineStore`到`False`：

```
# Online Store Configuration
online_store_config = {
    "EnableOnlineStore": False
}
```

笔记本使用您导出的数据帧的列名和类型来创建用于创建要素组的要素组架构。功能组是在要素商店中定义的用于描述记录的一组要素。功能组定义了功能组中包含的架构和功能。功能组定义由功能列表、记录标识符功能名称、事件时间功能名称以及其在线商店和离线商店的配置组成。

要素组中的每个要素都可以具有以下类型之一：字符串、分数，或者积分. 如果导出的数据框中的列不是这些类型之一，则默认为`String`.

以下是功能组架构的示例。

```
column_schema = [
```

```
    {
        "name": "Height",
        "type": "long"
    },
    {
        "name": "Input",
        "type": "string"
    },
    {
        "name": "Output",
        "type": "string"
    },
    {
        "name": "Sum",
        "type": "string"
    },
    {
        "name": "Time",
        "type": "string"
    }
]
```

此外，您必须指定记录标识符名称和事件时间功能名称：

- 这些区域有：记录标识符名称是功能的名称，其值唯一标识功能 feature store 中定义的记录的值。在线商店中只存储每个标识符值的最新记录。记录标识符要素名称必须是要素定义的名称之一。
- 这些区域有：事件时间功能名称是存储功能的名称。`EventTime`是功能组中的记录的。网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`EventTime`是发生新事件的时间点，该事件对应于创建或更新功能中的记录。功能组中的所有记录必须具有对应的`EventTime`.

笔记本使用这些配置创建功能组、大规模处理数据，然后将处理过的数据提取到在线和离线功能商店中。要了解更多信息，请参阅数据源和摄取.

笔记本使用这些配置创建功能组、大规模处理数据，然后将处理过的数据提取到在线和离线功能商店中。要了解更多信息，请参阅数据源和摄取.

安全和权限

当您查询来自 Athena 或 Amazon Redshift 的数据时，查询的数据集将自动存储在默认值中 SageMaker 用于Amazon您正在使用 Studio 的区域。此外，当您从亚马逊导出 Jupyter 笔记本时 SageMaker Data Wrangler 然后运行它，您的数据流或 .flow 文件将保存到前缀下的同一默认存储桶data_wrangler_flow.

对于高级安全需求，您可以配置存储桶策略来限制Amazon有权访问此默认设置的角色 SageMaker S3 存储桶。使用以下部分将此类策略添加到 S3 存储桶。要遵循此页面上的说明，请使用Amazon Command Line Interface(Amazon CLI)。要了解如何操作，请参阅配置AmazonCLI(在 IAM 用户指南中)。

此外，您需要授予每个使用 Data Wrangler 权限的 IAM 角色才能访问所需资源。如果您不需要用于访问 Data Wrangler 的 IAM 角色的粒度权限，则可以添加 IAM 托管策略，`AmazonSageMakerFullAccess`，转换为用于创建 Studio 用户的 IAM 角色。本政策授予您使用 Data Wrangler 的完全权限。如果您需要更精细的权限，请参阅以下部分：授予 IAM 角色使用 Data Wrangler 的权限 (p. 677).

添加存储桶策略以限制对导入到 Data Wrangler 的数据集的访问

您可以使用 Amazon S3 存储桶策略向包含 Data Wrangler 资源的 S3 存储桶添加策略。Data Wrangler 上传到默认设置的资源 SageMaker 中的 S3 存储桶Amazon您正在使用 Studio 的区域包括以下内容：

- 已查询 Amazon Redshift 结果。这些存储在redshift/prefix.
- 已查询 Athena 的结果。这些存储在athena/prefix.

- 运行导出的 Jupyter 笔记本数据 Wrangler 生成时上传到 Amazon S3 的 .flow 文件。这些存储在data_wrangler_flow/prefix.

使用以下过程创建 S3 存储桶策略，您可以添加该策略以限制 IAM 角色对该存储桶的访问。要了解如何将策略添加到 S3 存储桶，请参阅如何添加 S3 存储桶策略？.

要在存储数据 Wrangler 资源的 S3 存储桶上设置存储桶策略，请执行以下操作：

1. 配置您希望能够访问数据 Wrangler 的一个或多个 IAM 角色。
2. 打开命令提示符或 shell。对于您创建的每个角色，请替换*####*使用角色的名称并运行以下命令：

```
$ aws iam get-role --role-name role-name
```

   在回复中，你会看到RoleId以开头的字符串AROA. 复制此字符串。
3. 将以下策略添加到 SageMaker 中的默认存储桶Amazon您正在使用数据 Wrangler 的区域。Replace*##*使用Amazon以及存储桶所在的区域*account-id*和你的Amazon账户 ID。ReplaceuserIds 开始于*AROA#PLEID*用一个的 IDAmazon要授予使用数据 Wrangler 的权限的角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::sagemaker-region-account-id/data_wrangler_flows/",
        "arn:aws:s3:::sagemaker-region-account-id/data_wrangler_flows/*",
        "arn:aws:s3:::sagemaker-region-account-id/athena",
        "arn:aws:s3:::sagemaker-region-account-id/athena/*",
        "arn:aws:s3:::sagemaker-region-account-id/redshift",
        "arn:aws:s3:::sagemaker-region-account-id/redshift/*"

      ],
      "Condition": {
        "StringNotLike": {
          "aws:userId": [
            "AROAEXAMPLEID_1:*",
            "AROAEXAMPLEID_2:*"
          ]
        }
      }
    }
  ]
}
```

授予 IAM 角色使用 Data Wrangler 的权限

您可以授予 IAM 角色权限，以便将 Data Wrangler 与一般 IAM 托管策略一起使用，AmazonSageMakerFullAccess. 这是一项通用政策，其中包括许可需要使用所有 SageMaker 服务。此策略授予对数据 Wrangler 的 IAM 角色的完全访问权限。在使用时，您应了解以下事项AmazonSageMakerFullAccess要授予对数据 Wrangler 的访问权限：

- 如果您从 Amazon Redshift 导入数据，数据库用户必须具有前缀sagemaker_access.
- 此托管策略仅授予访问名称中具有以下内容之一的存储桶的权限：SageMaker、SageMaker、sagemaker，或者aws-glue. 如果想使用 Data Wrangler 从 S3 存储桶中导入名称中没有这些短语，请参阅本页的最后一节了解如何向 IAM 实体授予访问 S3 存储桶的权限。

如果您有高安全性需求，可以将本部分中的策略附加到 IAM 实体，以授予使用 Data Wrangler 所需的权限。

如果您在 Amazon Redshift 或 Athena 中有数据集需要从 Data Wrangler 导入 IAM 角色，则必须向该实体添加策略才能访问这些资源。以下策略是您可以用来授予 IAM 角色从 Amazon Redshift 和 Athena 导入数据的权限最严格的策略。

要了解如何将自定义策略附加到 IAM 角色，请参阅管理 IAM 策略(在 IAM 用户指南中)。

授予 Athena 数据集导入访问权限的策略示例

以下策略假定 IAM 角色有权访问通过单独的 IAM 策略存储数据的底层 S3 存储桶。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:ListDataCatalogs",
                "athena:ListDatabases",
                "athena:ListTableMetadata",
                "athena:GetQueryExecution",
                "athena:GetQueryResults",
                "athena:StartQueryExecution",
                "athena:StopQueryExecution"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateTable"
            ],
            "Resource": [
                "arn:aws:glue:*:*:table/*/sagemaker_tmp_*",
                "arn:aws:glue:*:*:table/sagemaker_featurestore/*",
                "arn:aws:glue:*:*:catalog",
                "arn:aws:glue:*:*:database/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:DeleteTable"
            ],
            "Resource": [
                "arn:aws:glue:*:*:table/*/sagemaker_tmp_*",
                "arn:aws:glue:*:*:catalog",
                "arn:aws:glue:*:*:database/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabases",
                "glue:GetTable",
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:*:*:table/*",
                "arn:aws:glue:*:*:catalog",
                "arn:aws:glue:*:*:database/*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase"
            ],
            "Resource": [
                "arn:aws:glue:*:*:catalog",
                "arn:aws:glue:*:*:database/sagemaker_featurestore",
                "arn:aws:glue:*:*:database/sagemaker_processing",
                "arn:aws:glue:*:*:database/default",
                "arn:aws:glue:*:*:database/sagemaker_data_wrangler"
            ]
        }
    ]
}
```

授予对 Amazon Redshift 数据集导入权限的策略示例

以下策略授予使用具有前缀的数据库用户设置与 Data Wrangler 的 Amazon Redshift 连接的权限。`sagemaker_access`在名字里。要授予使用其他数据库用户进行连接的权限，请在`"Resources"`在以下策略中。以下策略假定 IAM 角色有权访问底层 S3 存储桶（如果适用），其中数据通过单独的 IAM 策略存储。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "redshift-data:ExecuteStatement",
                "redshift-data:DescribeStatement",
                "redshift-data:CancelStatement",
                "redshift-data:GetStatementResult",
                "redshift-data:ListSchemas",
                "redshift-data:ListTables"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "redshift:GetClusterCredentials"
            ],
            "Resource": [
                "arn:aws:redshift:*:*:dbuser:*/sagemaker_access*",
                "arn:aws:redshift:*:*:dbname:*"
            ]
        }
    ]
}
```

授予对 S3 存储桶访问权限的策略

如果您的数据集存储在 Amazon S3 中，则可以使用类似于以下策略授予 IAM 角色访问此存储桶的权限。此示例授予对名为的存储桶的编程读写访问权限##.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::test"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject"
      ],
      "Resource": ["arn:aws:s3:::test/*"]
    }
  ]
}
```

要从 Athena 和 Amazon Redshift 导入数据，您必须授予 IAM 角色权限，才能访问Amazon正在使用的区域数据 Wrangler：`athena/`、`redshift/`. 如果在中尚不存在默认 Amazon S3 存储桶Amazon区域，您还必须向 IAM 角色授予在此区域创建存储桶的权限。

此外，如果您希望 IAM 角色能够使用亚马逊 SageMaker 特色商店， SageMaker 管道和 Data Wrangler 作业导出选项，您必须授予对前缀的访问权限`data_wrangler_flows/`在此存储桶中。

数据 Wrangler 使用`athena/`和`redshift/`用于存储预览文件和导入的数据集的前缀。要了解更多信息，请参阅 导入的数据存储 (p. 600)。

数据 Wrangler 使用`data_wrangler_flows/`在运行从 Data Wrangler 导出的 Jupyter 笔记本时存储 .flow 文件的前缀。要了解更多信息，请参阅 Export (p. 664)。

使用类似于以下内容的策略授予上述段落中描述的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::sagemaker-region-account-id/data_wrangler_flows/",
                "arn:aws:s3:::sagemaker-region-account-id/data_wrangler_flows/*",
                "arn:aws:s3:::sagemaker-region-account-id/athena",
                "arn:aws:s3:::sagemaker-region-account-id/athena/*",
                "arn:aws:s3:::sagemaker-region-account-id/redshift",
                "arn:aws:s3:::sagemaker-region-account-id/redshift/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::sagemaker-region-account-id"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:ListAllMyBuckets",
                "s3:GetBucketLocation"
            ],
            "Resource": "*"
        }
    ]
}
```

您还可以从另一个存储桶访问 Amazon S3 存储桶中的数据Amazon通过指定 Amazon S3 存储桶 URI 来执行账户。为此，授予对另一账户中 Amazon S3 存储桶的访问权限的 IAM 策略应使用类似于以下示例的策略，其中`BucketFolder`是用户存储桶中的特定目录`UserBucket`. 应将此策略添加到向另一个用户授予其存储桶访问权限的用户中。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::UserBucket/BucketFolder/*"
            },
                {
                "Effect": "Allow",
                "Action": [
                    "s3:ListBucket"
                ],
                "Resource": "arn:aws:s3:::UserBucket",
                "Condition": {
                "StringLike": {
                    "s3:prefix": [
                    "BucketFolder/*"
                    ]
                }
            }
        }
    ]
}
```

访问存储桶的用户（而不是存储桶拥有者）必须向其用户添加类似以下示例的策略。请注意，`AccountX`和`TestUser`下面分别指的是存储桶拥有者及其用户。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountX:user/TestUser"
            },
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::UserBucket/BucketFolder/*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountX:user/TestUser"
            },
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::UserBucket"
            ]
        }
    ]
}
```

授予使用 SageMaker Studio 的访问权限的策略示例

使用类似以下内容的策略创建 IAM 执行角色，该角色可用于设置 Studio 实例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedDomainUrl",
                "sagemaker:DescribeDomain",
                "sagemaker:ListDomains",
                "sagemaker:DescribeUserProfile",
                "sagemaker:ListUserProfiles",
                "sagemaker:*App",
                "sagemaker:ListApps"
            ],
            "Resource": "*"
        }
    ]
}
```

雪花和数据牧马人

所有的权限Amazon资源通过附加到 Studio 实例的 IAM 角色进行管理。Snowflake 管理员管理 Snowflake 特定的权限，因为他们可以向每个 Snowflake 用户授予精细的权限和权限。这包括数据库、架构、仓库和存储集成对象。您必须确保在 Data Wrangler 之外设置了正确的权限。

请注意，雪花`COPY INTO Amazon S3`命令默认情况下，通过公共互联网将数据从 Snowflake 移动到 Amazon S3，但使用 SSL 保护传输中的数据。Amazon S3 中的静态数据使用默认值使用 SSE-KMS 进行加密Amazon KMS key.

关于 Snowflake 凭据存储，Data Wrangler 不存储客户凭据。作为最佳实践安全计划的一部分，Data Wrangler 使用 Secret Manager 将证书存储在密钥中，并轮换密钥。Snowflake 或 Studio 管理员需要确保向数据科学家的 Studio 执行角色授予执行权限`GetSecretValue`在存储凭据的秘密上。如果已附加到 Studio 执行角色，则`AmazonSageMakerFullAccess`策略具有读取 Data Wrangler 创建的密钥以及按照上述说明中的命名和标记约定创建的密钥的必要权限。必须单独授予不遵守约定的机密访问权限。我们建议使用 Secrets Manager 来防止通过不安全的渠道共享凭据；但请注意，登录用户可以通过在 Studio 中启动终端或 Python 笔记本电脑然后从 Secret Manager API 调用 API 调用来检索纯文本密码。

使用数据加密Amazon KMS

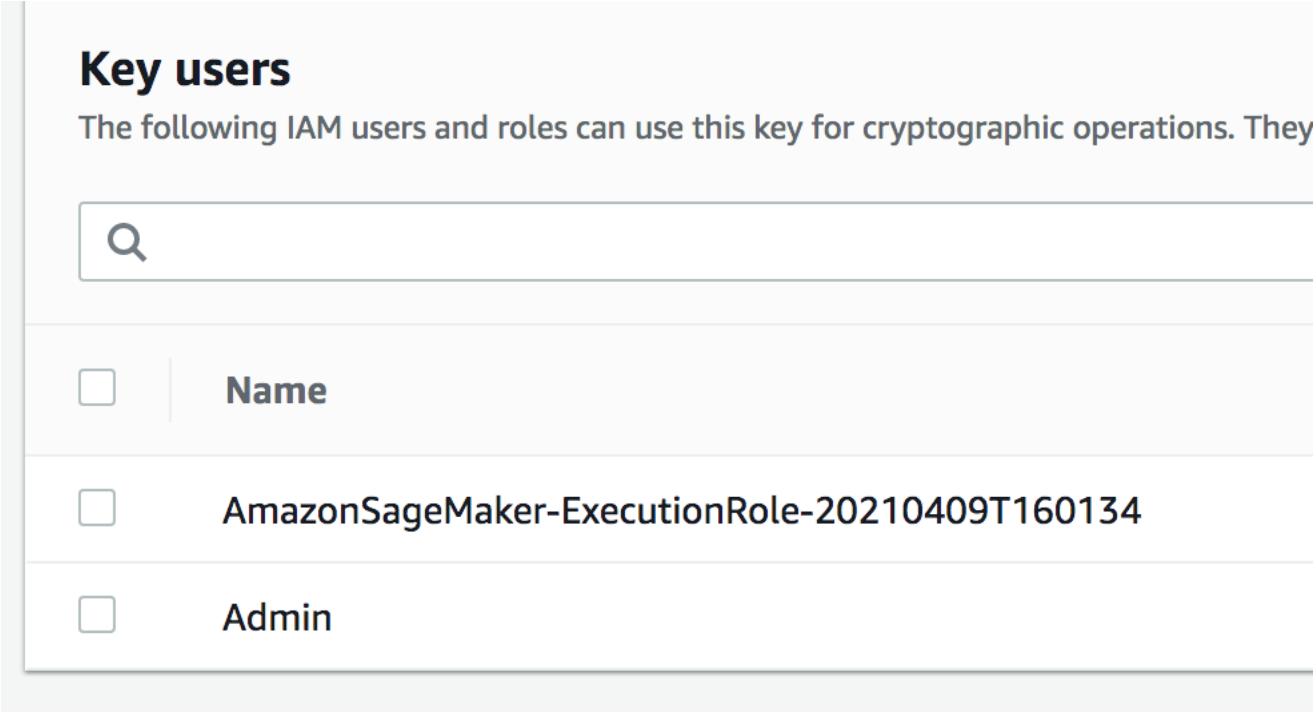
在 Data Wrangler 中，您可以解密加密的文件并将其添加到 Data Wrangler 流程中。您也可以使用默认值加密变换的输出Amazon KMS钥匙或者你提供的。

如果文件具有以下内容，则可以导入文件：

- 服务器端加密
- SSE-KMS 作为加密类型

要解密文件并导入数据 Wrangler 流程，您必须添加 SageMaker 作为关键用户使用的 Studio 用户。

以下屏幕截图显示作为关键用户添加的 Studio 用户角色。请参阅IAM 角色以访问左侧面板下的用户以进行此更改。



Amazon S3 客户管理的 Data Wrangler 导入数据存储的密钥设置

默认情况下，Data Wrangler 使用具有以下命名约定的 Amazon S3 存储桶：`sagemaker-region-account number`. 例如，如果您的账号是`111122223333`而且您在 us-east-1 中使用 Studio 时，导入的数据集将按照以下命名约定存储：`sagemaker-us-east-1-111122223333`.

以下说明说明如何为默认 Amazon S3 存储桶设置客户托管密钥。

1. 要启用服务器端加密并为默认 S3 存储桶设置客户托管密钥，请参阅使用 KMS 加密.
2. 在执行步骤 1 之后，导航到Amazon KMS在您的Amazon Web Services Management Console. 找到您在上一步的步骤 1 中选择的客户管理密钥，然后将 Studio 角色添加为关键用户。为此，请遵循中的说明允许密钥用户使用客户托管密钥.

加密您导出的数据

您可以使用以下方法之一加密导出的数据：

- 指定您的 Amazon S3 存储桶具有对象使用 SSE-KMS 加密。
- 指定Amazon KMS密钥用于加密您从 Data Wrangler 导出的数据。

在存储库的导出数据页面中，指定一个值Amazon KMS或 ARN 密钥.

有关使用的更多信息Amazon KMS钥匙，请参阅通过服务器端加密保护数据Amazon KMS存储在中的密钥 AmazonAmazon Key Management Service（SSE-KMS）.

发行说明

数据 Wrangler 会定期更新新功能和错误修复。要升级你在 Studio 中使用的 Data Wrangler 版本，请按照中的说明操作关闭并更新 Studio 应用 (p. 130).

发行说明

2/2022

新功能：

- 您现在可以使用目标节点进行导出。有关更多信息，请参阅 。Export (p. 664)
- 您可以导入 ORC 和 JSON 文件。有关文件类型的更多信息，请参阅导入 (p. 566).
- 数据 Wrangler 现在支持使用 SMOTE 转换。有关更多信息，请参阅 余额数据 (p. 628)。
- Data Wrangler 现在支持分类数据的相似性编码。有关更多信息，请参阅 相似性编码 (p. 635)。
- Data Wrangler 现在支持取消嵌套 JSON 数据。有关更多信息，请参阅 取消嵌套 JSON 数据 (p. 656)。
- Data Wrangler 现在支持将数组的值扩展为单独的列。有关更多信息，请参阅 分解数组 (p. 657)。
- Data Wrangler 现在支持在遇到问题时与服务团队联系。有关更多信息，请参阅 问题排查 (p. 685)。
- Data Wrangler 支持编辑和删除数据流中的步骤。有关更多信息，请参阅 从数据流中删除步骤 (p. 605) 和 编辑数据 Wrangler 流程中的步骤 (p. 609)。
- 您现在可以对多列执行转换。有关更多信息，请参阅 转换数据 (p. 623)。
- 数据 Wrangler 现在支持成本分配标签。有关更多信息，请参阅使用成本分配标签。

10/16/2021

新功能：

Data Wrangler 现在支持 Athena 工作组。有关更多信息，请参阅 从 Athena 导入数据 (p. 570)。

10/6/2021

新功能：

Data Wrangler 现在支持转换时间序列数据。有关更多信息，请参阅 转换时间序列 (p. 638)。

7/15/2021

新功能：

- 雪花和数据牧马人 (p. 682)现在支持。您可以在 Data Wrangler 中使用 Snowflake 作为数据源。
- 添加了对 CSV 中自定义字段分隔符的支持。现在支持逗号、冒号、分号、管道 (|) 和 Tab。
- 现在，您可以直接将结果导出到 Amazon S3。
- 添加了一些新的多共线性分析仪：差异通货膨胀因素、主成分分析和套索功能选择。

增强功能：

- 分析图表不能再包含重叠的标签。

错误修复：

发行说明

- One Hot 编码器优雅地处理空字符串。
- 修复了当数据框列名包含点时发生的崩溃问题。

4/26/2021

增强功能：

- 添加对分布式处理作业的支持。运行处理作业时，您可以使用多个实例。
- 当估计结果大小小于 1 GB 时，Data Wrangler 处理作业现在会自动合并小输出。
- 功能商店笔记本：改进了 feature store 摄取性能
- Data Wrangler 处理作业现在使用 1.x 作为 future 版本的权威容器标签。

错误修复：

- 修复了多面直方图的渲染问题。
- Fixed导出到处理 Job以支持矢量类型列。
- Fixed`Extract using regex`运算符返回第一个捕获的组，如果正则表达式或正则表达式中存在一个或多个组。

2/8/2021

新功能：

- Data Wrangler Flow 支持多个实例。
- 更新了导出到数据 Wrangler Job 笔记本以使用 SageMaker SDK 2.20.0。
- 更新了导出到管道笔记本以使用 SageMaker SDK 2.20.0。
- 更新了导出到流水线笔记本以添加 xgBoost 训练示例作为可选步骤。

增强功能：

- 为了提高性能，不再支持导入在单个字段中包含多行的 CSV 文件。

错误修复：

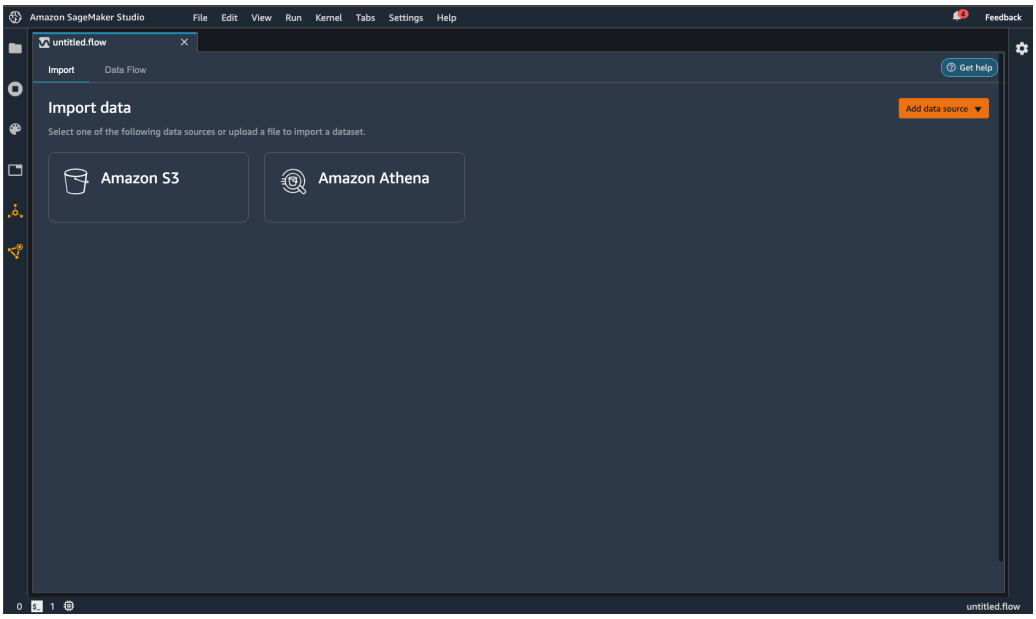
- 修复了 Quick 模型中的类型推理问题。
- 修复了偏差报告中的偏差指标错误。
- 修复了 “功能化” 文本转换以处理缺少值的列的问题。
- 修复了直方图和散点图内置的可视化效果，以处理包含类似数组列的数据集。
- 如果查询执行 ID 已过期，Athena 查询现在会重新运行。

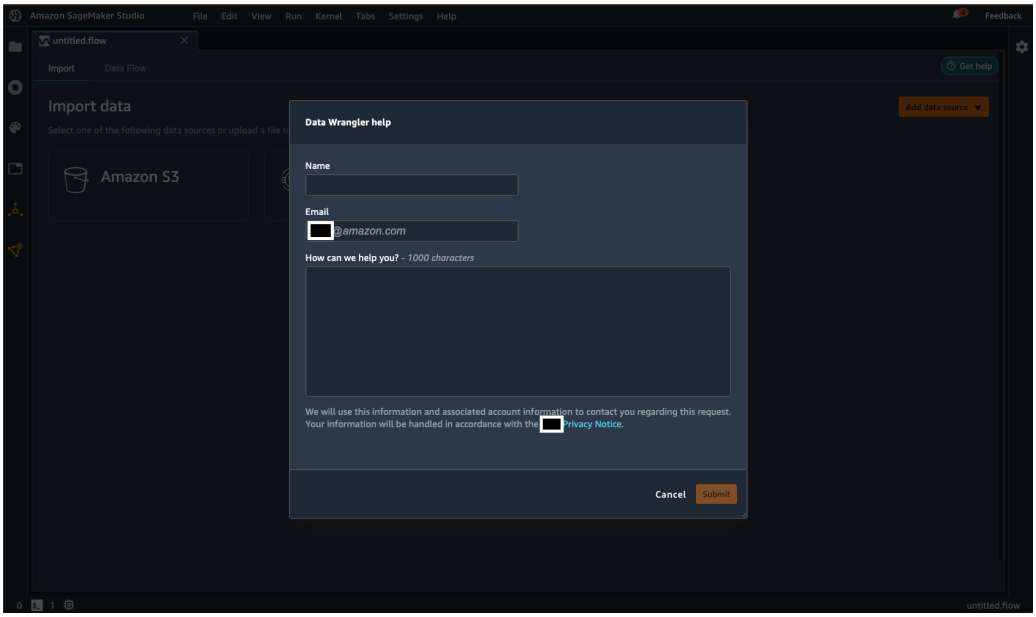
问题排查

如果使用亚马逊时出现问题 SageMaker 数据 Wrangler，建议执行以下操作：

- 如果提供了错误消息，请阅读该消息并解决其报告的问题（如果可能）。
- 确保 Studio 用户的 IAM 角色具有执行操作所需的权限。有关更多信息，请参阅 安全和权限 (p. 676)。
- 如果在尝试从另一个导入时出现问题Amazon服务，例如 Amazon Redshift 或 Athena，请确保您已配置必要的权限和资源来执行数据导入。有关更多信息，请参阅 导入 (p. 566)。

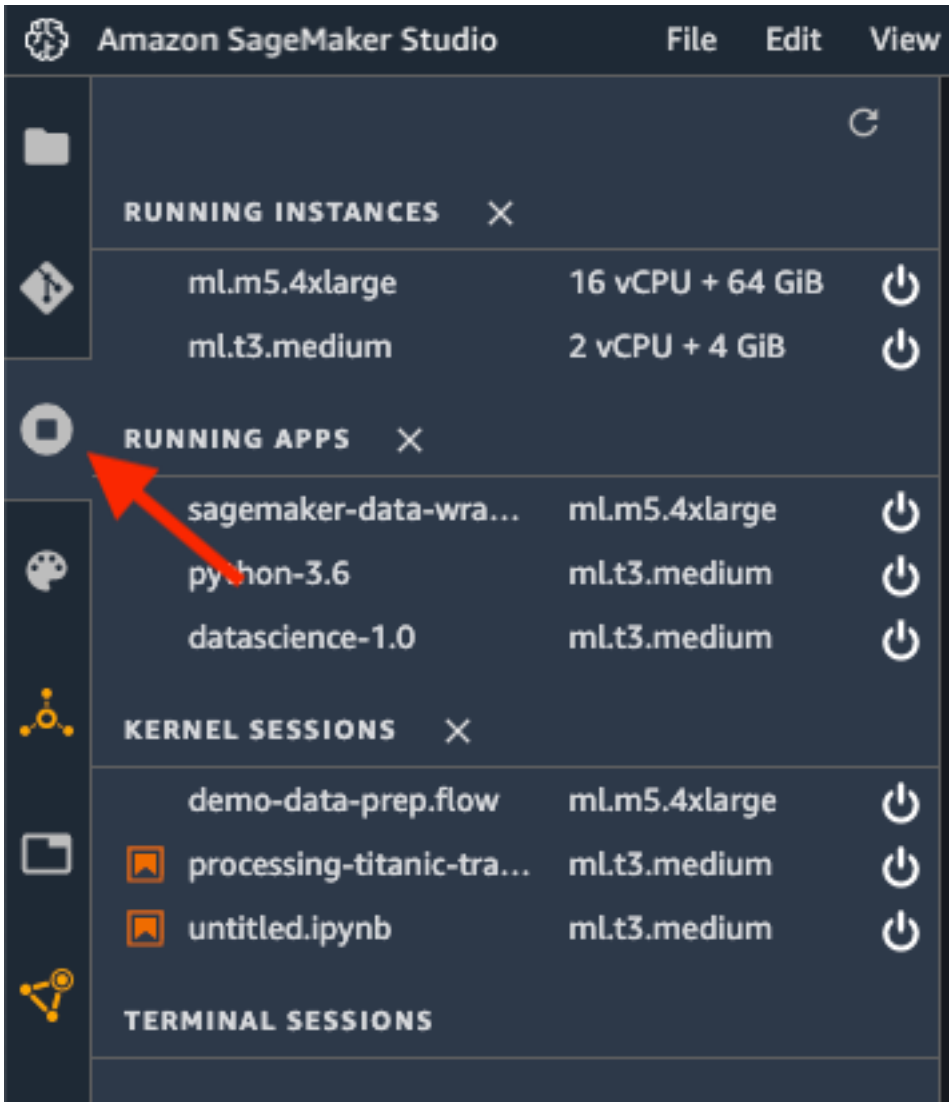
- 如果您仍然遇到问题，可以选择获取帮助在屏幕右上角联系 Data Wrangler 团队。有关更多信息，请参阅下图：.



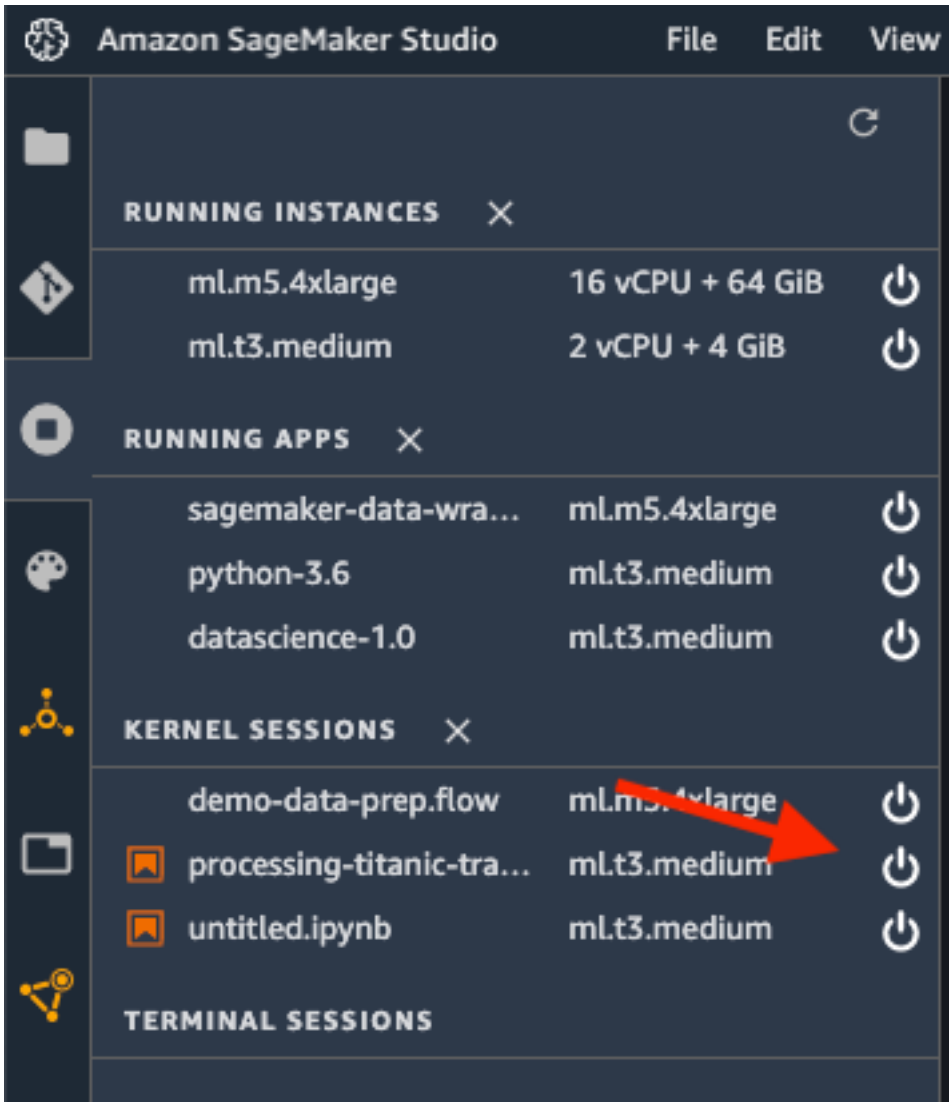


作为最后的手段，你可以尝试重新启动运行 Data Wrangler 的内核。

1. 保存并退出要重新启动内核的 .flow 文件。
2. Select.运行终端和内核图标，如下图所示。



3. Select.停止图标位于要终止内核的 .flow 文件右侧，如下图所示。



4. 刷新浏览器。
5. 重新打开你正在处理的 .flow 文件。

更新数据 Wrangler

要将 Data Wrangler 更新到最新版本，请首先从亚马逊关闭相应的 KernelGateway 应用程序 SageMaker Studio 控制面板。After KernelGateway 应用程序已关闭，通过在 Studio 中打开新的或现有的 Data Wrangler 流程来重新启动它。当你打开新的或现有的 Data Wrangler 流程时，启动的内核包含最新版本的 Data Wrangler。

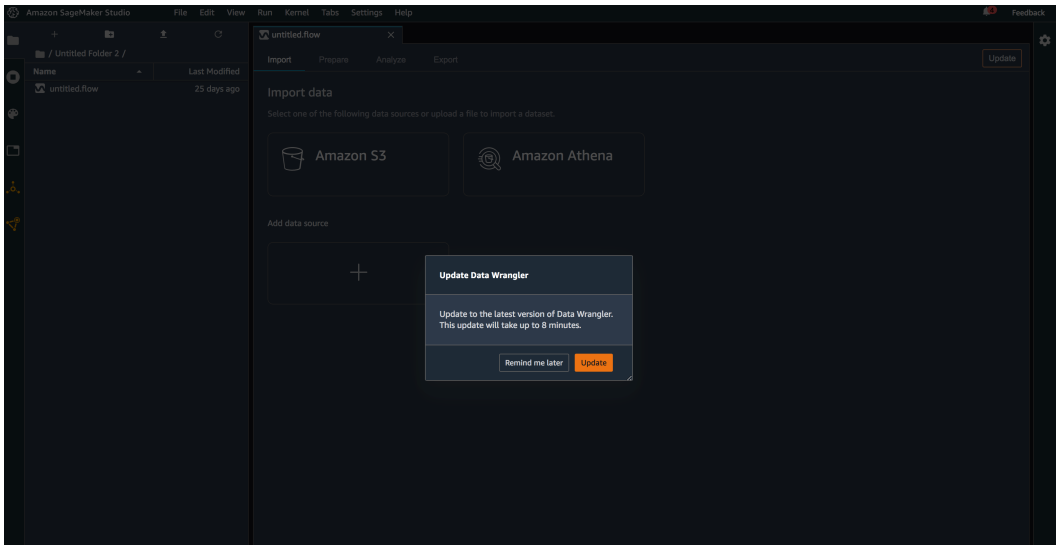
更新你的 Studio 和数据 Wrangler 实例

1. 导航到您的SageMaker 控制台.
2. 选择 SageMaker 然后是 Studio。
3. 选择您的用户名。
4. UDER应用程序，在显示应用程序名，选择删除 App对于以开头的应用程序`sagemaker-data-wrang`，以及 JupyterServer 应用程序。

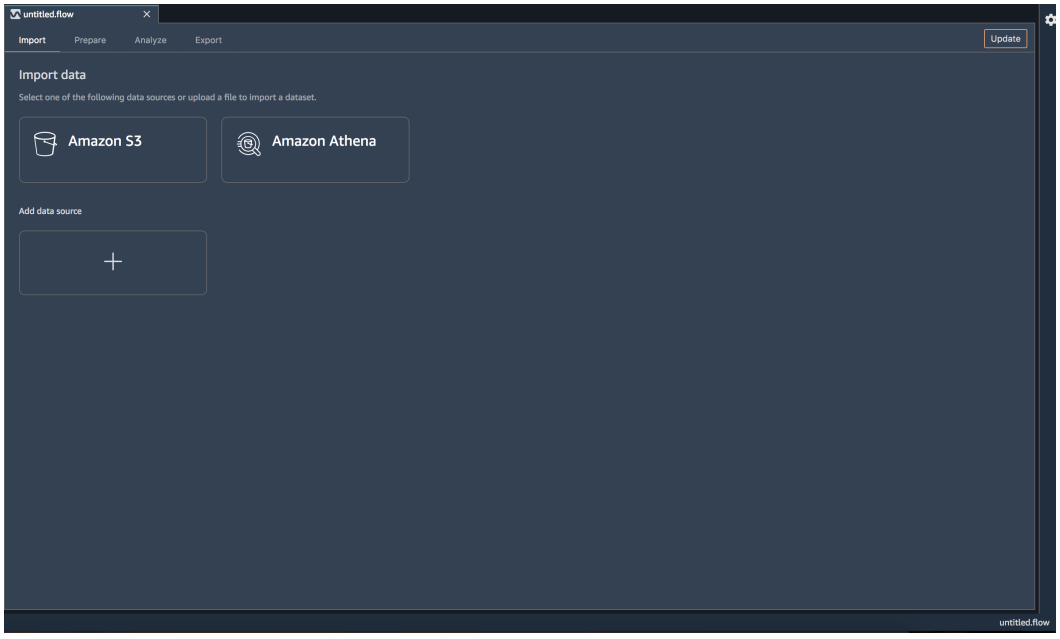
5. 选择是的，删除应用程序.
6. 类型`delete`在确认框中。
7. 请选择 Delete（删除）。
8. 重新打开你的 Studio 实例。当您开始创建 Data Wrangler 流程时，您的实例现在使用最新版本的 Data Wrangler。

或者，如果您使用的是不是最新版本的 Data Wrangler 应用程序版本，并且现有的 Data Wrangler 流程已打开，则系统会提示您在 Studio UI 中更新 Data Wrangler 应用程序版本。以下屏幕截图显示此提示。

> **Important**
>
> 这仅更新 Data Wrangler 内核网关应用程序。您还需要关闭 JupyterServer 应用程序在您的用户帐户中。为此，请按照前面的步骤进行操作。



还可以选择稍后提醒我，在这种情况下更新按钮显示在屏幕的右上角。



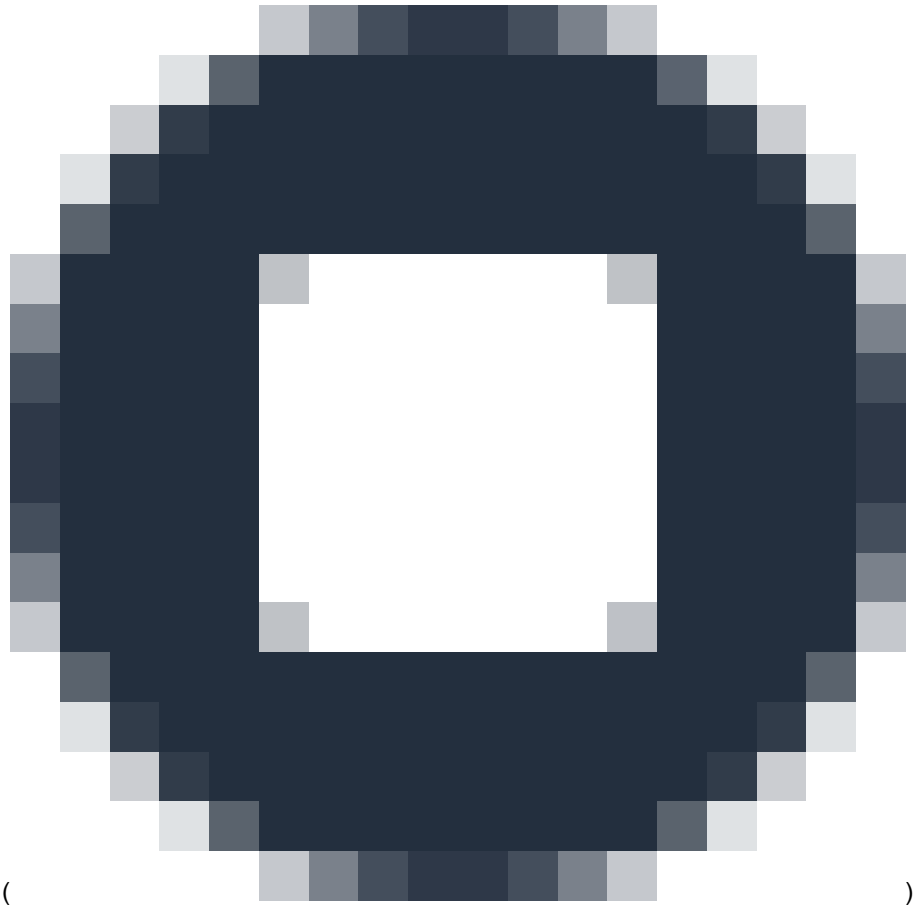
关闭数据 Wrangler

当您不使用 Data Wrangler 时，为了避免产生额外费用，必须关闭其运行的实例。

为避免丢失工作，请在关闭 Data Wrangler 之前保存数据流。要在 Studio 中保存数据流，请选择File (文件)然后选择保存数据 Wrangler Flow. Data Wrangler 每 60 秒自动保存一次数据流。

在 Studio 中关闭 Data Wrangler 实例

1. 在 Studio 中，选择运行实例和内核图标

   ( )。
2. 下运行应用是sagemaker-1.0应用。选择此应用旁边的关机图标

   ( )。

   数据 Wrangler 在 ml.m5.4xlarge 实例上运行。此实例从中消失运行实例当你关闭 Data Wrangler 应用程序时。

关闭 Data Wrangler 应用程序后，下次打开 Data Wrangler 流程文件时，它必须重新启动。这可能需要几分钟的时间。

使用 Studio 笔记本大规模准备数据

Studio 为数据科学家、机器学习 (ML) 工程师和全科医生提供了大规模执行数据分析和数据准备的工具。在 Studio 笔记本中，您可以直观地浏览、发现和连接到Amazon EMR. 连接后，您可以使用 Apache Spark、Hive 和 Presto 以交互方式探索、可视化和准备 PB 级数据以供机器学习 (ML) 使用。

分析、转换和准备大量数据是任何数据科学和机器学习工作流程的基本步骤。在 Amazon EMR 和 SageMaker Studio 笔记本上运行交互式分析和数据准备可以作为完整的数据科学和数据工程工作流程的统一环境。

Studio 还支持与同事共享笔记本电脑的工具，以便通过 UI 进行协作。借助此功能，您现在可以直接从 Studio 笔记本构建机器学习工作流程。使用 SageMaker Studio 连接到 Amazon EMR 集群还可以通过简化机器学习工作流程的设置来帮助提高团队效率。

用于连接到 Amazon EMR 集群的支持映像和内核如下所示：

- 映像：数据科学、Spark Magic、PyTorch Clow 1.8、TensorFlow 2.8
- 内核：PySpark 和 Spark 内核用于正在运行的应用程序下的 SparkMagic 映像，Python 3（iPython）用于数据科学映像。

有关如何从 Studio 连接到 Amazon EMR 集群的指导说明，请参阅。使用 SageMaker Studio 笔记本电脑执行交互式数据工程和数据科学.

有关所需权限的详细信息，请参阅。所需权限 (p. 725).

先决条件

- 您需要访问设置为使用 Amazon Virtual Private Cloud (Amazon VPC) 模式的 SageMaker Studio。
  - 要连接到亚马逊 EMR，必须将 Studio 配置为仅限亚马逊 VPC 模式。有关更多信息，请参阅 。将 SageMaker Studio 笔记本 Connect 到 Amazon VPC 中的资源. 有关如何登录的更多信息，请参阅注册到 SageMaker 域名.
- SageMaker Studio 使用的所有子网必须是私有子网。
- 如果您将`sm-analytics`实用程序来配置 SparkMagic 内核，请遵循以下两个先决条件之一：
  - 确保 Amazon VPC 接口终端节点已连接到 SageMaker Studio 使用的所有子网。
  - 确保 SageMaker Studio 使用的所有子网都路由为使用 NAT 网关。有关更多信息，请参阅 。NAT 网关.
- 如果以下任一点适用于您，则在使用亚马逊 EMR 时必须安装 Spark 和 Livy。
  - 您的亚马逊 EMR 集群与 Studio 位于同一亚马逊 VPC 中。
  - 您的集群位于连接到 Studio 中的 Amazon VPC 中的亚马逊 VPC 中。
- Amazon SageMaker Studio 和亚马逊 EMR 的安全组必须允许相互访问和相互访问。
- 您的亚马逊 EMR 安全组必须打开端口 8998，以便 Amazon SageMaker Studio 可以通过 Livy 与 Spark 集群进行通信。有关设置安全组的更多信息，请参阅。在亚马逊 EMR 中构建 Spark 支持的 SageMaker 笔记本.
- 要从 Studio 连接到亚马逊 EMR 群集，您必须首先访问 SageMaker Studio。如果您尚未设置 SageMaker Studio，请按照入门指南.
- 如果您在 Studio 设置期间创建了新域，那么从 Studio 发现 Amazon EMR 群集应该可供您使用。
  - 如果要重复使用现有域，则必须同时更新 Studio 和 Studio 应用程序。有关详细说明，请参阅更新 Studio和更新 Studio 应用程序.

自带图片

如果你想自带映像，请先将以下依赖项安装到你的内核中。以下列表显示：`pip`带有要安装的库名称的命令。

```
pip install sparkmagic
pip install sagemaker-studio-sparkmagic-lib
pip install sagemaker-studio-analytics-extension
```

如果不是最新版本，则可以手动更新之前列表中的库。

如果要使用 Kerberos 身份验证连接到 Amazon EMR，则必须安装 kinit 客户端。根据您的操作系统，安装 kinit 客户端的命令可能会有所不同。要携带 Ubuntu（基于 Debian）的映像，请使用`apt-get install -y -qq krb5-user`命令。

主题

- 从 Studio 发现亚马逊 EMR 集群 (p. 692)
- 从 Studio Connect 到 Amazon EMR 集群 (p. 695)
- 在 Amazon EMR 中排除和监控工作负载 (p. 709)
- 从 Studio 管理亚马逊 EMR 集群 (p. 710)
- 所需权限 (p. 725)

从 Studio 发现亚马逊 EMR 集群

在 Studio 内，数据科学家和数据工程师可以轻松发现、连接和管理 Amazon EMR 集群。您的亚马逊 EMR 集群可能位于相同Amazon帐户为 SageMaker Studio 或者他们可能位于不同的Amazonaccount. 本节介绍了如何发现与 SageMaker Studio 位于同一账户中的亚马逊 EMR 集群。有关在其他地方发现 Amazon EMR 集群的信息Amazon账户，请参阅跨账户发现亚马逊 EMR 集群 (p. 695).

在激活发现功能之前，必须向 Studio 执行角色添加必需的策略。添加此策略后，您可以连接到 Studio 中的 Amazon EMR 群集。

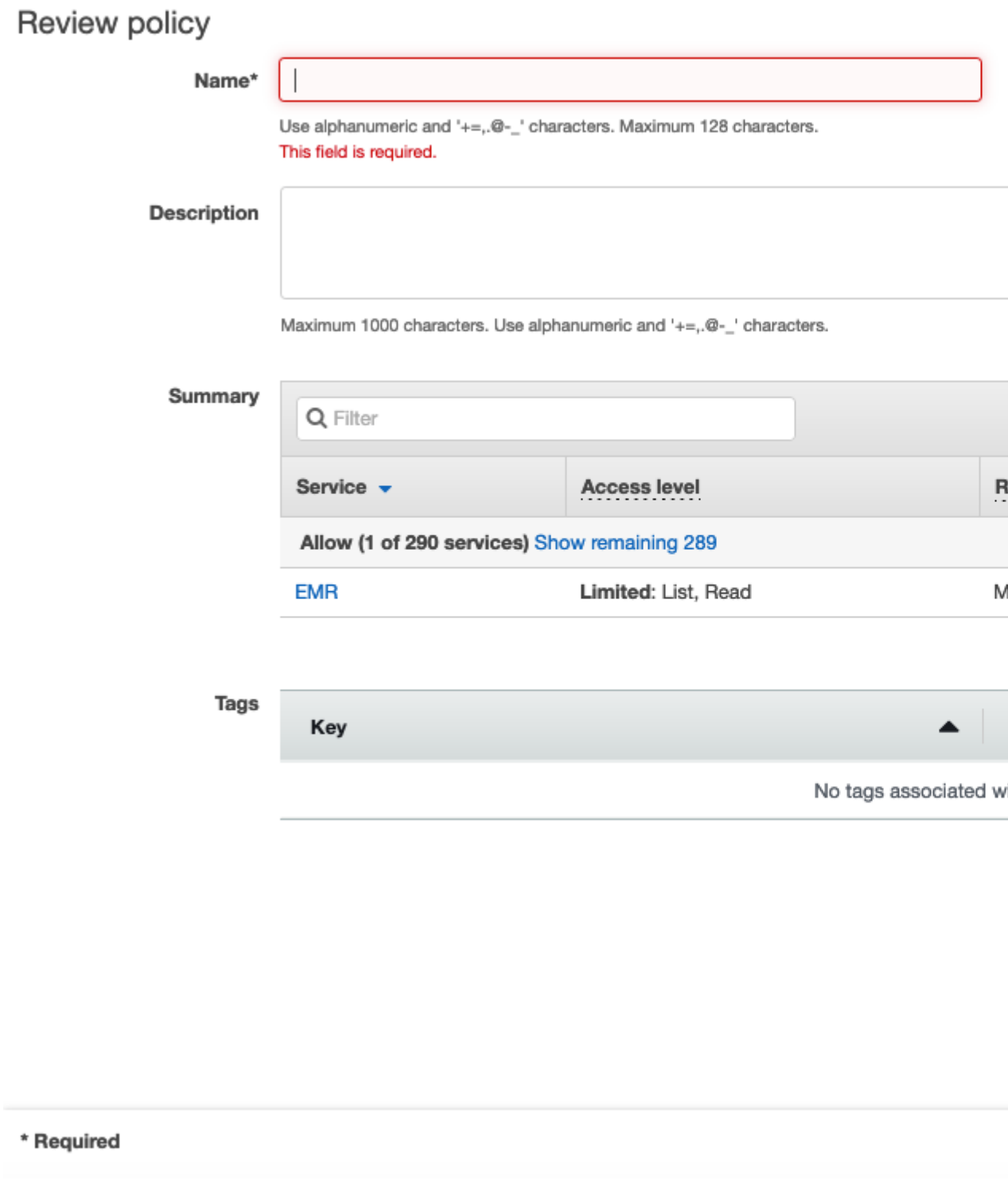
将所需的策略添加到 Studio 执行角色

1. 打开 IAM 控制台。
2. Select角色在左侧面板中。
3. 找到你要使用的 Studio 执行角色并选择它。
4. 在Permissions (权限)选项卡，选择附加策略.
5. Select创建策略.
6. SelectJSON.
7. 复制并粘贴以下策略。有关所需权限的更多信息，请参阅。所需权限 (p. 725).

```
{
    "Statement":
    [
        {
            "Action":
            [
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Effect": "Allow",
            "Resource":
            [
                "arn:aws:elasticmapreduce:*:*:cluster/*"
            ]
        },
        {
```

```
            "Action":
            [
                "elasticmapreduce:ListClusters"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
```

8. Select后续：标签。
9. Select后续：审核。
10. 输入名称、说明，然后选择创建策略.



11. 现在你可以登录 Studio 了。如果这是您首次登录，请按照中的说明操作。登录到 Studio.

跨账户发现亚马逊 EMR 集群

如果你想在不同的地方发现 Amazon EMR 集群Amazon账户比 SageMaker Studio 所在的账户，您必须在远程账户中指定 IAM 角色 ARN。必须承担此 IAM 角色 ARN 才能列出和描述 Amazon EMR 集群。在名为的文件中指定此远程角色`emr-discovery-iam-role-arns-DO_NOT_DELETE.json`，并在名为的目录中`.cross-account-configuration-DO_NOT_DELETE`. 您可以在主目录中找到此信息，该目录位于 SageMaker Studio 使用的 Amazon EFS 存储卷中。您可以使用生命周期配置 (LCC) 脚本自动执行此过程。您可以将 LCC 附加到 Studio 域或用户配置文件。您可以选择将 LCC 脚本设置为在 Jupyter 服务器启动时默认运行。

您使用的 LCC 脚本必须是 JupyterServer 配置。有关如何创建和使用 LCC 脚本以及如何将其附加到的更多信息。`Domain`和`UserProfile`级别，请参阅在 Studio 中使用生命周期配置. 有关所需权限的更多信息，请参阅。所需权限 (p. 725).

1. 以下是您可以使用的示例 LCC 脚本。要修改脚本，请使用以下详细信息填充脚本。确保您更换`ASSUMABLE-ROLE`和 123456789012 分别提供你的角色名称和账户 ID。角色名称和账户 ID 组合限制为一个。

```
# This script creates the file that informs SageMaker Studio that the role
 "arn:aws:iam::123456789012:role/ASSUMABLE-ROLE" in remote account "123456789012" must be
 assumed to list and describe EMR clusters in the remote account.

#!/bin/bash

set -eux

FILE_DIRECTORY="/home/sagemaker-user/.cross-account-configuration-DO_NOT_DELETE"
FILE_NAME="emr-discovery-iam-role-arns-DO_NOT_DELETE.json"
FILE="$FILE_DIRECTORY/$FILE_NAME"

mkdir -p $FILE_DIRECTORY

cat > "$FILE" <<- "EOF"
{
  "123456789012": "arn:aws:iam::123456789012:role/ASSUMABLE-ROLE"
}
EOF
```

从 Studio Connect 到 Amazon EMR 集群

本指南介绍了如何在选择了 PySpark 内核的情况下从 SageMaker Studio 连接到亚马逊 EMR 集群。

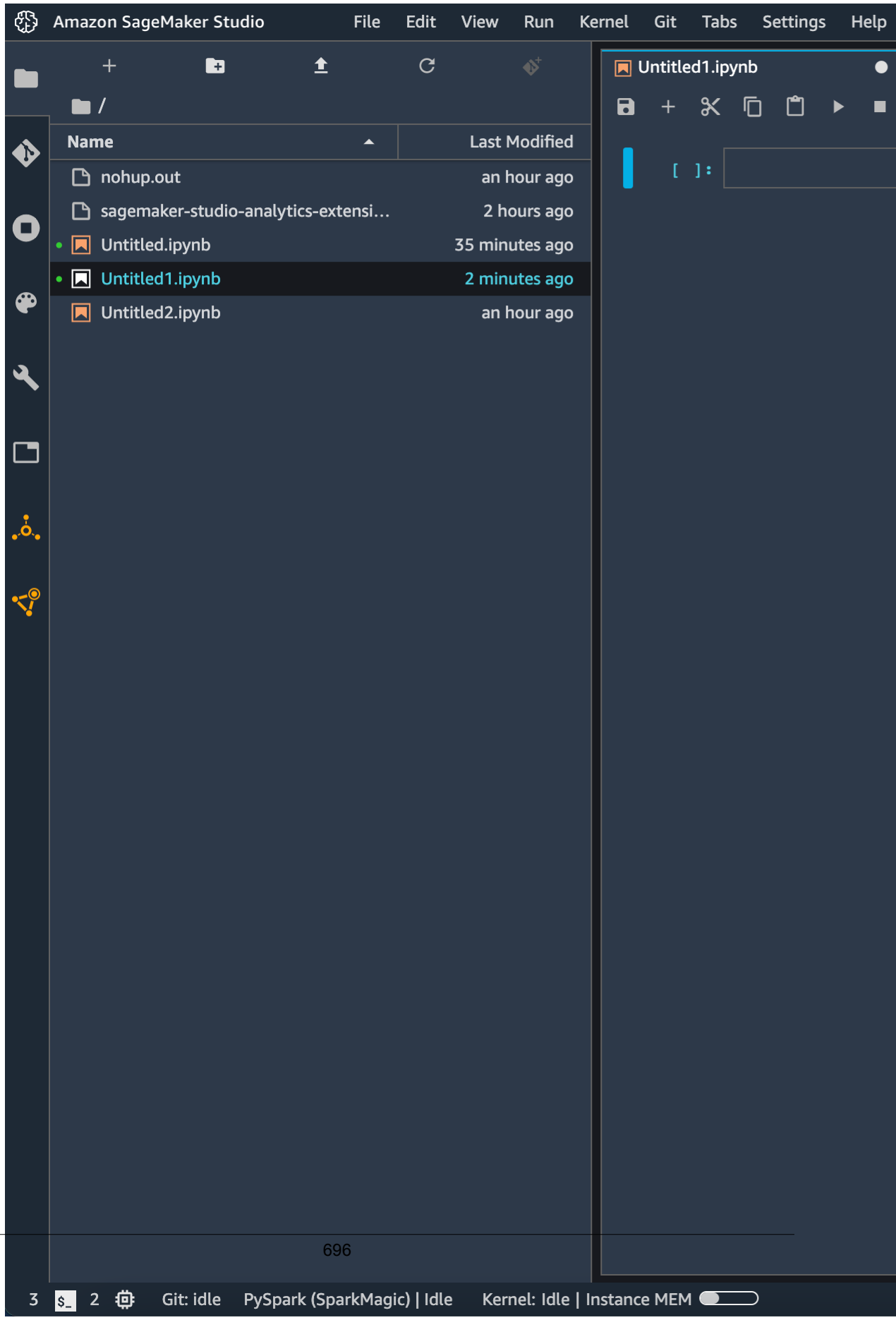
要将亚马逊 EMR 集群与已选择 PySpark 内核连接起来

1. 连接到 Studio 后，如果您有现有的 Studio 笔记本实例，请打开该实例。否则，要创建新的笔记本实例，请选择File (文件)选择，然后选择New.
2. 打开 Studio 笔记本实例后，请选择内核和实例。

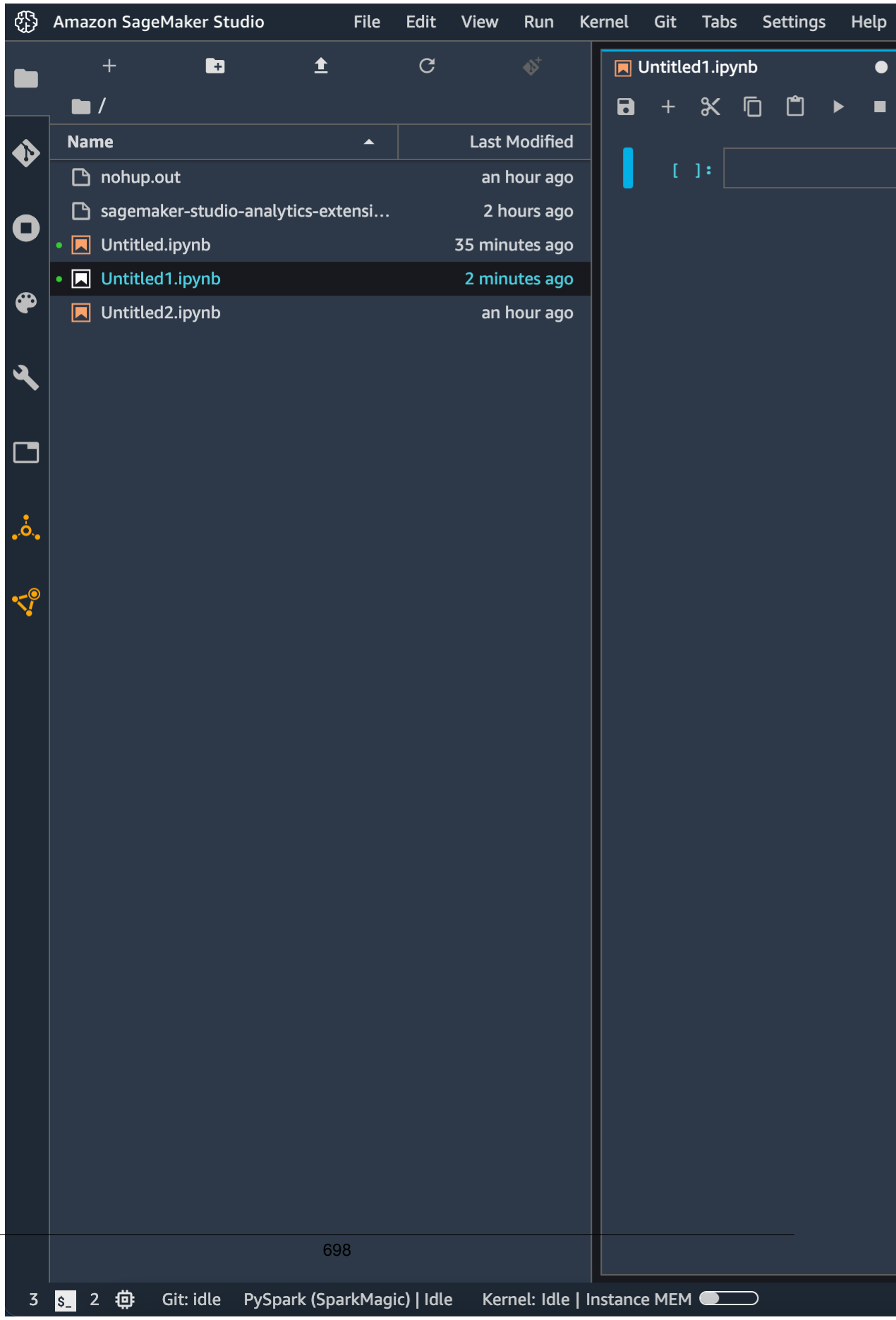
   Note

   只有内核的子集可以连接到 Amazon EMR 集群。支持的图像是数据科学和 SparkMagic。支持的内核是来自 SparkMagic 图像的 PySpark 和数据科学图像中的 Python3 (iPython) 。Studio 同时支持 PySpark 和斯卡拉内核。

   要切换内核，请在 UI 右上角选择出现弹出窗口的当前选定内核。然后从内核下拉菜单中选择您选择的内核。最后，选择Select按钮进行更改。

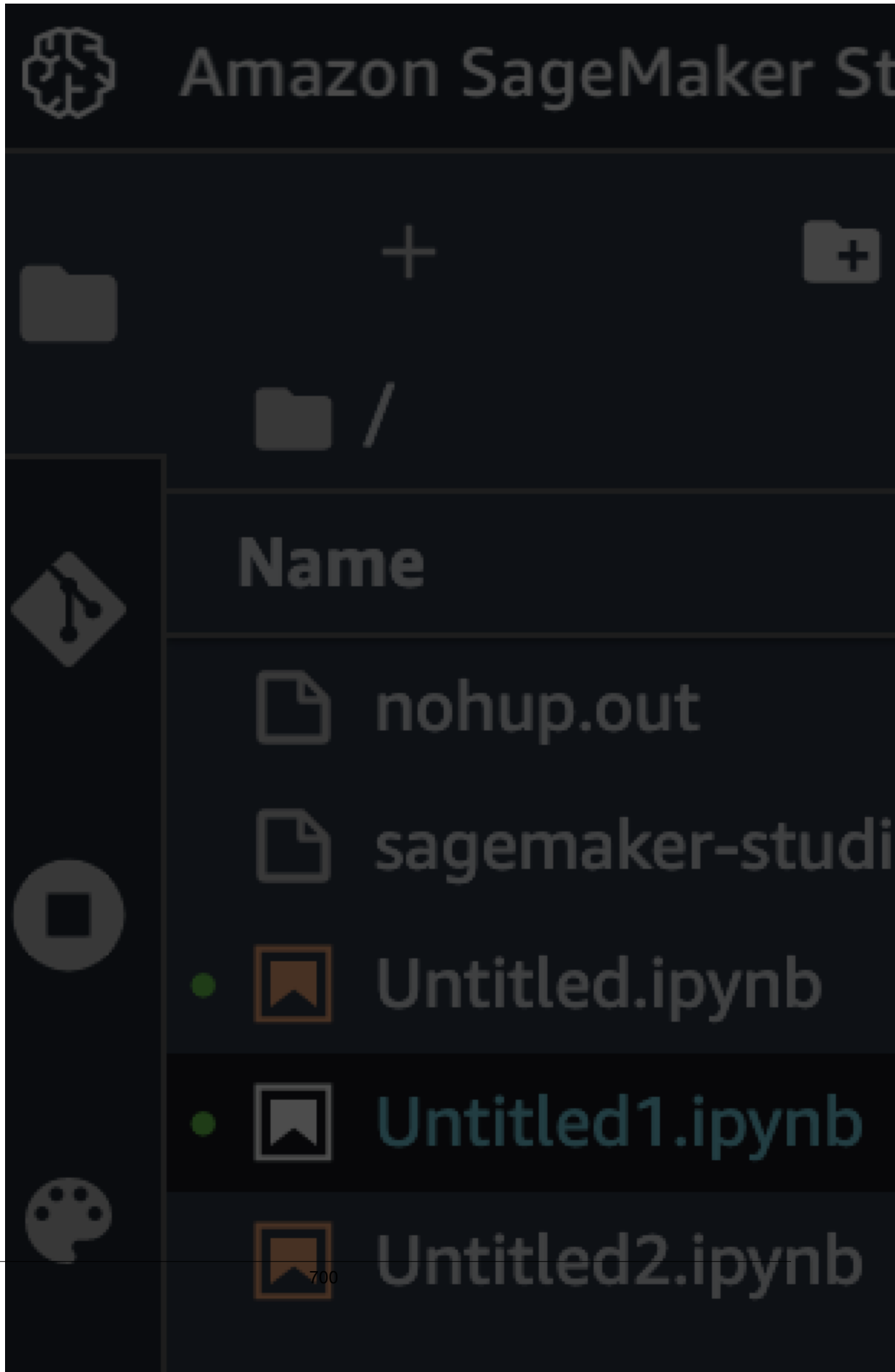

Amazon SageMaker Studio
File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
/
Name
Last Modified
nohup.out
an hour ago
sagemaker-studio-analytics-extensi...
2 hours ago
Untitled.ipynb
35 minutes ago
Untitled1.ipynb
2 minutes ago
Untitled2.ipynb
an hour ago
Untitled1.ipynb
[ ]:
3
2
Git: idle
PySpark (SparkMagic) | Idle
Kernel: Idle | Instance MEM

3. 选择自己选择的内核之后，选择群集.

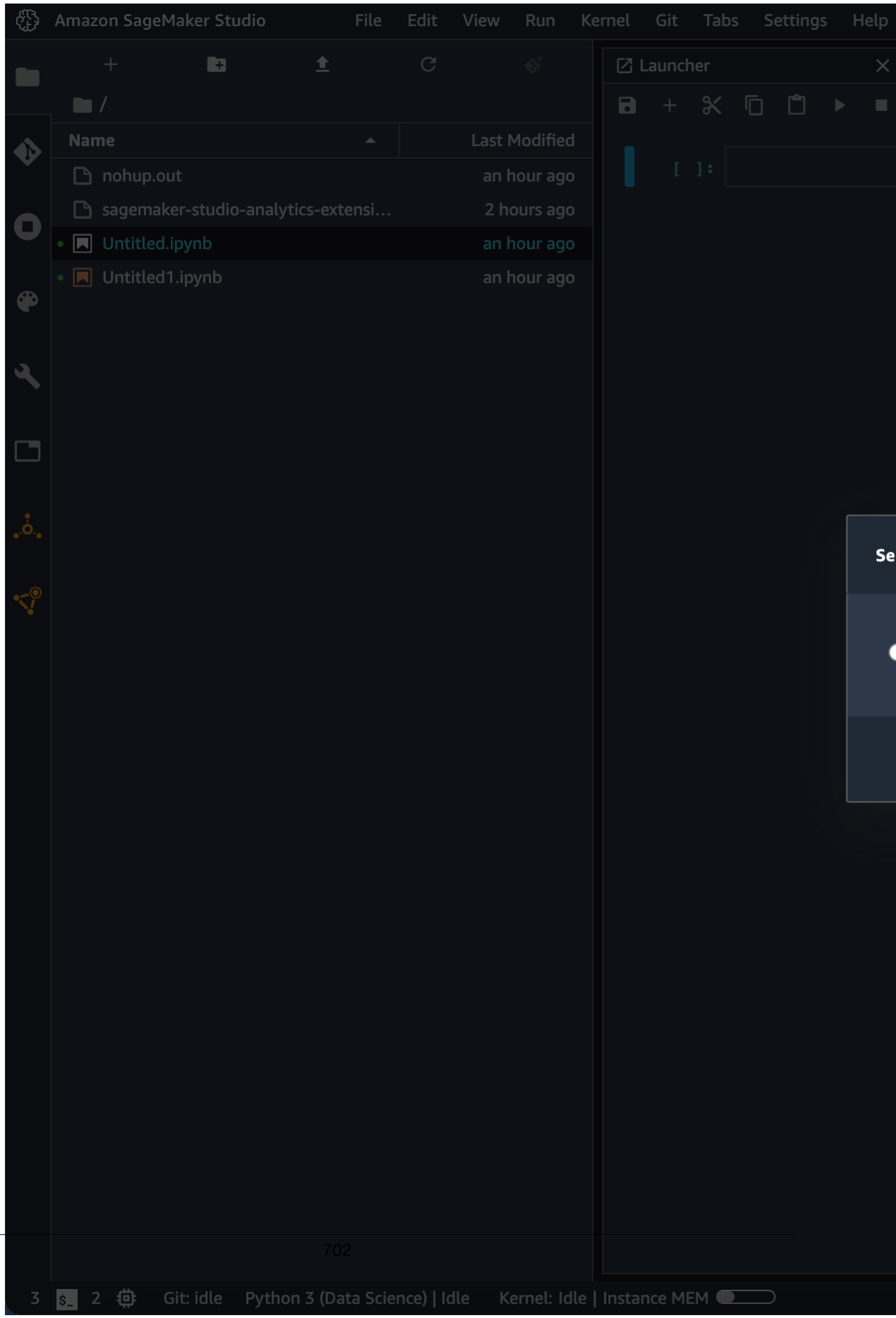

Amazon SageMaker Studio
File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
/
Name
Last Modified
nohup.out
an hour ago
sagemaker-studio-analytics-extensi...
2 hours ago
Untitled.ipynb
35 minutes ago
Untitled1.ipynb
2 minutes ago
Untitled2.ipynb
an hour ago
Untitled1.ipynb
[ ]:
3
2
Git: idle
PySpark (SparkMagic) | Idle
Kernel: Idle | Instance MEM

4. 一个Connect 到集群此时会显示 UI 屏幕。选择群集然后选择Connect (连接). 并非所有的 Amazon EMR 集群都可以连接到 Studio。有关更多信息，请参阅 。在 Studio 笔记本中使用 Spark 执行交互式数据处.

   1. 当您连接到群集时，它会向活动单元添加一个代码块以建立连接。

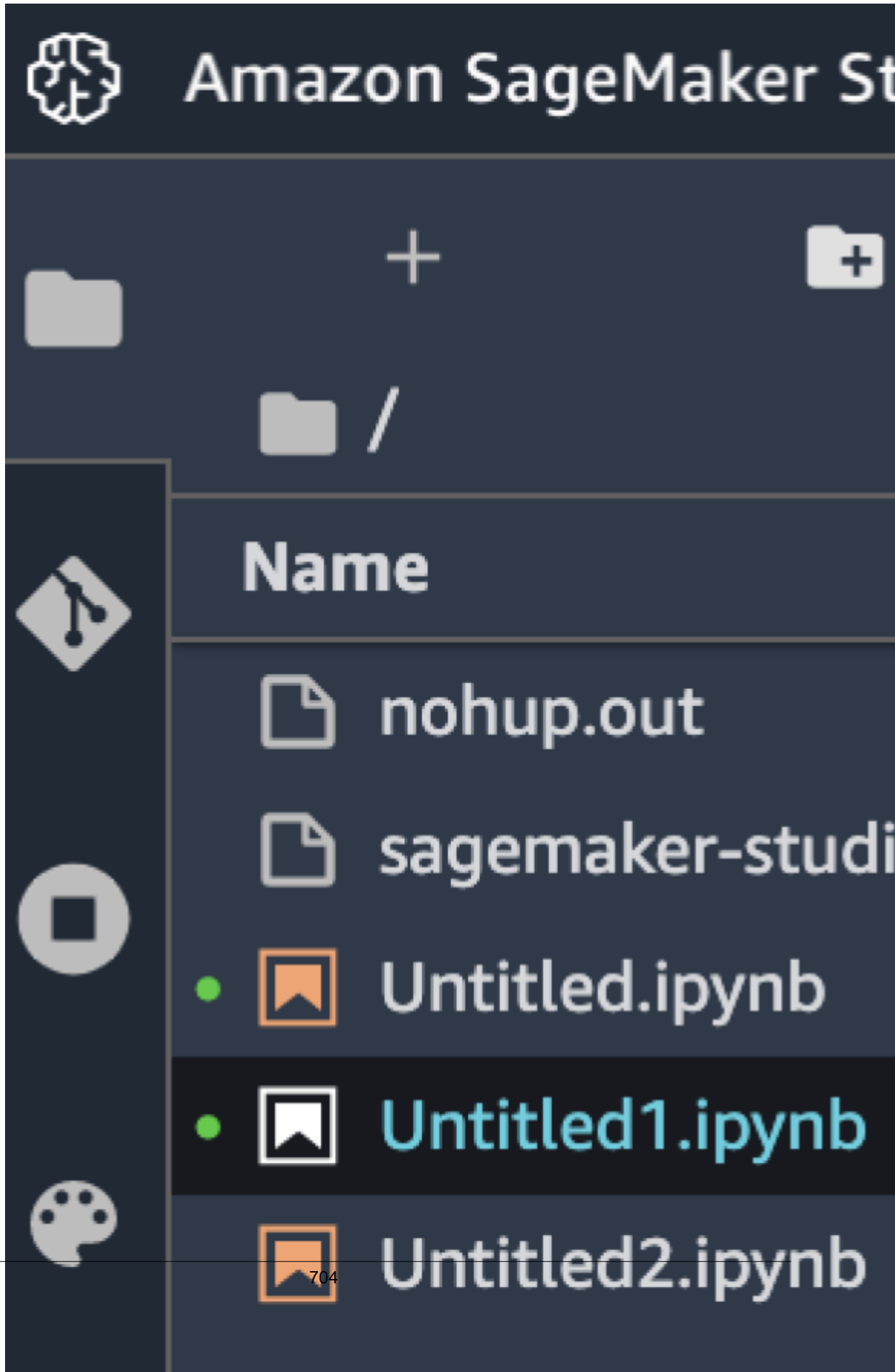

Amazon SageMaker St
/
Name
nohup.out
sagemaker-studi
Untitled.ipynb
Untitled1.ipynb
Untitled2.ipynb

5. 如果要连接的群集没有使用 Kerberos 或轻量级目录访问协议 (LDAP) 连接，系统将提示您选择凭据类型。您可以选择HTTP 基本身份验证要么无凭证.

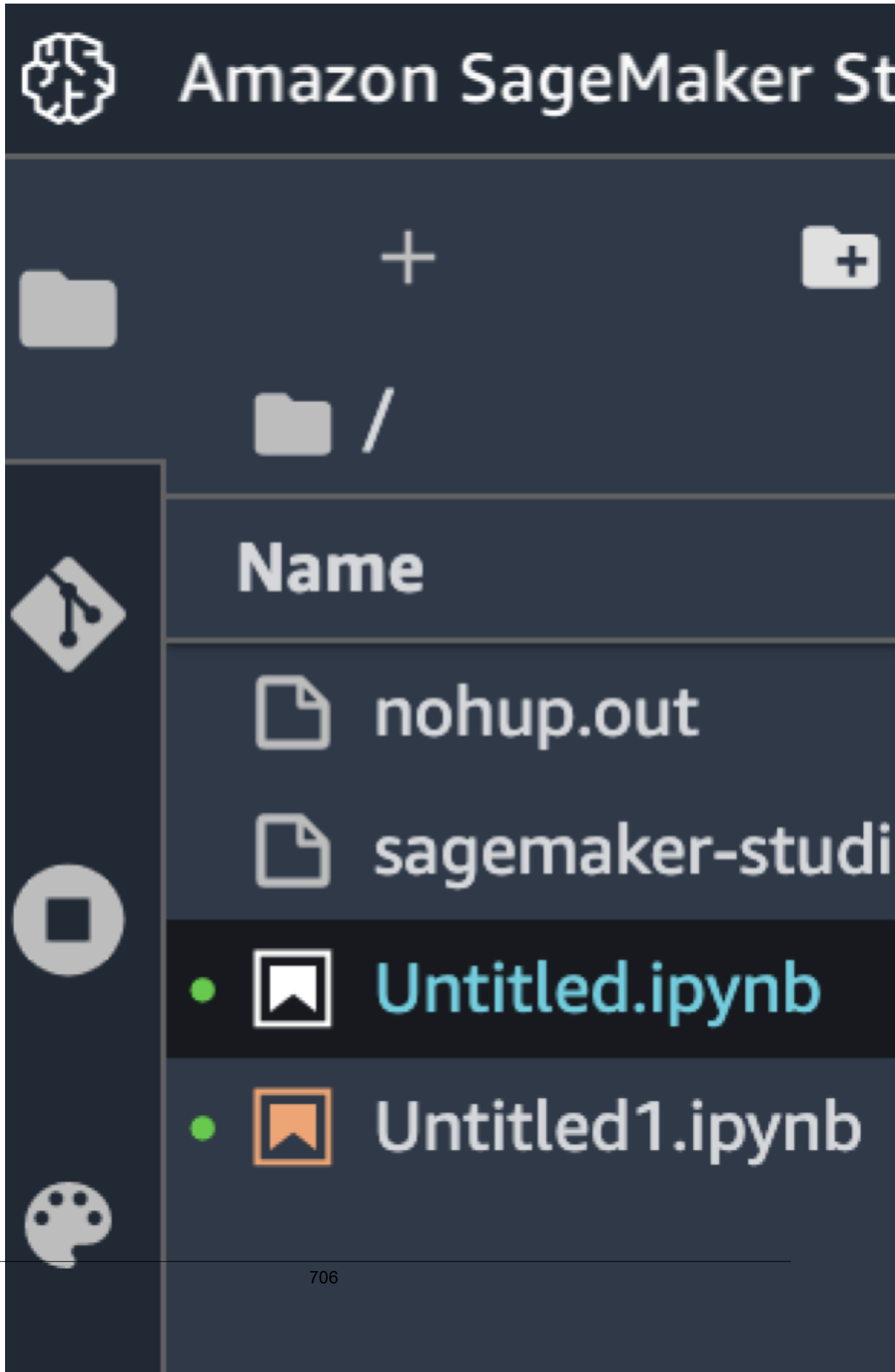

Amazon SageMaker Studio
File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
Launcher
/
Name
Last Modified
nohup.out
an hour ago
sagemaker-studio-analytics-extensi...
2 hours ago
Untitled.ipynb
an hour ago
Untitled1.ipynb
an hour ago
[ ]:
Se
3
2
Git: idle
Python 3 (Data Science) | Idle
Kernel: Idle | Instance MEM

6. 将填充活动单元格。这将包含连接到您选择的 Amazon EMR 集群所需的连接信息。

    1. 当身份验证类型为 Kerberos 和 HTTP Basic Auth 时，将在活动单元格中创建一个小部件供您提供用户名和密码. 以下屏幕截图显示了输入这些凭据后成功连接。


Amazon SageMaker St
/
Name
nohup.out
sagemaker-studi
Untitled.ipynb
Untitled1.ipynb
Untitled2.ipynb

2. 如果要连接到的群集不使用 Kerberos 或 LDAP，并且您选择了`No credentials`，您将自动连接到 Amazon EMR 集群。以下屏幕截图显示了成功输入凭据后的 UI。

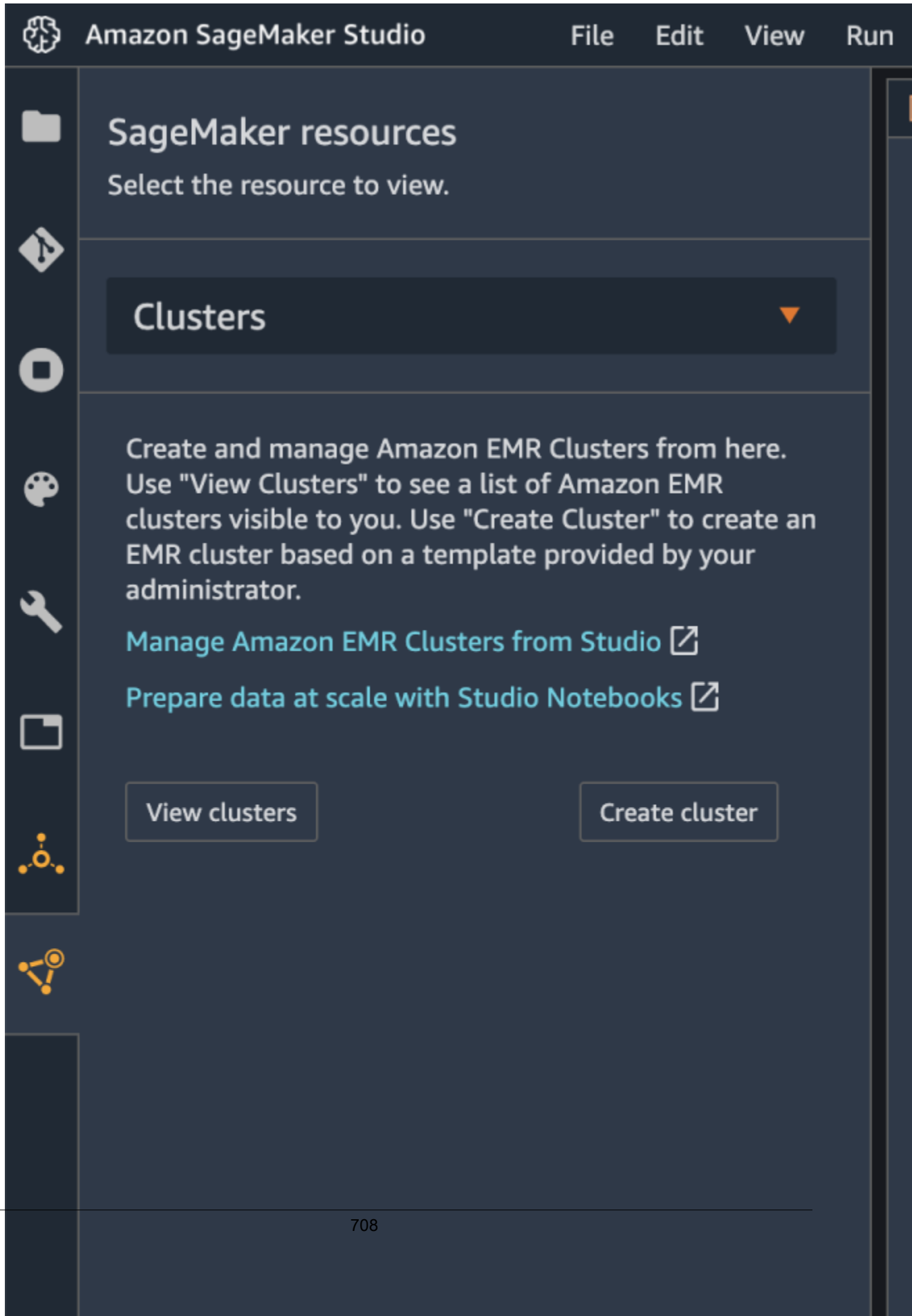

Amazon SageMaker St
/
Name
nohup.out
sagemaker-studi
Untitled.ipynb
Untitled1.ipynb

7. • 此为可选步骤。如果要更改 Studio 笔记本电脑连接到的 Amazon EMR 集群，请选择群集位于笔记本电脑的右上方。选择之后群集中，浏览集群列表并选择其他集群。

有关所需权限的更多信息，请参阅。所需权限 (p. 725).

跨账户 Connect 亚马逊 EMR 群集

如果您已经设置了跨账户可发现性和连接性，则选择群集，将显示来自 Studio 和远程帐户的所有集群。选择之后Connect (连接)，Studio 将在远程账户中启动并建立与 Amazon EMR 集群的连接。以下屏幕截图显示此连接。


Amazon SageMaker Studio
File
Edit
View
Run
SageMaker resources
Select the resource to view.
Clusters
Create and manage Amazon EMR Clusters from here. Use "View Clusters" to see a list of Amazon EMR clusters visible to you. Use "Create Cluster" to create an EMR cluster based on a template provided by your administrator.
Manage Amazon EMR Clusters from Studio
Prepare data at scale with Studio Notebooks
View clusters
Create cluster

在 Amazon EMR 中排除和监控工作负载

以下各节提供了从 SageMaker Studio 笔记本访问 Spark UI 的说明。Spark UI 允许您从 Studio 笔记本电脑监控和调试提交在亚马逊 EMR 上运行的 Spark 作业。SSH 隧道和预签名 URL 是访问 Spark UI 的两种方式。

为 Spark UI 访问设置 SSH 隧道

要设置 SSH 隧道以访问 Spark UI，请按照本节中的两个选项之一进行操作。请注意，第 6b 步中的屏幕截图从 Studio Connect 到 Amazon EMR 集群 (p. 695)显示 Spark UI 和驱动程序日志下的链接。只有在完成 SSH 隧道设置后，这些链接才会激活。

设置 SSH 隧道的选项：

- 选项 1：使用本地端口转发设置通往主节点的 SSH 隧道
- 备选方案 2，第 1 部分：使用动态端口转发设置通往主节点的 SSH 隧道
  - 备选方案 2，第 2 部分：配置代理设置以查看主节点上托管的网站。

有关查看 Amazon EMR 集群上托管的 Web 界面的信息，请参阅。查看 Amazon EMR 集群上托管的 Web 界面. 您还可以访问您的亚马逊 EMR 控制台以访问 Spark UI。

> Note
>
> 即使您不可用预签名的 URL，也可以设置 SSH 隧道。

预签名 URL

要创建可以从 SageMaker Studio 笔记本中访问 Amazon EMR 上的 Spark UI 的一键 URL，您必须启用以下 IAM 权限。选择适用于你的选项：

- 对于与 SageMaker Studio 笔记本相同的账户中的 Amazon EMR 集群：向 SageMaker Studio IAM 执行角色添加以下权限。
- 对于位于不同账户（不是 SageMaker Studio 笔记本电脑）中的亚马逊 EMR 集群：将以下权限添加到您为其创建的跨账户角色：从 Studio 发现亚马逊 EMR 集群 (p. 692).

> Note
>
> 您可以从以下区域的控制台访问预签名 URL：
>
> - 美国东部（弗吉尼亚北部）区域
> - 美国西部（加利福尼亚北部）区域
> - 加拿大（中部）区域
> - 欧洲（法兰克福）区域
> - 欧洲（斯德哥尔摩）区域
> - 欧洲（爱尔兰）区域
> - 欧洲（伦敦）区域
> - 欧洲（巴黎）区域
> - 亚太区域（东京）
> - 亚太区域（首尔）
> - 亚太区域（悉尼）
> - 亚太（孟买）区域
> - 亚太区域（新加坡）
> - 南美洲（圣保罗）

以下策略允许对执行角色的预签名 URL 的访问权限。

```
{
            "Sid": "AllowPresignedUrl",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:CreatePersistentAppUI",
                "elasticmapreduce:DescribePersistentAppUI",
                "elasticmapreduce:GetPersistentAppUIPresignedURL",
                "elasticmapreduce:GetOnClusterAppUIPresignedURL"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:<region>:<account-id>:cluster/*"
            ]
}
```

有关所需权限的更多信息，请参阅所需权限 (p. 725)。

从 Studio 管理亚马逊 EMR 集群

管理员可以使用Amazon Service Catalog来定义模板。这些模板允许您创建 Amazon EMR 集群并将其提供给所选用户。使用AmazonService Catalog，管理员可以完全控制 Amazon EMR 群集中的组织、安全性和网络。管理员可以查看、选择和自定义特定工作负载的模板。如果他们选择，管理员可以限制对所选配置参数的曝光，例如仅限于特定的数据工作者。管理员可以在 SageMaker Studio 中创建按需亚马逊 EMR 集群。这可以在不手动设置复杂的配置的情况下完成。用户还可以在使用后直接从 SageMaker Studio 终止 Amazon EMR 集群。

本指南中的主题提供有关设置和使用的更多详细信息。Amazon Service Catalog模板，以及创建和终止 Amazon EMR 集群。

主题

- 配置Amazon Service Catalog产品 (p. 710)
- 从 Studio 预配亚马逊 EMR 集群 (p. 713)

配置Amazon Service Catalog产品

在本指南中，您将获得用于配置的参数Amazon Service Catalog产品，以便您可以发现 CloudFormation (CFN) 模板。CFN 模板用于创建和管理 Amazon EMR 集群。这些区域有：Amazon Service Catalog产品可以与Amazon Service Catalog与 Studio 执行角色 IAM 实体共享的产品组合。

要从 Studio 启用 Amazon EMR 集群管理，您需要一个包含 Amazon EMR 集群详细信息和可配置参数的 CloudFormation (CFN) 模板。有关如何使用 CloudFormation 的信息，请参阅。入门 CloudFormation. 有关的更多信息Amazon Service Catalog产品，请参阅步骤 4: 创建AmazonService Catalog.

这些区域有：Amazon Service Catalog具有亚马逊 EMR 资源的商品必须具有以下标签：

```
sagemaker:studio-visibility:emr true
```

中的 CFN 模板Amazon Service Catalog产品必须具有以下强制性堆栈参数：

```
SageMakerProjectName:
Type: String
Description: Name of the project

SageMakerProjectId:
```

```
Type: String
Description: Service generated Id of the project.
```

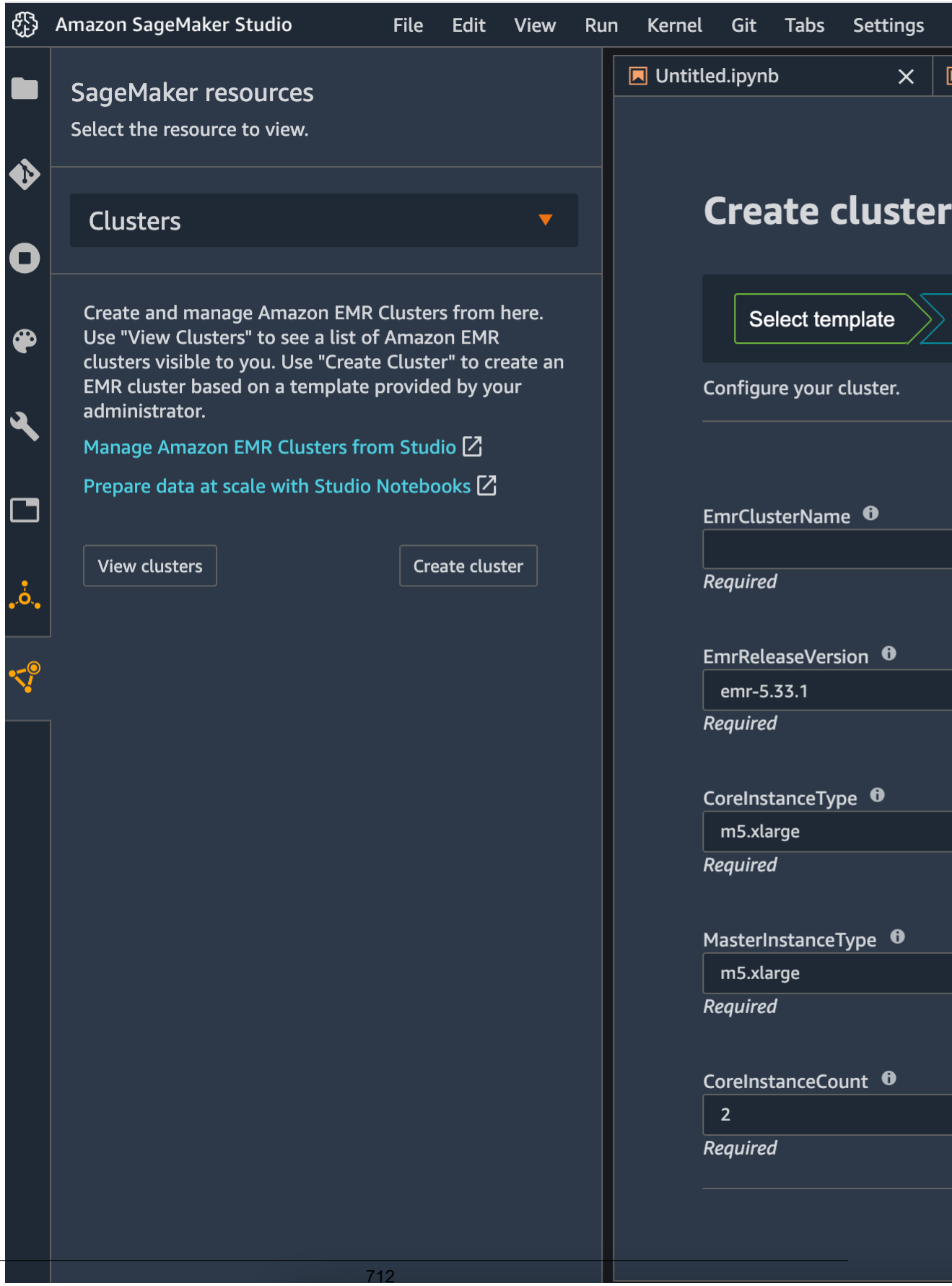
有关创建的更多信息Amazon Service Catalog产品组合和产品，请参阅步骤 3: 创建AmazonService Catalog及其后面的章节。

可选模板参数

从 SageMaker Studio 中创建 Amazon EMR 模板时，您可以在模板的参数部分添加额外选项。本节允许用户为集群输入或选择自定义值。以下示例参数定义了在 Studio 内创建 Amazon EMR 模板时可以使用的其他输入参数。

```
"Parameters": {
    "EmrClusterName": {
      "Type": "String",
      "Description": "EMR cluster Name."
    },
    "MasterInstanceType": {
      "Type": "String",
      "Description": "Instance type of the EMR master node.",
      "Default": "m5.xlarge",
      "AllowedValues": [
        "m5.xlarge",
        "m5.2xlarge",
        "m5.4xlarge"
      ]
    },
    "CoreInstanceType": {
      "Type": "String",
      "Description": "Instance type of the EMR core nodes.",
      "Default": "m5.xlarge",
      "AllowedValues": [
        "m5.xlarge",
        "m5.2xlarge",
        "m5.4xlarge",
        "m3.medium",
        "m3.large",
        "m3.xlarge",
        "m3.2xlarge"
      ]
    },
    "CoreInstanceCount": {
      "Type": "String",
      "Description": "Number of core instances in the EMR cluster.",
      "Default": "2",
      "AllowedValues": [
        "2",
        "5",
        "10"
      ]
    },
    "EmrReleaseVersion": {
      "Type": "String",
      "Description": "The release version of EMR to launch.",
      "Default": "emr-5.33.1",
      "AllowedValues": [
        "emr-5.33.1",
        "emr-6.4.0"
      ]
    }
  }
```

下图显示了从 Studio 创建 Amazon EMR 集群时 Studio UI 的外观。


Amazon SageMaker Studio
File
Edit
View
Run
Kernel
Git
Tabs
Settings
SageMaker resources
Select the resource to view.
Clusters
Create and manage Amazon EMR Clusters from here. Use "View Clusters" to see a list of Amazon EMR clusters visible to you. Use "Create Cluster" to create an EMR cluster based on a template provided by your administrator.
Manage Amazon EMR Clusters from Studio
Prepare data at scale with Studio Notebooks
View clusters
Create cluster
Untitled.ipynb
Create cluster
Select template
Configure your cluster.
EmrClusterName
Required
EmrReleaseVersion
emr-5.33.1
Required
CoreInstanceType
m5.xlarge
Required
MasterInstanceType
m5.xlarge
Required
CoreInstanceCount
2
Required

从 Studio 预配亚马逊 EMR 集群

本节提供了带有 Studio 的屏幕截图的指导说明，其中展示了如何创建群集、查看可用集群列表以及终止集群。

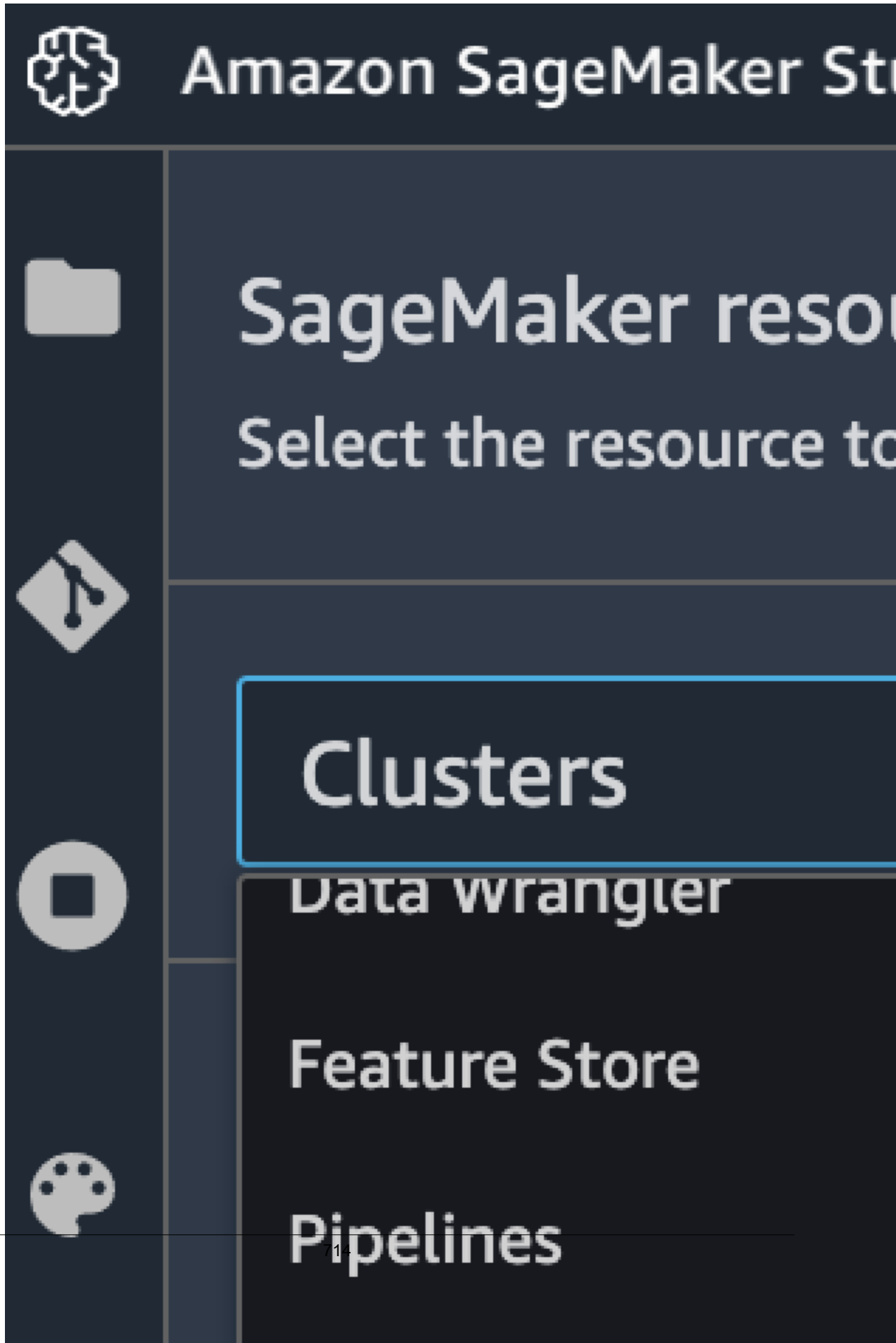
如果配置了跨账户发现，则会在 Studio 和远程账户中看到集群的综合列表。

首先，请向将访问 Studio 的 IAM 执行角色添加以下权限。有关所需权限的更多信息，请参阅所需权限 (p. 725)。

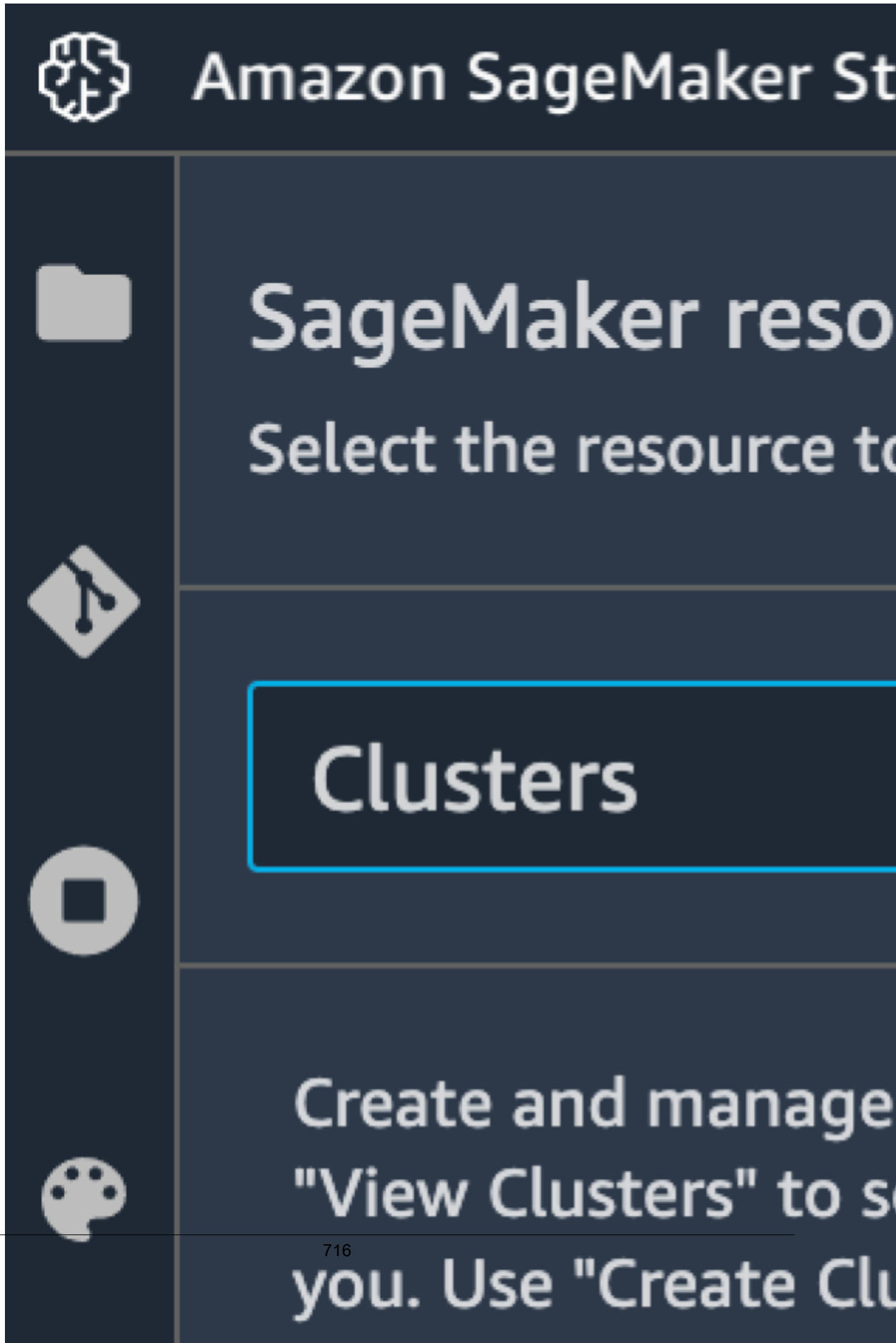
```
{
            "Sid": "AllowSagemakerProjectManagement",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateProject",
                "sagemaker:DeleteProject"
            ],
            "Resource": "arn:aws:sagemaker:<region>:<account-id>:project/*"
}
```

在添加以前的权限后，您可以连接到 Studio。如果您有现有的 Studio 笔记本实例，则可以打开它。否则，如果要创建新的笔记本实例，请选择File (文件)选择，然后选择New.

- 要打开群集管理页面，请选择集群从左侧面板上。

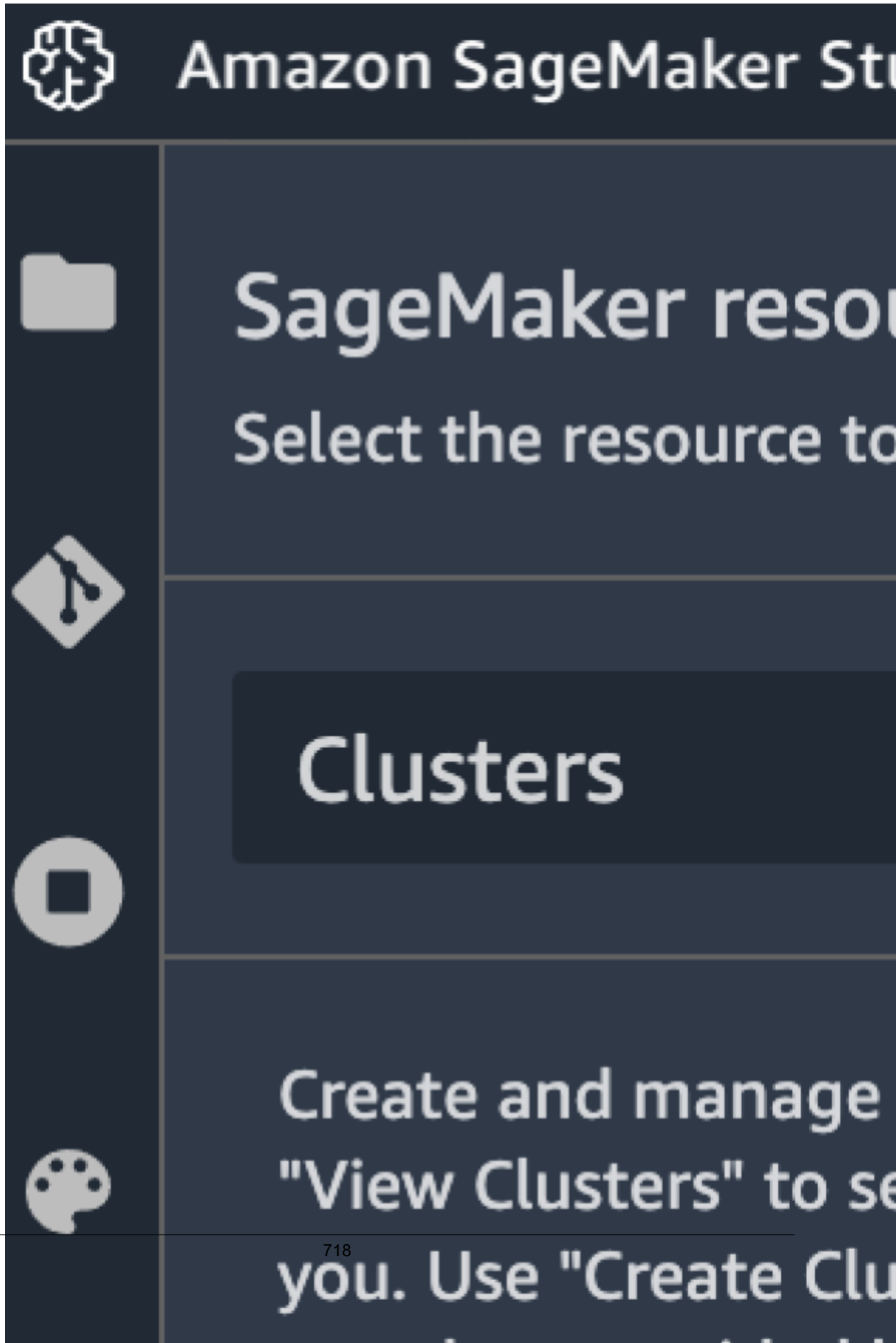

Amazon SageMaker St
SageMaker reso
Select the resource to
Clusters
Data Wrangler
Feature Store
Pipelines

以下屏幕截图显示集群管理页面。从此处，您可以创建和管理 Amazon EMR 集群。


Amazon SageMaker St
SageMaker reso
Select the resource to
Clusters
Create and manage
"View Clusters" to s
you. Use "Create Clu

集群可以处于不同的状态。要按状态筛选集群，请选择下拉箭头图标。集群一次只能处于一个状态。

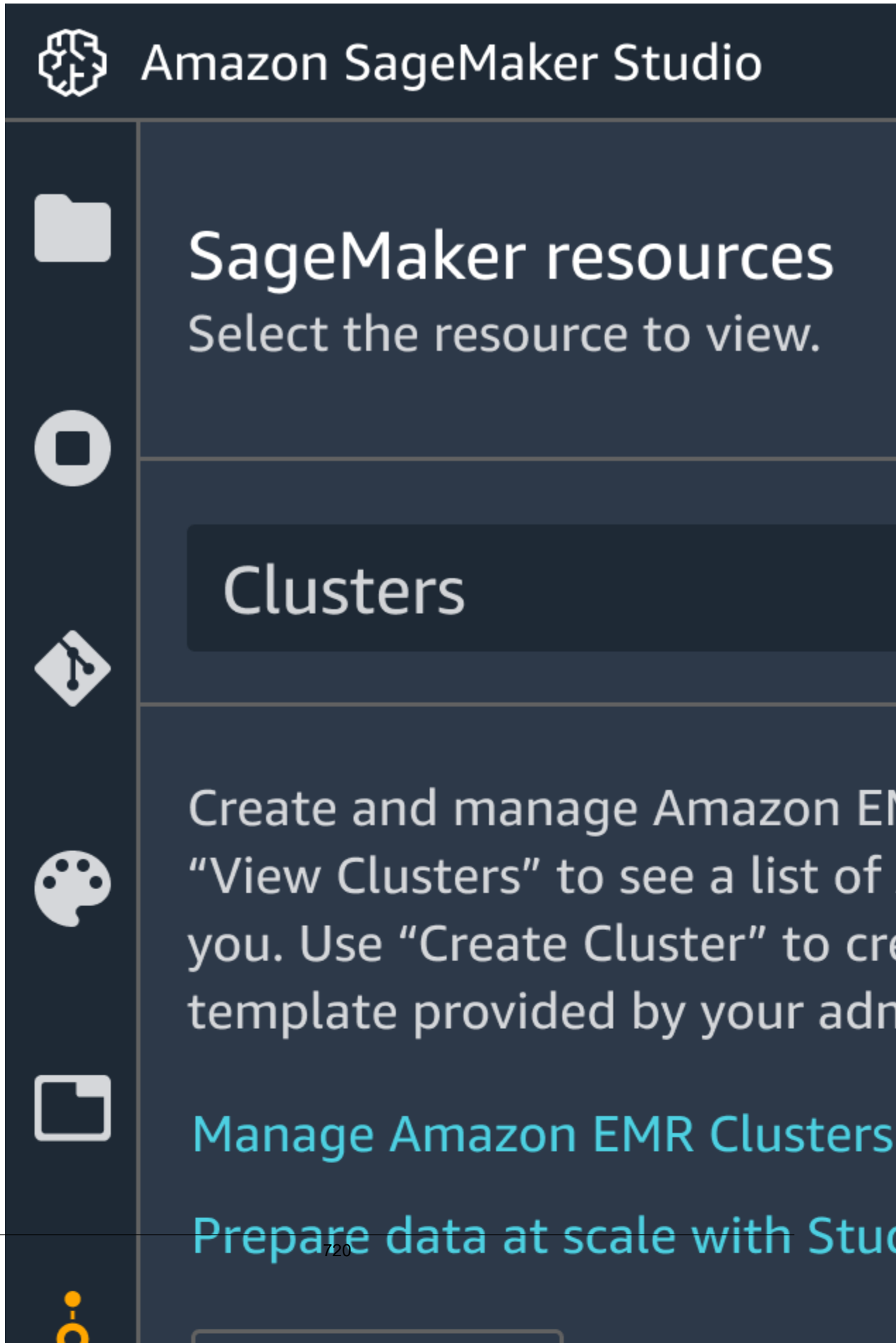
状态选项包括 “开始”、“引导”、“跑步/步行”、“终止”、“终止” 和 “因错误而终止”。以下屏幕截图显示了状态下拉列表中的集群状态选项。


Amazon SageMaker Stu
SageMaker reso
Select the resource to
Clusters
Create and manage
"View Clusters" to se
you. Use "Create Clu

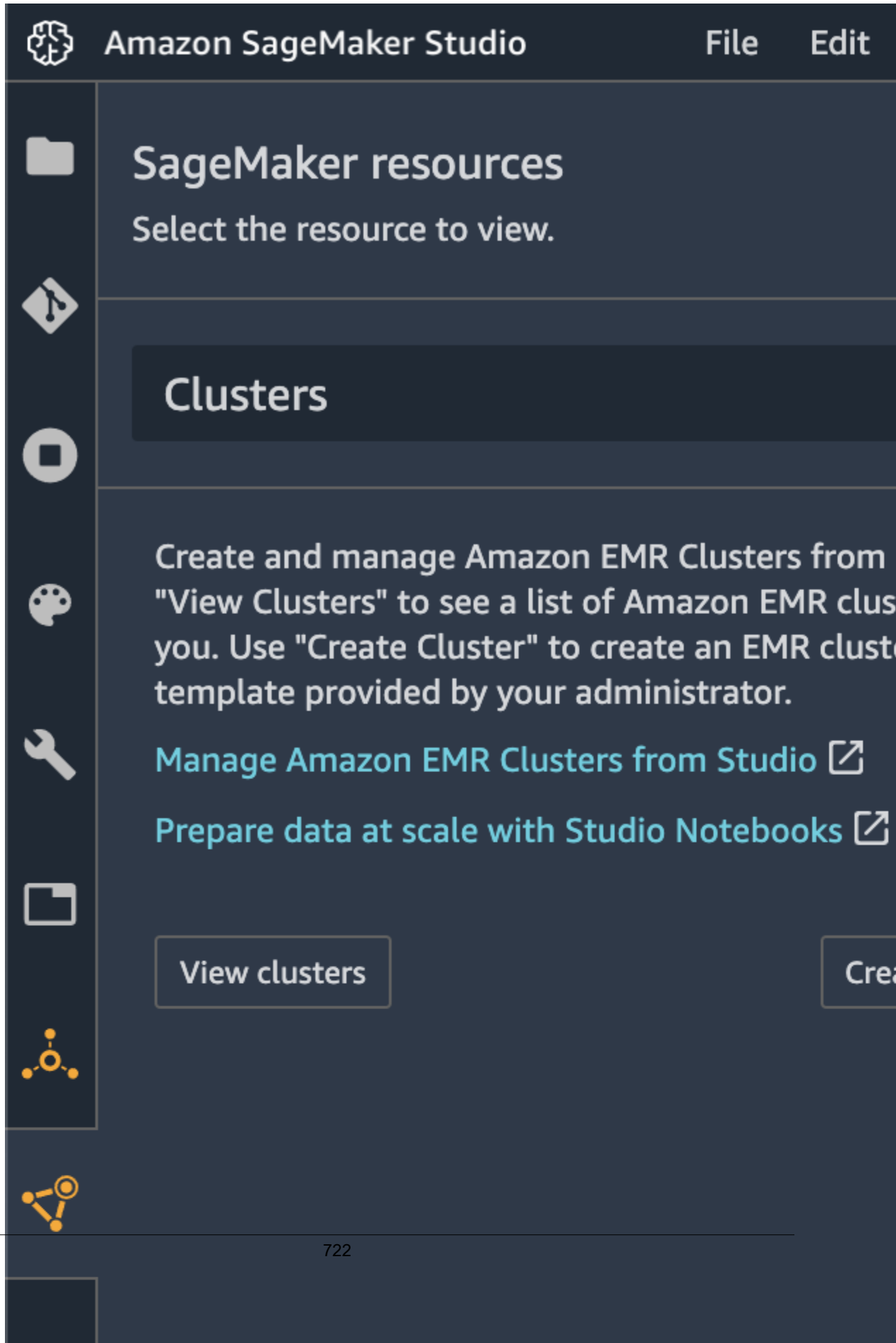
以下过程将介绍如何使用 Studio 中的模板创建集群。

要从模板创建集群：

1. 通过选择集群从左侧面板上。然后，选择创建集群.


Amazon SageMaker Studio
SageMaker resources
Select the resource to view.
Clusters
Create and manage Amazon E
“View Clusters” to see a list of
you. Use “Create Cluster” to cr
template provided by your adn
Manage Amazon EMR Clusters
Prepare data at scale with Stu

2. 输入您的 SageMaker Studio 子网 ID、亚马逊 EMR 群集名称、亚马逊 VPC ID 和 SageMaker Studio 安全组 ID。


Amazon SageMaker Studio
File
Edit
SageMaker resources
Select the resource to view.
Clusters
Create and manage Amazon EMR Clusters from
"View Clusters" to see a list of Amazon EMR clus
you. Use "Create Cluster" to create an EMR cluste
template provided by your administrator.
Manage Amazon EMR Clusters from Studio
Prepare data at scale with Studio Notebooks
View clusters
Crea

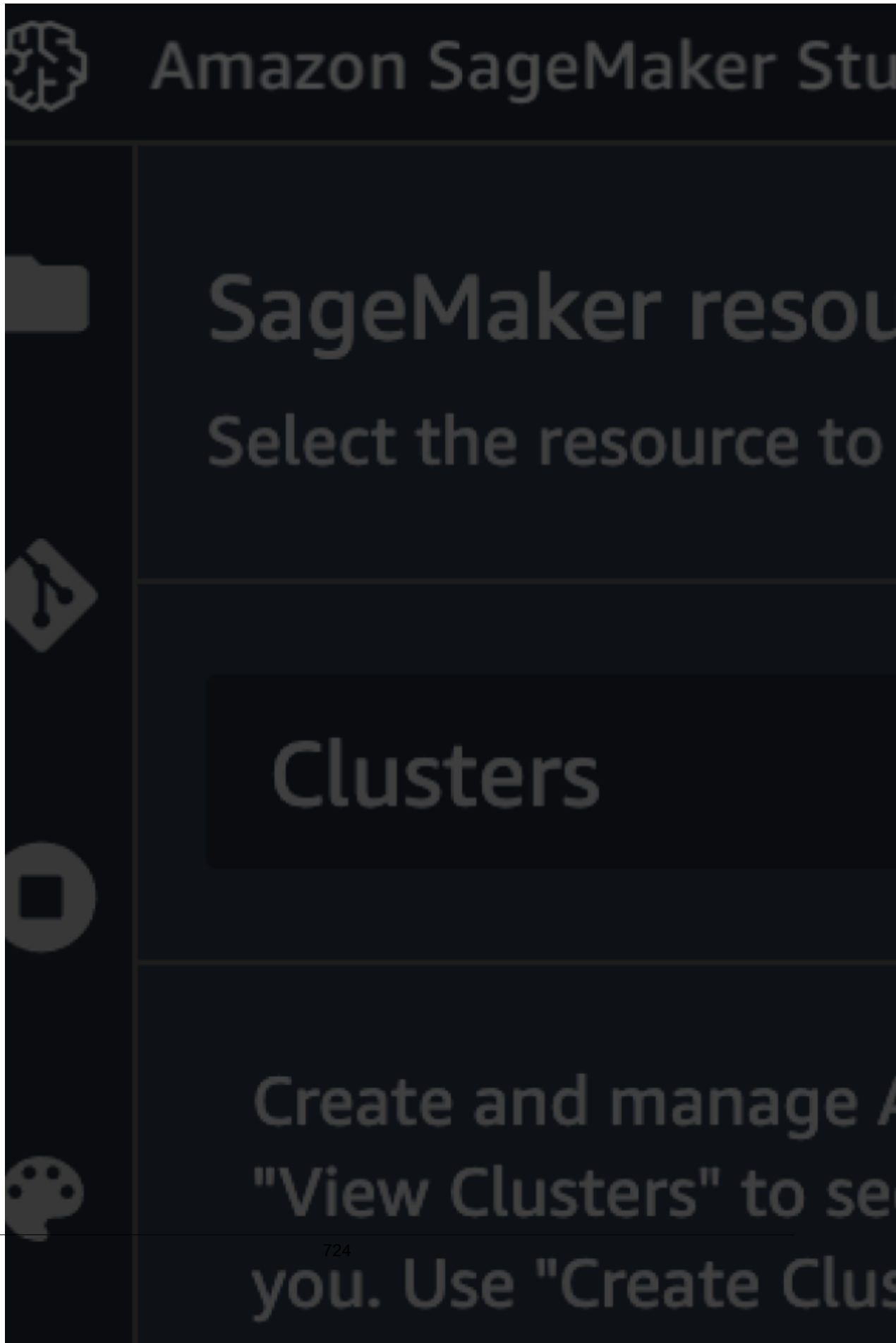
3. 输入上一步中的信息后，选择创建集群.

以下过程演示如何终止集群。

终止集群

1. 通过选择集群从左侧面板上。

   选择要终止的集群，然后选择。终止旁边的创建集群在 UI 中。
2. 将出现一个窗口，提示集群上的任何待处理工作或数据将在终止后丢失，并且终止是不可逆转的。Select终止.


Amazon SageMaker Stu
SageMaker resou
Select the resource to
Clusters
Create and manage A
"View Clusters" to se
you. Use "Create Clus

所需权限

本指南介绍如何将 IAM 策略示例应用于您的执行角色。本指南中的示例为您提供了各种权限和监控功能。在本指南的最后，您将找到一个全面的 IAM 策略角色，其中包括所有必需的权限。这些权限提供了从 Studio 内访问 Amazon EMR 群集的权限，并允许从 Studio 内发现群集（用于预签名 URL）。访问IAM 控制台将策略添加到角色。

要从 Studio 中发现并连接到 Amazon EMR 集群，请确保您的执行角色具有以下权限：

```
{
            "Sid": "AllowClusterDetailsDiscovery",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:<region>:<account-id>:cluster/*"
            ]
        },
        {
            "Sid": "AllowClusterDiscovery",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ListClusters"
            ],
            "Resource": "*"
}
```

要增强监控功能，请将以下权限附加到您的 Studio 执行角色：

```
{
            "Sid": "AllowPresignedUrl",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:CreatePersistentAppUI",
                "elasticmapreduce:DescribePersistentAppUI",
                "elasticmapreduce:GetPersistentAppUIPresignedURL",
                "elasticmapreduce:GetOnClusterAppUIPresignedURL"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:<region>:<account-id>:cluster/*"
            ]
}
```

要允许用户创建 Amazon EMR 集群，请向您的 Studio 执行角色添加以下权限：

```
{
            "Sid": "AllowSagemakerProjectManagement",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateProject",
                "sagemaker:DeleteProject"
            ],
            "Resource": "arn:aws:sagemaker:<region>:<account-id>:project/*"
}
```

如果您向 Studio 提供了跨账户发现和连接角色，则非 Studio 帐户中的角色必须允许对 Studio 执行角色的“代入角色”权限。可以通过向 Studio 执行角色附加以下权限来完成此操作：

Note

对于所有跨账户使用案例，Studio 执行角色必须具有`sts:AssumeRole`添加了允许 Studio 在远程帐户中担任角色的权限。

```
{
            "Sid": "AllowRoleAssumptionForCrossAccountDiscovery",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": ["arn:aws:iam::<cross-account>:role/<studio-execution-role>" ]
}
```

要允许 Amazon EMR 模板的可发现性，请将以下权限附加到您的 Studio 执行角色：

```
{
            "Sid": "AllowEMRTemplateDiscovery",
            "Effect": "Allow",
            "Action": [
              "servicecatalog:SearchProducts"
            ],
            "Resource": "*"
}
```

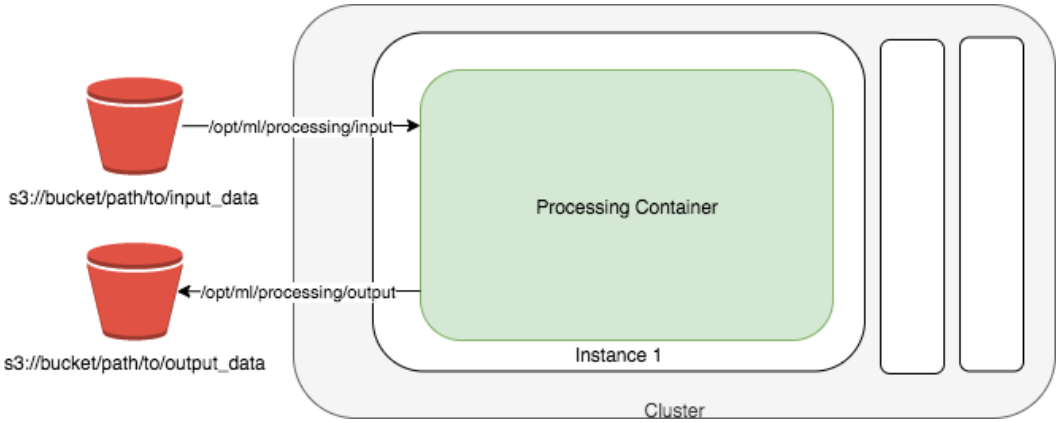
以下是一个全面的示例 IAM 策略，其中包括本指南中之前的所有权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPresignedUrl",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:CreatePersistentAppUI",
                "elasticmapreduce:DescribePersistentAppUI",
                "elasticmapreduce:GetPersistentAppUIPresignedURL",
                "elasticmapreduce:GetOnClusterAppUIPresignedURL"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:<region>:<account-id>:cluster/*"
            ]
        },
        {
            "Sid": "AllowClusterDetailsDiscovery",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:<region>:<account-id>:cluster/*"
            ]
        },
        {
            "Sid": "AllowClusterDiscovery",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ListClusters"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "AllowSagemakerProjectManagement",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateProject",
                "sagemaker:DeleteProject"
            ],
            "Resource": "arn:aws:sagemaker:<region>:<account-id>:project/*"
        },
        {
            "Sid": "AllowRoleAssumptionForCrossAccountDiscovery",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": ["arn:aws:iam::<cross-account>:role/<studio-execution-role>" ]
        },
        {
            "Sid": "AllowEMRTemplateDiscovery",
            "Effect": "Allow",
            "Action": [
              "servicecatalog:SearchProducts"
            ],
            "Resource": "*"
        }
    ]
}
```

处理数据

要在 Amazon SageMaker 上分析数据并评估机器学习模型，请使用 Amazon SageMaker 处理。借助处理，您可以在 SageMaker 上使用简化的托管体验来运行数据处理工作负载，例如功能工程、数据验证、模型评估和模型解释。您还可以在实验阶段以及在生产中部署代码之后使用 Amazon SageMaker 处理 API 来评估性能。



上图显示了 Amazon SageMaker 如何启动处理任务。Amazon SageMaker 将您的脚本从 Amazon Simple Storage Service (Amazon S3) 复制数据，然后提取处理容器。处理的容器映像可以是 Amazon SageMaker 内置映像，也可以是您提供的自定义映像。处理作业的底层基础设施完全由 Amazon SageMaker 管理。集群资源将在作业期间预配置，并在作业完成时进行清理。处理作业的输出存储在您指定的 Amazon S3 存储桶中。

Note

您的输入数据必须存储在 Amazon S3 存储桶中。或者，您也可以将 Amazon Athena 或 Amazon Redshift 作为输入源。

使用 Amazon SageMaker 处理示例笔记本

我们提供两个示例 Jupyter 笔记本，以展示如何执行数据预处理、模型评估或这两者。

有关演示如何运行 scikit-learn 脚本以使用适用于处理的 SageMaker Python 开发工具包执行数据预处理以及模型训练和评估的示例笔记本，请参阅。scikit-learn 处理. 此笔记本还展示如何使用您自己的自定义容器，通过 Python 库和其他特定依赖项来运行处理工作负载。

有关演示如何使用 Amazon SageMaker 处理通过 Spark 执行分布式数据预处理的示例笔记本，请参阅。分布式处理（Spark）. 此笔记本还展示如何在预处理的数据集上使用 XGBoost 训练回归模型。

有关如何创建和访问可用于在 SageMaker 中运行这些示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 创建笔记本实例并打开后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

使用 CloudWatch Logs 和指标监控 Amazon SageMaker 处理作业

Amazon SageMaker 处理提供 Amazon CloudWatch 日志和指标来监控处理作业。CloudWatch 提供 CPU、GPU、内存、GPU 内存和磁盘指标以及事件日志记录。有关更多信息，请参阅 使用 Amazon

CloudWatch 监控 Amazon SageMaker (p. 2810) 和 使用 Amazon CloudWatch 记录 Amazon SageMaker 事件 (p. 2820)。

使用 Apache Spark 的数据处理

Apache Spark 是用于大规模数据处理的统一分析引擎。Amazon SageMaker 提供了预构建的 Docker 映像，其中包括 Apache Spark 和运行分布式数据处理作业所需的其他依赖项。使用Amazon SageMaker Python 开发工具包，您可以使用 Spark 框架轻松应用数据转换和提取功能（功能工程）。有关使用 SageMaker Python 开发工具包运行 Spark 处理作业的信息，请参阅使用 Spark 的数据处理中的Amazon SageMaker Python 开发工具包.

可在上找到包含 Spark 映像的源代码和 Dockerfile 的代码库。GitHub.

运行 Spark 处理 Job

您可以使用`sagemaker.spark.PySparkProcessor`要么`sagemaker.spark.SparkJarProcessor`类来在处理作业中运行 Spark 应用程序。注意，您可以将 MaxRuntimeInSeconds 设置为 5 天的最大运行时限制。关于执行时间和使用的实例数量，简单的火花工作负载可以看到实例数量与完成时间之间的近线性关系。

以下代码示例说明如何运行调用的处理作业： 你的 PySpark 脚本`preprocess.py`.

```
from sagemaker.spark.processing import PySparkProcessor

spark_processor = PySparkProcessor(
    base_job_name="spark-preprocessor",
    framework_version="2.4",
    role=role,
    instance_count=2,
    instance_type="ml.m5.xlarge",
    max_runtime_in_seconds=1200,
)

spark_processor.run(
    submit_app="preprocess.py",
    arguments=['s3_input_bucket', bucket,
               's3_input_key_prefix', input_prefix,
               's3_output_bucket', bucket,
               's3_output_key_prefix', output_prefix]
)
```

有关深入了解，请参阅使用 Apache Spark 进行的分布式数据处理和 SageMaker 处理示例笔记本.

如果不使用Amazon SageMaker Python 开发工具包以及其中一个处理器类来检索预构建映像，则可以自行检索这些映像。Amazon Elastic SageMaker tainer Registry（Amazon ECR））存储在 Amazon EElastic Container Registry (Amazon ECR)。有关可用的预构建 Docker 映像的完整列表，请参阅可用图片文档.

要了解有关将 SageMaker Python SDK 与处理容器结合使用的更多信息，请参Amazon SageMaker Python 开发工具包.

使用 scikit-learn 处理数据

要了解演示如何使用 SageMaker 提供和维护的 Docker 映像运行 scikit-learn 脚本以预处理数据和评估模型的示例笔记本，请参阅scikit-learn 处理. 要使用此笔记本，您需要安装适用于处理的 SageMaker Python 开发工具包。

这个笔记本运行处理作业`SKLearnProcessor`来运行您提供的 scikit-learn 脚本。此脚本预处理数据，使用 SageMaker 训练作业训练模型，然后运行处理作业来评估训练的模型。处理作业估计模型在生产中的预期效果。

要了解有关将 SageMaker Python SDK 与处理容器结合使用的更多信息，请参阅SageMaker Python SDK.

以下代码示例演示笔记本如何使用。`SKLearnProcessor`使用 SageMaker 提供和维护的 Docker 映像（而不是您自己的 Docker 映像）运行您自己的 scikit-learn 脚本。

```
from sagemaker.sklearn.processing import SKLearnProcessor
from sagemaker.processing import ProcessingInput, ProcessingOutput

sklearn_processor = SKLearnProcessor(framework_version='0.20.0',
                                     role=role,
                                     instance_type='ml.m5.xlarge',
                                     instance_count=1)

sklearn_processor.run(code='preprocessing.py',
                      inputs=[ProcessingInput(
                        source='s3://path/to/my/input-data.csv',
                        destination='/opt/ml/processing/input')],
                      outputs=[ProcessingOutput(source='/opt/ml/processing/output/train'),
                               ProcessingOutput(source='/opt/ml/processing/output/
validation'),
                               ProcessingOutput(source='/opt/ml/processing/output/test')]
                     )
```

要使用 Scikit-Learn 在 Amazon SageMaker 处理中并行处理数据，您可以通过设置来按 S3 键对输入对象分片。`s3_data_distribution_type='ShardedByS3Key'`内`ProcessingInput`以便每个实例接收大约相同数量的输入对象。

使用框架处理器处理数据

一个`FrameworkProcessor`可以使用指定的机器学习框架运行处理作业，为您选择的任何机器学习框架提供 Amazon SageMaker 托管容器。`FrameworkProcessor`为以下机器学习框架提供预制容器：拥抱面部、MXNet、PyTorch、TensorFlow 和 XGBoost。

这些区域有：`FrameworkProcessor`类还为您提供了容器配置的自定义功能。这些区域有：`FrameworkProcessor`类支持指定源目录`source_dir`用于处理脚本和依赖项。借助此功能，您可以授予处理器访问目录中的多个脚本的权限，而不是只指定一个脚本。`FrameworkProcessor`还支持包括`requirements.txt`中的文件`source_dir`用于自定义要安装在容器中的 Python 库。

有关`FrameworkProcessor`类及其方法和参数，请参见框架处理器中的Amazon SageMaker Python 开发工具包.

查看使用`FrameworkProcessor`有关每个受支持的机器学习框架，请参阅以下主题。

主题

- 拥抱 Face 框架处理器 (p. 731)
- MXNet 框架处理器 (p. 731)
- PyTorch 框架处理器 (p. 732)
- TensorFlow 框架处理器 (p. 733)
- XGBoost 框架处理器 (p. 734)

拥抱 Face 框架处理器

Huging Face 是自然语言处理 (NLP) 模型的开源提供商。这些区域有：`HuggingFaceProcessor`在 Amazon SageMaker Python SDK 中，您可以使用 Huging Face 脚本运行处理作业。当您使用`HuggingFaceProcessor`，您可以利用具有托管 Huging Face 环境的亚马逊构建的 Docker 容器，这样您就不需要自带容器。

以下代码示例显示了如何使用`HuggingFaceProcessor`使用 SageMaker 提供和维护的 Docker 映像来运行处理作业。请注意，运行作业时，可以在`source_dir`争论，你可以有`requirements.txt`文件位于你的`source_dir`指定处理脚本的依赖关系的目录。SageMaker 处理中安装依赖项`requirements.txt`在容器里给你。

```
from sagemaker.huggingface import HuggingFaceProcessor
from sagemaker.processing import ProcessingInput, ProcessingOutput
from sagemaker import get_execution_role

#Initialize the HuggingFaceProcessor
hfp = HuggingFaceProcessor(
    role=get_execution_role(),
    instance_count=1,
    instance_type='ml.g4dn.xlarge',
    transformers_version='4.4.2',
    pytorch_version='1.6.0',
    base_job_name='frameworkprocessor-hf'
)

#Run the processing job
hfp.run(
    code='processing-script.py',
    source_dir='scripts',
    inputs=[
        ProcessingInput(
            input_name='data',
            source=f's3://{BUCKET}/{S3_INPUT_PATH}',
            destination='/opt/ml/processing/input/data/'
        )
    ],
    outputs=[
        ProcessingOutput(output_name='train', source='/opt/ml/processing/output/train/',
 destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}'),
        ProcessingOutput(output_name='test', source='/opt/ml/processing/output/test/',
 destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}'),
        ProcessingOutput(output_name='val', source='/opt/ml/processing/output/val/',
 destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}')
    ]
)
```

如果您有`requirements.txt`文件，它应该是要在容器中安装的库的列表。的路径`source_dir`可以是相对路径、绝对路径或 Amazon S3 URI 路径。但是，如果您使用 Amazon S3 URI，则必须指向 tar.gz 文件。您可以在指定的目录中有多个脚本`source_dir`. 要了解有关`HuggingFaceProcessor`上课，请参阅拥抱人脸估计器中的Amazon SageMaker Python 开发工具包.

MXNet 框架处理器

Apache MxNet 是一个开源深度学习框架，通常用于训练和部署神经网络。这些区域有：`MXNetProcessor`在 Amazon SageMaker Python SDK 中，您可以使用 MxNet 脚本运行处理作业。当您使用`MXNetProcessor`，您可以利用亚马逊构建的 Docker 容器和托管的 MXNet 环境，这样您就不需要自带容器。

以下代码示例显示了如何使用`MXNetProcessor`使用 SageMaker 提供和维护的 Docker 映像来运行处理作业。请注意，运行作业时，可以在`source_dir`争论，你可以有`requirements.txt`文件位于你

的source_dir指定处理脚本的依赖关系的目录。SageMaker 处理中安装依赖项requirements.txt在容器里给你。

```
from sagemaker.mxnet import MXNetProcessor
from sagemaker.processing import ProcessingInput, ProcessingOutput
from sagemaker import get_execution_role

#Initialize the MXNetProcessor
mxp = MXNetProcessor(
    framework_version='1.8.0',
    py_version='py37',
    role=get_execution_role(),
    instance_count=1,
    instance_type='ml.c5.xlarge',
    base_job_name='frameworkprocessor-mxnet'
)

#Run the processing job
mxp.run(
    code='processing-script.py',
    source_dir='scripts',
    inputs=[
        ProcessingInput(
            input_name='data',
            source=f's3://{BUCKET}/{S3_INPUT_PATH}',
            destination='/opt/ml/processing/input/data/'
        )
    ],
    outputs=[
        ProcessingOutput(
            output_name='processed_data',
            source='/opt/ml/processing/output/',
            destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}'
        )
    ]
)
```

如果您有requirements.txt文件，它应该是要在容器中安装的库的列表。的路径source_dir可以是相对路径、绝对路径或 Amazon S3 URI 路径。但是，如果您使用 Amazon S3 URI，则必须指向 tar.gz 文件。您可以在指定的目录中有多个脚本source_dir. 要了解有关MXNetProcessor上课，请参阅MXNet 估算器中的Amazon SageMaker Python 开发工具包.

PyTorch 框架处理器

PyTorch 是一个开源机器学习框架。这些区域有：PyTorchProcessor在 Amazon SageMaker Python SDK 中，您可以使用 PyTorch 脚本运行处理作业。当您使用PyTorchProcessor，您可以在托管 PyTorch 环境中利用亚马逊构建的 Docker 容器，这样您就不需要自带容器。

以下代码示例显示了如何使用PyTorchProcessor使用 SageMaker 提供和维护的 Docker 映像来运行处理作业。请注意，运行作业时，可以在source_dir争论，你可以有requirements.txt文件位于你的source_dir指定处理脚本的依赖关系的目录。SageMaker 处理中安装依赖项requirements.txt在容器里给你。

```
from sagemaker.pytorch.processing import PyTorchProcessor
from sagemaker.processing import ProcessingInput, ProcessingOutput
from sagemaker import get_execution_role

#Initialize the PyTorchProcessor
pytorch_processor = PyTorchProcessor(
    framework_version='1.8',
    role=get_execution_role(),
```

```
    instance_type='ml.m5.xlarge',
    instance_count=1,
    base_job_name='frameworkprocessor-PT'
)

#Run the processing job
pytorch_processor.run(
    code='processing-script.py',
    source_dir='scripts',
    inputs=[
        ProcessingInput(
            input_name='data',
            source=f's3://{BUCKET}/{S3_INPUT_PATH}',
            destination='/opt/ml/processing/input'
        )
    ],
    outputs=[
        ProcessingOutput(output_name='data_structured', source='/opt/ml/processing/tmp/
data_structured', destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}'),
        ProcessingOutput(output_name='train', source='/opt/ml/processing/output/train',
 destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}'),
        ProcessingOutput(output_name='validation', source='/opt/ml/processing/output/val',
 destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}'),
        ProcessingOutput(output_name='test', source='/opt/ml/processing/output/test',
 destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}'),
        ProcessingOutput(output_name='logs', source='/opt/ml/processing/logs',
 destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}')
    ]
)
```

如果您有`requirements.txt`文件，它应该是要在容器中安装的库的列表。的路径`source_dir`可以是相对路径、绝对路径或 Amazon S3 URI 路径。但是，如果您使用 Amazon S3 URI，则必须指向 tar.gz 文件。您可以在指定的目录中有多个脚本`source_dir`. 要了解有关`PyTorchProcessor`上课，请参阅PyTorch 估算器中的Amazon SageMaker Python 开发工具包.

TensorFlow 框架处理器

TensorFlow 是一个开源机器学习和人工智能库。这些区域有：`TensorFlowProcessor`在 Amazon SageMaker Python SDK 中，您可以使用 TensorFlow 脚本运行处理作业。当您使用`TensorFlowProcessor`，您可以利用具有托管 TensorFlow 环境的亚马逊构建的 Docker 容器，这样您就不需要自带容器。

以下代码示例显示了如何使用`TensorFlowProcessor`使用 SageMaker 提供和维护的 Docker 映像来运行处理作业。请注意，运行作业时，可以在`source_dir`争论，你可以有`requirements.txt`文件位于你的`source_dir`指定处理脚本的依赖关系的目录。SageMaker 处理中安装依赖项`requirements.txt`在容器里给你。

```
from sagemaker.tensorflow import TensorFlowProcessor
from sagemaker.processing import ProcessingInput, ProcessingOutput
from sagemaker import get_execution_role

#Initialize the TensorFlowProcessor
tp = TensorFlowProcessor(
    framework_version='2.3',
    role=get_execution_role(),
    instance_type='ml.m5.xlarge',
    instance_count=1,
    base_job_name='frameworkprocessor-TF',
    py_version='py37'
)

#Run the processing job
```

```
tp.run(
    code='processing-script.py',
    source_dir='scripts',
    inputs=[
        ProcessingInput(
            input_name='data',
            source=f's3://{BUCKET}/{S3_INPUT_PATH}',
            destination='/opt/ml/processing/input/data'
        ),
        ProcessingInput(
            input_name='model',
            source=f's3://{BUCKET}/{S3_PATH_TO_MODEL}',
            destination='/opt/ml/processing/input/model'
        )
    ],
    outputs=[
        ProcessingOutput(
            output_name='predictions',
            source='/opt/ml/processing/output',
            destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}'
        )
    ]
)
```

如果您有`requirements.txt`文件，它应该是要在容器中安装的库的列表。的路径`source_dir`可以是相对路径、绝对路径或 Amazon S3 URI 路径。但是，如果您使用 Amazon S3 URI，则必须指向 tar.gz 文件。您可以在指定的目录中有多个脚本`source_dir`. 要了解有关`TensorFlowProcessor`上课，请参阅TensorFlow 评估程序中的Amazon SageMaker Python 开发工具包.

XGBoost 框架处理器

XGBoost 是一个开源机器学习框架。这些区域有：`XGBoostProcessor`在 Amazon SageMaker Python SDK 中，您可以使用 xgBoost 脚本运行处理作业。当您使用 xgBoostProcess 时，您可以利用亚马逊构建的 Docker 容器和托管 xgBoost 环境，这样您就不需要自带容器。

以下代码示例显示了如何使用`XGBoostProcessor`使用 SageMaker 提供和维护的 Docker 映像来运行处理作业。请注意，运行作业时，可以在`source_dir`争论，你可以有`requirements.txt`文件位于你的`source_dir`指定处理脚本的依赖关系的目录。SageMaker 处理中安装依赖项`requirements.txt`在容器里给你。

```
from sagemaker.xgboost import XGBoostProcessor
from sagemaker.processing import ProcessingInput, ProcessingOutput
from sagemaker import get_execution_role

#Initialize the XGBoostProcessor
xgb = XGBoostProcessor(
    framework_version='1.2-2',
    role=get_execution_role(),
    instance_type='ml.m5.xlarge',
    instance_count=1,
    base_job_name='frameworkprocessor-XGB',
)

#Run the processing job
xgb.run(
    code='processing-script.py',
    source_dir='scripts',
    inputs=[
        ProcessingInput(
            input_name='data',
            source=f's3://{BUCKET}/{S3_INPUT_PATH}',
            destination='/opt/ml/processing/input/data'
```

```
        )
    ],
    outputs=[
        ProcessingOutput(
            output_name='processed_data',
            source='/opt/ml/processing/output/',
            destination=f's3://{BUCKET}/{S3_OUTPUT_PATH}'
        )
    ]
)
```

如果您有`requirements.txt`文件，它应该是要在容器中安装的库的列表。的路径`source_dir`可以是相对路径、绝对路径或 Amazon S3 URI 路径。但是，如果您使用 Amazon S3 URI，则必须指向 tar.gz 文件。您可以在指定的目录中有多个脚本`source_dir`. 要了解有关`XGBoostProcessor`上课，请参阅XGBoost 估算器中的Amazon SageMaker Python 开发工具包.

使用您自己的处理代码

您可以安装库以在您自己的处理容器中运行脚本，或者在更高级的情况下，您可以自行构建满足要在 Amazon SageMaker 中运行的合同的处理容器。有关 SageMaker 中容器的更多信息，请参阅将 Docker 容器用于 SageMaker (p. 2400). 有关定义 Amazon SageMaker Processing 容器的合同的正式规范，请参阅构建您自己的处理容器（高级方案） (p. 736).

主题

- 使用您自己的处理容器运行脚本 (p. 735)
- 构建您自己的处理容器（高级方案） (p. 736)

使用您自己的处理容器运行脚本

您可以使用 scikit-learn 脚本来预处理数据并评估模型。要了解如何运行 scikit-learn 脚本来执行这些任务，请参阅scikit-learn 处理样本笔记本。这本笔记本使用`ScriptProcessor`类来自 Amazon SageMaker Python 开发工具包中的处理工具包。

以下示例显示使用`ScriptProcessor`类具有您自己的处理容器。该工作流程显示了如何创建自己的映像、构建容器以及如何使用`ScriptProcessor`类来使用容器运行 Python 预处理脚本。处理作业处理您的输入数据并将处理的数据保存在Amazon Simple Storage Service(Amazon S3 S3)。

在使用以下示例之前，您需要准备自己的输入数据和 Python 脚本来处理数据。有关此流程的端到端指导示例，请参阅scikit-learn 处理样本笔记本。

1. 创建 Docker 目录并添加用于创建处理容器的 Dockerfile。将 pandas 和 scikit-learn 安装到此目录中。（您也可以使用类似的 `RUN` 命令安装自己的依赖项。）

```
mkdir docker

%%writefile docker/Dockerfile

FROM python:3.7-slim-buster

RUN pip3 install pandas==0.25.3 scikit-learn==0.21.3
ENV PYTHONUNBUFFERED=TRUE

ENTRYPOINT ["python3"]
```

2. 使用 docker 命令构建容器，创建 Amazon Elastic Container Registry (Amazon ECR) 存储库，并将映像推送到 Amazon ECR。

```
import boto3

account_id = boto3.client('sts').get_caller_identity().get('Account')
region = boto3.Session().region_name
ecr_repository = 'sagemaker-processing-container'
tag = ':latest'
processing_repository_uri = '{}.dkr.ecr.{}.amazonaws.com/{}'.format(account_id, region,
 ecr_repository + tag)

# Create ECR repository and push docker image
!docker build -t $ecr_repository docker
!aws ecr get-login-password --region {region} | docker login --username Amazon --
password-stdin {account_id}.dkr.ecr.{region}.amazonaws.com
!aws ecr create-repository --repository-name $ecr_repository
!docker tag {ecr_repository + tag} $processing_repository_uri
!docker push $processing_repository_uri
```

3. 设置`ScriptProcessor`从 SageMaker Python 开发工具包中运行脚本。Replace*## URI*使用您创建的映像的 URI，然后替换*role_arn*用 ARN 获取Amazon Identity and Access Management有权访问您的目标 Amazon S3 存储桶的角色。

```
from sagemaker.processing import ScriptProcessor, ProcessingInput, ProcessingOutput

script_processor = ScriptProcessor(command=['python3'],
                image_uri='image_uri',
                role='role_arn',
                instance_count=1,
                instance_type='ml.m5.xlarge')
```

4. 运行脚本。Replace*preprocessing.py*使用您自己的 Python 处理脚本的名称，然后替换*s3: // path/to/my/input-data.csv*通过 Amazon S3 指向输入数据的路径。

```
script_processor.run(code='preprocessing.py',
                     inputs=[ProcessingInput(
                        source='s3://path/to/my/input-data.csv',
                        destination='/opt/ml/processing/input')],
                     outputs=[ProcessingOutput(source='/opt/ml/processing/output/train'),
                              ProcessingOutput(source='/opt/ml/processing/output/
validation'),
                              ProcessingOutput(source='/opt/ml/processing/output/
test')])
```

您可以将此相同过程与任何其他库或系统依赖项结合使用。您还可以使用现有 Docker 映像。这包括您在其他平台上运行的映像，例如Kubernetes.

构建您自己的处理容器（高级方案）

您可以为 Amazon SageMaker Processing 提供具有您自己的代码和依赖项的 Docker 映像，以运行数据处理、工程设计和模型评估工作负载。

下面的 Dockerfile 示例使用可作为处理作业运行的 Python 库 scikit-learn 和 pandas 来构建容器。

```
FROM python:3.7-slim-buster

# Install scikit-learn and pandas
RUN pip3 install pandas==0.25.3 scikit-learn==0.21.3

# Add a Python script and configure Docker to run it
ADD processing_script.py /
```

```
ENTRYPOINT ["python3", "/processing_script.py"]
```

构建此 Docker 映像并将其推送到 Amazon Elastic Container Registry (Amazon ECR) 存储库，并确保您的 SageMaker IAM 角色可从 Amazon ECR 中提取映像。然后，您可以在 Amazon SageMaker Processing 中运行此映像。

Amazon SageMaker Processing 如何运行处理容器映像

Amazon SageMaker Processing 采用类似于以下命令的方式运行处理容器映像。`AppSpecification.ImageUri`是您在中指定的 Amazon ECR 映像 URI。`CreateProcessingJob`operation.

```
docker run [AppSpecification.ImageUri]
```

此命令运行在 Docker 映像中配置的 `ENTRYPOINT` 命令。

您还可以通过在 `CreateProcessingJob` 请求中使用 `AppSpecification.ContainerEntrypoint` 和 `AppSpecification.ContainerArgument` 参数来覆盖映像中的入口点命令，或者向入口点命令提供命令行参数。指定这些参数会将 Amazon SageMaker Processing 配置为以类似于以下命令的方式运行容器。

```
docker run --entry-point [AppSpecification.ContainerEntrypoint]
[AppSpecification.ImageUri] [AppSpecification.ContainerArguments]
```

例如，如果您指定`ContainerEntrypoint`要成为`[python3, -v, /processing_script.py]`在您的`CreateProcessingJob` 和请求`ContainerArguments`要成为`[data-format, csv]`，Amazon SageMaker 处理使用以下命令运行您的容器。

```
python3 -v /processing_script.py data-format csv
```

构建处理容器时，请考虑以下详细信息：

- Amazon SageMaker Processing 决定作业是完成还是失败取决于命令运行的退出代码。如果所有处理容器都成功退出且退出代码为 0，处理作业已完成；如果任何容器以非零退出代码退出，则处理作业失败。
- Amazon SageMaker Processing 允许您像使用 Docker API 一样覆盖处理容器的入口点并设置命令行参数。Docker 映像还可以使用 `ENTRYPOINT` 和 CMD 指令配置入口点和命令行参数。`CreateProcessingJob` 的 `ContainerEntrypoint` 和 `ContainerArgument` 参数配置 Docker 映像入口点和参数的方式反映了 Docker 如何通过 Docker API 覆盖入口点和参数的方式：
  - 如果都不是`ContainerEntrypoint`也不`ContainerArguments`已提供，“处理” 使用默认值`ENTRYPOINT`或者图片中的 CMD。
  - 如果`ContainerEntrypoint`已提供但未提供`ContainerArguments`，Processing 将使用给定的入口点运行映像，并忽略`ENTRYPOINT`和图片中的 CMD。
  - 如果`ContainerArguments`已提供但未提供`ContainerEntrypoint`，“处理” 使用默认值运行图像`ENTRYPOINT`在图片中以及提供的参数。
  - 如果两者都`ContainerEntrypoint`和`ContainerArguments`Processing 将使用给定的入口点和参数运行映像，并忽略`ENTRYPOINT`和图片中的 CMD。
- 在您的 Dockerfile 中使用 `ENTRYPOINT` 指令的 exec 形式 (`ENTRYPOINT ["executable", "param1", "param2"]`)，而不是 shell 形式 (`ENTRYPOINT command param1 param2`)。这可以让你的处理容器接收`SIGINT`和`SIGKILL`Processing 使用信号来停止处理作业`StopProcessingJob`API。
- `/opt/ml`SageMaker 将保留其所有子目录。在构建 Processing Docker 映像时，请不要将处理容器所需的任何数据放置在这些目录中。
- 如果您计划使用 GPU 设备，请确保容器与 nvidia-docker 兼容。在容器中仅包含 CUDA 工具包。不要将 NVIDIA 驱动程序与镜像捆绑。有关 nvidia-docker 的更多信息，请参阅 NVIDIA/nvidia-docker。

Amazon SageMaker Processing 如何为处理容器配置输入和输出

使用 `CreateProcessingJob` 操作创建处理作业时，可以指定多个 `ProcessingInput` 和 `ProcessingOutput` 值。

您将`ProcessingInput`参数指定要从中下载数据的 Amazon Simple Storage Service (Amazon S3) URI，以及要将数据下载到的处理容器路径。这些区域有：`ProcessingOutput`参数配置从中上传数据的处理容器路径，以及在 Amazon S3 中将该数据上传到的位置。对于 `ProcessingInput` 和 `ProcessingOutput`，处理容器中的路径必须以 `/opt/ml/processing/` 开头。

例如，您可以创建一个处理作业，该处理作业具有一个 `ProcessingInput` 参数（此参数可将数据从 `s3://your-data-bucket/path/to/input/csv/data` 下载到处理容器中的 `/opt/ml/processing/csv`）；还具有一个 `ProcessingOutput` 参数（此参数可将数据从 `/opt/ml/processing/processed_csv` 上传到 `s3://your-data-bucket/path/to/output/csv/data`）。您的处理作业将读取输入数据，并将输出数据写入 `/opt/ml/processing/processed_csv`。然后，它将写入此路径的数据上传到指定的 Amazon S3 输出位置。

> **Important**
>
> 符号链接（符号链接）不能用于将输出数据上传到 Amazon S3。上传输出数据时不遵循符号链接。

Amazon SageMaker Processing 如何为处理容器提供日志和指标

当你的处理容器写入`stdout`要么`stderr`，Amazon SageMaker Processing 会保存每个处理容器的输出并将输出放入 Amazon CloudWatch 日志中。有关日志记录的信息，请参阅使用 Amazon CloudWatch 记录 Amazon SageMaker 事件 (p. 2820)。

Amazon SageMaker Processing 还会为运行处理容器的每个实例提供 CloudWatch 指标。有关指标的信息，请参阅使用 Amazon CloudWatch 监控 Amazon SageMaker (p. 2810)。

Amazon SageMaker Processing 如何配置处理容器

Amazon SageMaker Processing 通过环境变量和两个 JSON 文件为处理容器提供配置信息。`/opt/ml/config/processingjobconfig.json`和`/opt/ml/config/resourceconfig.json`— 在容器中的预定义位置。

当处理作业启动时，它将使用您在 `CreateProcessingJob` 请求中通过 `Environment` 映射指定的环境变量。`/opt/ml/config/processingjobconfig.json` 文件包含有关处理容器的主机名的信息，并且也在 `CreateProcessingJob` 请求中指定。

以下示例显示了 `/opt/ml/config/processingjobconfig.json` 文件的格式。

```
{
    "ProcessingJobArn": "<processing_job_arn>",
    "ProcessingJobName": "<processing_job_name>",
    "AppSpecification": {
        "ImageUri": "<image_uri>",
        "ContainerEntrypoint": null,
        "ContainerArguments": null
    },
    "Environment": {
        "KEY": "VALUE"
    },
    "ProcessingInputs": [
        {
            "InputName": "input-1",
            "S3Input": {
                "LocalPath": "/opt/ml/processing/input/dataset",
                "S3Uri": "<s3_uri>",
```

```
                "S3DataDistributionType": "FullyReplicated",
                "S3DataType": "S3Prefix",
                "S3InputMode": "File",
                "S3CompressionType": "None",
                "S3DownloadMode": "StartOfJob"
            }
        }
    ],
    "ProcessingOutputConfig": {
        "Outputs": [
            {
                "OutputName": "output-1",
                "S3Output": {
                    "LocalPath": "/opt/ml/processing/output/dataset",
                    "S3Uri": "<s3_uri>",
                    "S3UploadMode": "EndOfJob"
                }
            }
        ],
        "KmsKeyId": null
    },
    "ProcessingResources": {
        "ClusterConfig": {
            "InstanceCount": 1,
            "InstanceType": "ml.m5.xlarge",
            "VolumeSizeInGB": 30,
            "VolumeKmsKeyId": null
        }
    },
    "RoleArn": "<IAM role>",
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 86400
    }
}
```

/opt/ml/config/resourceconfig.json 文件包含有关处理容器的主机名的信息。使用以下命令：HostName在创建或运行分布式处理代码时。

```
{
  "current_host": "algo-1",
  "hosts": ["algo-1","algo-2","algo-3"]
}
```

请勿使用 /etc/hostname 或 /etc/hosts 中包含的有关主机名的信息，因为它可能不准确。

处理容器可能无法立即获得主机名信息。我们建议当节点在集群中可用时，在主机名解析操作上添加重试策略。

保存和访问有关处理作业的元数据信息

要在退出处理容器后保存其中的元数据，容器可以将 UTF-8 编码的文本写入 /opt/ml/output/message 文件。处理作业进入任何终端状态后（"Completed"，"Stopped"、或"Failed"），"ExitMessage"字段中的字段DescribeProcessingJob包含此文件的前 1 KB。通过调用 DescribeProcessingJob 访问文件的初始部分，这会通过 ExitMessage 参数返回它。对于失败的处理作业，您可以使用此字段传达有关处理容器失败原因的信息。

> **Important**
>
> 请勿将敏感数据写入 /opt/ml/output/message 文件。

如果此文件中的数据未经 UTF-8 编码，则作业将失败并返回 ClientError。如果多个容器退出且显示 ExitMessage，，则会将每个处理容器中的 ExitMessage 的内容连接起来，然后截断为 1 KB。

使用 SageMaker Python 开发工具包运行处理容器

您可以使用 SageMaker Python 开发工具包运行您自己的处理映像，`Processor`类。以下示例演示如何运行您自己的处理容器，该容器具有一个来自 Amazon Simple Storage Service (Amazon S3) 的输入和一个到 Amazon S3 的输出。

```
from sagemaker.processing import Processor, ProcessingInput, ProcessingOutput

processor = Processor(image_uri='<your_ecr_image_uri>',
                      role=role,
                      instance_count=1,
                      instance_type="ml.m5.xlarge")

processor.run(inputs=[ProcessingInput(
                         source='<s3_uri or local path>',
                         destination='/opt/ml/processing/input_data')],
                      outputs=[ProcessingOutput(
                         source='/opt/ml/processing/processed_data',
                         destination='<s3_uri>')],
                      ))
```

您可以向 `ScriptProcessor` 提供您的映像、要运行的命令以及要在该容器内运行的代码，而不是将处理代码构建到处理映像中。有关示例，请参阅 使用您自己的处理容器运行脚本 (p. 735)。

您还可以使用 Amazon SageMaker Processing 通过提供的 scikit-learn 映像。`SKLearnProcessor`来运行 scikit-learn 脚本。有关示例，请参阅使用 scikit-learn 处理数据 (p. 729)。

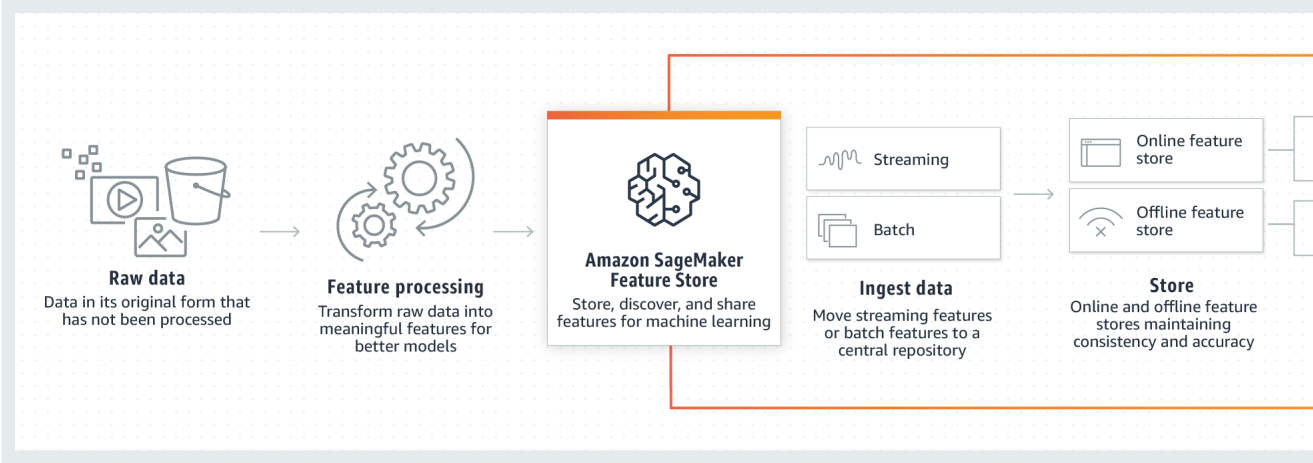
创建、存储和与亚马逊共享功能 SageMaker 功能商店

机器学习 (ML) 开发过程通常从提取数据信号开始，也称为功能从数据到训练机器学习模型。亚马逊 SageMaker 功能存储使数据科学家、机器学习工程师和全科医生能够轻松创建、共享和管理机器学习 (ML) 开发的功能。功能商店通过减少将原始数据转换为训练 ML 算法的功能所需的重复性数据处理和策划工作来加速此过程。

此外，只编写一次数据的处理逻辑，生成的要素用于训练和推理，从而减少了训练服务偏差。要素存储是要素和相关元数据的集中式存储，因此可以轻松发现和重复使用要素。您可以创建在线或离线商店。在线商店用于低延迟的实时推理用例，离线商店用于训练和批量推断。

下图演示了如何使用亚马逊。 SageMaker 作为机器学习管道的一部分的功能商店。首先，你读取原始数据并处理它。您可以通过流式传输到在线和离线商店，或者直接将数据批量提取到离线商店。你先创建`FeatureGroup`然后将其配置为在线或离线商店，或者两者兼而有。然后，你可以将数据提取到`FeatureGroup`然后将其存放在你的商店里。 一个`FeatureGroup`是一组功能，通过要素存储中的架构定义来描述记录。

在线商店主要用于支持需要低毫秒延迟读取和高吞吐量写入的实时预测。线下商店主要用于批量预测和模型训练。离线商店是仅限追加的商店，可用于存储和访问历史要素数据。离线商店可以帮助您存储和提供用于探索和模型训练的功能。在线商店仅保留最新的功能数据。要素组定义在创建之后是不可变的。



如何运作功能商店

在功能商店中，要素存储在名为功能组。 您可以将要素组可视化为表格，其中每列都是要素，每行都有唯一标识符。原则上，功能组由特定于每个要素的功能和值组成。一个`Record`是与唯一对应的要素的值的集合`RecordIdentifier`. 总而言之，`FeatureGroup`是在你的`FeatureStore`描述`Record`.

您可以在以下模式下使用功能商店：

- 在线— 在联机模式下，以低延迟（毫秒）读取功能进行读取，并用于高吞吐量预测。此模式要求将功能组存储在在线商店中。
- 离线— 在离线模式下，将大量数据流传送到离线商店，该商店可用于训练和批量推理。此模式要求将功能组存储在离线商店中。离线商店使用 S3 存储桶进行存储，还可以使用 Athena 查询获取数据。

- 线上和离线— 这包括在线和离线模式。

您可以通过两种方式将数据提取到功能商店中的要素组：流式传输或批量。当您通过流媒体提取数据时，通过调用同步功能商店将记录集合推送到功能商店`PutRecord`API 调用。此 API 使您能够在功能商店中维护最新的功能值，并在检测到更新后立即推送新功能值。

或者，功能商店可以批量处理和提取数据。您可以使用亚马逊创作功能 SageMaker Data Wrangler，在功能商店中创建要素组，然后使用 SageMaker 使用从 Data Wrangler 导出的笔记本处理作业。此模式允许批量提取到离线商店。如果功能组配置为在线和离线使用，它还支持对在线商店进行摄入。

创建功能组

要将要素摄录到要素存储中，必须首先为属于该要素组的所有要素定义要素组和要素定义（要素名称和数据类型）。创建之后，要素组是不可变的。要素组名称在Amazon地区和Amazonaccount. 创建要素组时，还可以为要素组创建元数据，例如简短描述、存储配置、识别每条记录的功能和事件时间，以及用于存储作者、数据源、版本等信息的标签。

Important

`FeatureGroup`姓名或相关元数据（如描述或标签）不应包含任何个人身份信息 (PII) 或机密信息。

查找、发现和共享功能

在 feature store 中创建功能组后，该功能商店的其他授权用户可以共享和发现该功能组。用户可以浏览功能商店中所有功能组的列表，或通过按要素组名称、描述、记录标识符名称、创建日期和标签进行搜索来发现现有功能组。

在线商店中存储的功能的实时推断

借助功能商店，您可以使用来自流媒体源的数据（来自另一个应用程序的干净流数据）实时丰富在线商店中存储的功能，并以低毫秒的延迟提供功能以实时推断。

您也可以跨不同的方式执行联接`FeatureGroups`通过查询两个不同的方式进行实时推理`FeatureGroups`在客户端应用程序中。

模型训练和 Batch 推断的离线商店

功能商店为 S3 存储桶中的功能值提供离线存储空间。使用基于事件时间的前缀方案将您的数据存储在 S3 存储桶中。 离线商店是一个仅追加商店，使功能商店能够维护所有功能值的历史记录。数据以 Pincia 格式存储在离线商店中，以便优化存储和查询访问。

您可以使用亚马逊的 Data Wrangler 查询、探索和可视化功能 SageMaker 工作室。 功能商店支持组合数据以生成、训练、验证和测试数据集，并允许您在不同的时间点提取数据。

功能数据摄取

可以创建要素生成管道来处理大批量（100 万行或更多数据）或小批处理，并将要素数据写入离线或在线商店。流媒体源（例如适用于 Apache Kafka 的亚马逊托管流媒体或 Amazon Kinesis）也可用作数据源，从中提取功能并直接馈送到在线商店以进行培训、推理或功能创建。

你可以通过调用同步将记录推送到功能商店PutRecordAPI 调用。由于这是同步 API 调用，因此它允许在单个 API 调用中推送小批量更新。这使您能够保持功能值的高新鲜度，并在检测到更新后立即发布值。这些也称为流媒体功能.

当要素数据被摄取和更新时，要素商店将在离线商店中存储所有要素的历史数据。对于批量提取，您可以从 S3 存储桶中提取功能值或使用 Athena 进行查询。您还可以使用 Data Wrangler 处理和设计新功能，然后可以导出到选定的 S3 存储桶以供功能商店访问。对于批量摄取，您可以配置处理作业以批量将数据提取到功能存储中，也可以使用 Athena 从 S3 存储桶中提取功能值。

移除Record从你的在线商店中，使用DeleteRecordAPI 调用。这也会将删除的记录添加到离线商店。

开始使用 Amazon SageMaker Feature Store

要开始使用 Amazon SageMaker feature store，请查看基本概念，了解如何为功能商店提取数据，然后演练功能商店示例。以下各节介绍了如何创建要素组、将数据提取到群组以及如何管理 feature store 的安全性。

主题

- Feature Store 概念 (p. 743)
- 创建要素组 (p. 744)
- 向 IAM 角色添加所需的策略 (p. 754)
- 在 Amazon SageMaker Studio 中使用 Amazon SageMaker 功能商 (p. 759)

Feature Store 概念

以下术语列表是了解 Amazon SageMaker 功能商店功能的关键：

- Feature Store— 作为存储、检索、删除、跟踪、共享、发现和控制对功能的访问的唯一真相来源。
- 功能— 包含观察到的现象的可测量属性或特征。在 Amazon SageMaker 功能商店 API 中，功能是记录的属性。您可以为要素商店中存储的每个要素定义名称和类型。 名称唯一标识要素组中的要素。类型标识要素值的数据类型。支持的数据类型有：字符串、积分和分数。
- 功能组— AFeatureGroup是主要的要素商店资源，其中包含存储在 Amazon SageMaker 功能商店中的所有数据的元数据。 要素组是要素存储中定义的用于描述记录的要素的逻辑分组。要素组的定义由功能定义列表、记录标识符名称以及其在线和离线商店的配置组成。
- 功能定义— AFeatureDefinition由名称和以下数据类型之一组成：整数、字符串或小数。 一个FeatureGroup包含功能定义列表。
- 记录标识符名称— 每个要素组都使用记录标识符名称定义。记录标识符名称必须引用要素组的要素定义中定义的要素的名称之一。
- RECORDS— ARecord是单个记录标识符值的要素值的集合。记录标识符名称和时间戳的组合可唯一标识要素组中的记录。
- 事件时间— 发生新事件的时间点，该事件对应于要素组中的记录的创建或更新。要素组中的所有记录都必须具有对应的Eventtime. 它可以用于跟踪记录随时间推移的更改。在线商店包含与最后一个对应的记录Eventtime对于记录标识符名称，而离线商店包含所有历史记录。事件时间值必须为格式的 ISO-8601 字符串。支持以下格式：yyyy-MM-dd'T'HH:mm:ssZ和yyyy-MM-dd'T'HH:mm:ss.SSSZ哪里yyyy、MM, 和dd分别表示年、月和日，HH、mm、ss，如果适用，SSS分别表示小时、月、秒和毫秒。T和Z是常量。
- 在线商店— 支持实时查找记录的功能组的低延迟、高可用性缓存。在线商店允许快速访问最新价值Record通过GetRecordAPI。要素组包含OnlineStoreConfig控制数据存储到的位置。
- 离线商店—OfflineStore中，将历史数据存储到 S3 存储桶中。当不需要低（亚秒）延迟读取时使用它。例如，当您想存储和提供用于探索、模型训练和批量推断的要素时。 要素组包含OfflineStoreConfig控制数据存储到的位置。

- 摄取— 在 feature store 中填充要素组的行为。

创建要素组

一个FeatureGroup是主要的要素商店资源，其中包含存储在亚马逊中的所有数据的元数据 SageMaker 功能商店。 要素组是要素存储中定义的用于描述记录的要素的逻辑分组。功能组的定义由的列表、记录标识符名称以及其在线和离线商店的配置组成。 本主题中的示例代码使用 SageMaker Python 开发工具包。底层 API 可供使用其他语言的开发人员使用。

在使用 feature store 之前，您通常会加载数据集、运行转换并设置要素以进行摄取。这个过程有很大差异，高度依赖于你的数据。以下主题中的示例代码参考简介、亚马逊进行欺诈侦测 SageMaker 功能还原例如笔记本。我们建议您在亚马逊运行此笔记本电脑。 SageMaker Studio 因为本指南中的代码是概念性的，如果复制的话不能完全正常运行。

功能存储支持以下数据类型：String、Fractional（IEEE 64 位浮点值），以及Integral（Int64-64 位有符号积分值）。默认类型设置为String. 这意味着，如果数据集中的列不是float要么long类型，默认为String在你的 feature store 里。

您可以使用架构来描述数据的列和数据类型。你把这个模式传递到FeatureDefinitions，的必需参数FeatureGroup. 您可以使用 SageMaker Python SDK，当您使用load_feature_definitionsfunction.

使用现有记录 ID 添加新要素记录时的默认行为如下。在离线商店中，将附加新记录。在在线商店中，如果新记录的事件时间小于现有事件时间不会发生，但是，如果新记录的事件时间大于或等于现有事件时间，则记录将被写入过多。

功能商店简介

本主题中的示例代码引用Amazon SageMaker Feature Store 简介笔记本示例。建议您在 Amazon SageMaker Studio 中运行此笔记本，因为本指南中的代码是概念性的，如果复制，则无法完全正常工作。

第 1 步：设置

要开始使用要素商店，请创建 SageMaker、boto3 和要素商店会话。然后，您要用于功能的 S3 存储桶。这是你的线下商店。以下代码使用 SageMaker 默认存储桶并向其添加自定义前缀。

> Note
>
> 您使用的角色必须附加以下托管策略：AmazonS3FullAccess和AmazonSageMakerFeatureStoreAccess.

```
# SageMaker Python SDK version 2.x is required
import sagemaker
import sys
```

```
import boto3
import pandas as pd
import numpy as np
import io
from sagemaker.session import Session
from sagemaker import get_execution_role

prefix = 'sagemaker-featurestore-introduction'
role = get_execution_role()
```

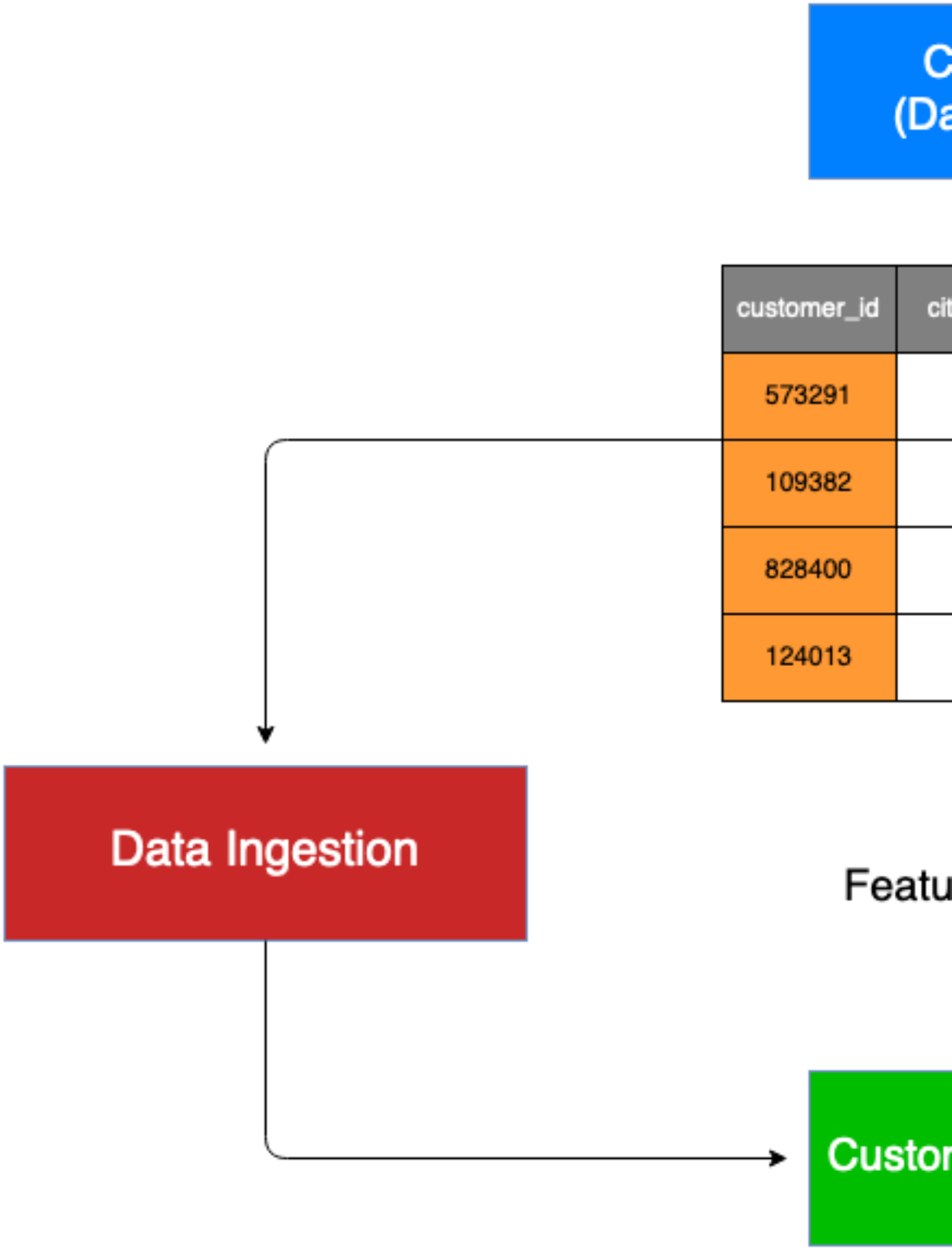
```
sagemaker_session = sagemaker.Session()
region = sagemaker_session.boto_region_name
s3_bucket_name = sagemaker_session.default_bucket()
```

第 2 步：检查您的数据

在本笔记本示例中，我们从GitHub 仓库承载完整的笔记本电脑。

```
customer_data = pd.read_csv("data/feature_store_introduction_customer.csv")
orders_data = pd.read_csv("data/feature_store_introduction_orders.csv")

print(customer_data.head())
print(orders_data.head())
```

下图说明了数据在将数据提取到要素商店之前所经历的步骤。在本笔记本中，我们展示了一个用例，即您拥有来自多个来源的数据并希望将它们独立存储在 feature store 中。我们的示例考虑来自数据仓库的数据（客户数据）和来自实时流媒体服务的数据（订单数据）。


customer_id
573291
109382
828400
124013
Data Ingestion

第 3 步：创建功能组

首先，我们首先为 customer_data 和 orders_data 创建要素组名称。之后，我们创建了两个功能组，一个用于customer_data另一个用于orders_data.

```
import time
from time import strftime, gmtime
customers_feature_group_name = 'customers-feature-group-' + strftime('%d-%H-%M-%S',
 gmtime())
orders_feature_group_name = 'orders-feature-group-' + strftime('%d-%H-%M-%S', gmtime())
```

实例化FeatureGroup对象customers_data和orders_data.

```
from sagemaker.feature_store.feature_group import FeatureGroup

customers_feature_group = FeatureGroup(
    name=customers_feature_group_name, sagemaker_session=sagemaker_session
)
orders_feature_group = FeatureGroup(
    name=orders_feature_group_name, sagemaker_session=sagemaker_session
)
```

```
import time
current_time_sec = int(round(time.time()))
record_identifier_feature_name = "customer_id"
```

附加EventTime要素到你的数据框中。此参数是必需的，并且每个数据点的时间戳。

```
customer_data["EventTime"] = pd.Series([current_time_sec]*len(customer_data),
 dtype="float64")
orders_data["EventTime"] = pd.Series([current_time_sec]*len(orders_data), dtype="float64")
```

将要素定义加载到功能组。

```
customers_feature_group.load_feature_definitions(data_frame=customer_data)
orders_feature_group.load_feature_definitions(data_frame=orders_data)
```

下面我们称之为 create 来创建两个功能组，customers_feature_group和orders_feature_group。

```
customers_feature_group.create(
    s3_uri=f"s3://{s3_bucket_name}/{prefix}",
    record_identifier_name=record_identifier_feature_name,
    event_time_feature_name="EventTime",
    role_arn=role,
    enable_online_store=True
)

orders_feature_group.create(
    s3_uri=f"s3://{s3_bucket_name}/{prefix}",
    record_identifier_name=record_identifier_feature_name,
    event_time_feature_name="EventTime",
    role_arn=role,
    enable_online_store=True
)
```

为了确认您的 FeatureGroup 已创建，我们使用DescribeFeatureGroup和ListFeatureGroups用于显示创建的功能组的 API。

```
customers_feature_group.describe()
```

```
orders_feature_group.describe()
```

```
sagemaker_session.boto_session.client('sagemaker',
 region_name=region).list_feature_groups() # We use the boto client to list FeatureGroups
```

第 4 步：将数据提取到要素组

在 FeatuRegGroups 创建之后，我们可以将数据放入 FeatuRegGroups 中。如果您使用的是 SageMaker Python SDK，请使用`ingest`API 调用。如果你通过使用 boto3 使用，那么请使用`PutRecord`API。摄取这两个 FageGroups 都需要不到 1 分钟的时间。本示例使用 SageMaker Python SDK，因此它使用`ingest`API 调用。

```
def check_feature_group_status(feature_group):
    status = feature_group.describe().get("FeatureGroupStatus")
    while status == "Creating":
        print("Waiting for Feature Group to be Created")
        time.sleep(5)
        status = feature_group.describe().get("FeatureGroupStatus")
    print(f"FeatureGroup {feature_group.name} successfully created.")

check_feature_group_status(customers_feature_group)
check_feature_group_status(orders_feature_group)
```

```
customers_feature_group.ingest(
    data_frame=customer_data, max_workers=3, wait=True
)
```

```
orders_feature_group.ingest(
    data_frame=orders_data, max_workers=3, wait=True
)
```

使用任意客户记录 ID，我们使用 573291`get_record`以检查数据是否已被摄入到功能组。

```
customer_id = 573291
sample_record = sagemaker_session.boto_session.client('sagemaker-featurestore-runtime',
 region_name=region).get_record(FeatureGroupName=customers_feature_group_name,
 RecordIdentifierValueAsString=str(customer_id))
```

```
print(sample_record)
```

下面演示如何使用`batch_get_record`以获取一批记录。

```
all_records = sagemaker_session.boto_session.client(
    "sagemaker-featurestore-runtime", region_name=region
).batch_get_record(
    Identifiers=[
        {
            "FeatureGroupName": customers_feature_group_name,
            "RecordIdentifiersValueAsString": ["573291", "109382", "828400", "124013"],
        },
        {
            "FeatureGroupName": orders_feature_group_name,
            "RecordIdentifiersValueAsString": ["573291", "109382", "828400", "124013"],
```

```
        },
    ]
)
```

```
print(all_records)
```

第 5 步：清除

在这里我们删除我们创建的功能组。

```
customers_feature_group.delete()
orders_feature_group.delete()
```

第 6 步：后续步骤

在本示例笔记本中，您学习了如何快速开始使用功能商店、创建要素组以及将数据提取到它们中。

有关如何将功能商店用于欺诈检测用例的高级示例，请参阅使用功能商店检测欺诈.

步骤 7：程序员注意

在这本笔记本中，我们使用了各种不同的 API 调用。其中大多数都可以通过 Python SDK 访问，但是有些只存在于 boto3 中。您可以直接在功能存储对象上调用 Python SDK API 调用，而要调用 boto3 中存在的 API 调用，必须首先通过 boto 和 sagemaker 会话访问 boto 客户端：例如`sagemaker_session.boto_session.client()`.

下面我们列出了本笔记本中使用的 Python SDK 中存在的 API 调用以及 boto3 中存在的 API 调用以供参考。

Python 开发工具包 API 调用

```
describe()
ingest()
delete()
create()
load_feature_definitions()
```

Boto3 API 调用

```
list_feature_groups()
get_record()
```

使用功能商店检测欺诈

第 1 步：设置功能商店

要开始使用要素商店，请创建 SageMaker 会话、boto3 会话和要素商店会话。此外，您要用于功能的 S3 存储桶。这是你的线下商店。以下代码使用 SageMaker 默认存储桶并向其添加自定义前缀。

> **Note**
>
> 您使用的角色必须附加以下托管策略：`AmazonSageMakerFullAccess`和`AmazonSageMakerFeatureStoreAccess`.

```
import boto3
import sagemaker
from sagemaker.session import Session

sagemaker_session = sagemaker.Session()
```

```
region = sagemaker_session.boto_region_name
boto_session = boto3.Session(region_name=region)
role = sagemaker.get_execution_role()
default_bucket = sagemaker_session.default_bucket()
prefix = 'sagemaker-featurestore'
offline_feature_store_bucket = 's3://{}/{}'.format(default_bucket, prefix)

sagemaker_client = boto_session.client(service_name='sagemaker', region_name=region)
featurestore_runtime = boto_session.client(service_name='sagemaker-featurestore-runtime',
 region_name=region)

feature_store_session = Session(
    boto_session=boto_session,
    sagemaker_client=sagemaker_client,
    sagemaker_featurestore_runtime_client=featurestore_runtime
)
```

第 2 步：将数据集和分区数据加载到要素组

将数据加载到每个要素的数据框中。设置要素组后，可以使用这些数据框。在欺诈检测示例中，您可以在以下代码中看到以下步骤。

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import io

fraud_detection_bucket_name = 'sagemaker-featurestore-fraud-detection'
identity_file_key = 'sampled_identity.csv'
transaction_file_key = 'sampled_transactions.csv'

identity_data_object = s3_client.get_object(Bucket=fraud_detection_bucket_name,
 Key=identity_file_key)
transaction_data_object = s3_client.get_object(Bucket=fraud_detection_bucket_name,
 Key=transaction_file_key)

identity_data = pd.read_csv(io.BytesIO(identity_data_object['Body'].read()))
transaction_data = pd.read_csv(io.BytesIO(transaction_data_object['Body'].read()))

identity_data = identity_data.round(5)
transaction_data = transaction_data.round(5)

identity_data = identity_data.fillna(0)
transaction_data = transaction_data.fillna(0)

# Feature transformations for this dataset are applied before ingestion into FeatureStore.
# One hot encode card4, card6
encoded_card_bank = pd.get_dummies(transaction_data['card4'], prefix = 'card_bank')
encoded_card_type = pd.get_dummies(transaction_data['card6'], prefix = 'card_type')

transformed_transaction_data = pd.concat([transaction_data, encoded_card_type,
 encoded_card_bank], axis=1)
transformed_transaction_data =
 transformed_transaction_data.rename(columns={"card_bank_american express":
 "card_bank_american_express"})
```

第 3 步：设置功能组

设置要素组时，需要使用唯一的名称自定义功能名称，然后使用`FeatureGroup`类。

```
from sagemaker.feature_store.feature_group import FeatureGroup
feature_group_name = "some string for a name"
```

```
feature_group = FeatureGroup(name=feature_group_name,
 sagemaker_session=feature_store_session)
```

例如，在欺诈检测示例中，两个功能组是identity和transaction. 在下面的代码中，您可以看到如何使用时间戳自定义名称，然后通过传入名称和会话来设置每个组。

```
import time
from time import gmtime, strftime, sleep
from sagemaker.feature_store.feature_group import FeatureGroup

identity_feature_group_name = 'identity-feature-group-' + strftime('%d-%H-%M-%S', gmtime())
transaction_feature_group_name = 'transaction-feature-group-' + strftime('%d-%H-%M-%S',
 gmtime())

identity_feature_group = FeatureGroup(name=identity_feature_group_name,
 sagemaker_session=feature_store_session)
transaction_feature_group = FeatureGroup(name=transaction_feature_group_name,
 sagemaker_session=feature_store_session)
```

第 4 步：设置记录标识符和事件时间功能

在此步骤中，您可以指定记录标识符名称和事件时间要素名称。此名称映射到数据中相应要素的列。例如，在欺诈检测示例中，感兴趣的一栏是TransactionID.EventTime当没有时间戳可用时，可以将其附加到数据中。在以下代码中，您将看到如何设置这些变量，然后EventTime将附加到两个要素的数据中。

```
record_identifier_name = "TransactionID"
event_time_feature_name = "EventTime"
current_time_sec = int(round(time.time()))
identity_data[event_time_feature_name] = pd.Series([current_time_sec]*len(identity_data),
 dtype="float64")
transformed_transaction_data[event_time_feature_name] =
 pd.Series([current_time_sec]*len(transaction_data), dtype="float64")
```

第 5 步：加载功能定义

现在，您可以通过传递包含要素数据的数据框来加载要素定义。在欺诈检测示例的以下代码中，身份功能和交易功能都是通过使用load_feature_definitions，此函数会自动检测每列数据的数据类型。对于使用模式而不是自动检测的开发人员，请参阅从 Data Wrangler 中导出要素组代码示例，该代码显示如何加载架构、映射它并将其添加为FeatureDefinition您可以使用创建FeatureGroup. 此示例还介绍了 boto3 实现，您可以使用它代替 SageMaker Python SDK。

```
identity_feature_group.load_feature_definitions(data_frame=identity_data); # output is
 suppressed
transaction_feature_group.load_feature_definitions(data_frame=transformed_transaction_data);
 # output is suppressed
```

第 6 步：创建功能组

在此步骤中，您将使用create函数来创建要素组。以下代码显示所有可用参数。默认情况下不会创建在线商店，因此您必须将其设置为True如果要启用它。这些区域有：s3_uri是离线商店的 S3 存储桶位置。

```
# create a FeatureGroup
feature_group.create(
    description = "Some info about the feature group",
    feature_group_name = feature_group_name,
    record_identifier_name = record_identifier_name,
    event_time_feature_name = event_time_feature_name,
    feature_definitions = feature_definitions,
    role_arn = role,
```

```
    s3_uri = offline_feature_store_bucket,
    enable_online_store = True,
    online_store_kms_key_id = None,
    offline_store_kms_key_id = None,
    disable_glue_table_creation = False,
    data_catalog_config = None,
    tags = ["tag1","tag2"])
```

欺诈检测示例中的以下代码显示了最小的`create`调用正在创建的两个要素组中的每个要素组。

```
identity_feature_group.create(
    s3_uri=offline_feature_store_bucket,
    record_identifier_name=record_identifier_name,
    event_time_feature_name=event_time_feature_name,
    role_arn=role,
    enable_online_store=True
)

transaction_feature_group.create(
    s3_uri=offline_feature_store_bucket,
    record_identifier_name=record_identifier_name,
    event_time_feature_name=event_time_feature_name,
    role_arn=role,
    enable_online_store=True
)
```

创建要素组时，加载数据需要一段时间，您需要等到要素组创建完毕后才能使用它。您可以使用以下方法检查状态。

```
status = feature_group.describe().get("FeatureGroupStatus")
```

在创建要素组时，您会收到`Creating`作为响应。当此步骤成功完成后，响应为`Created`. 其他可能状态包括`CreateFailed`、`Deleting`，或者`DeleteFailed`.

步骤 7：使用功能组

现在您已经设置了功能组，可以执行以下任意任何任务：

主题

- 描述功能组 (p. 752)
- 列出功能组 (p. 752)
- 将记录放入要素组 (p. 753)
- 从功能组获取记录 (p. 753)
- 生成 Hive DDL 命令 (p. 753)
- 构建训练数据集 (p. 753)
- 编写和执行 Athena 查询 (p. 754)
- 删除功能组 (p. 754)

描述功能组

您可以使用`describe`function.

```
feature_group.describe()
```

列出功能组

您可以使用列出所有功能组`list_feature_groups`function.

```
sagemaker_client.list_feature_groups()
```

将记录放入要素组

您可以使用ingest函数来加载要素数据。您可以传入要素数据的数据框，设置工作人员的数量，然后选择等待它返回或不返回。以下示例演示如何使用ingestfunction.

```
feature_group.ingest(
    data_frame=feature_data, max_workers=3, wait=True
)
```

对于你拥有的每个功能组，运行ingest函数用于要加载的要素数据。

从功能组获取记录

您可以使用get_record函数，用于通过记录标识符检索特定要素的数据。以下示例使用示例标识符来检索记录。

```
record_identifier_value = str(2990130)
featurestore_runtime.get_record(FeatureGroupName=transaction_feature_group_name,
 RecordIdentifierValueAsString=record_identifier_value)
```

欺诈检测示例中的一个示例响应：

```
...
'Record': [{'FeatureName': 'TransactionID', 'ValueAsString': '2990130'},
  {'FeatureName': 'isFraud', 'ValueAsString': '0'},
  {'FeatureName': 'TransactionDT', 'ValueAsString': '152647'},
  {'FeatureName': 'TransactionAmt', 'ValueAsString': '75.0'},
  {'FeatureName': 'ProductCD', 'ValueAsString': 'H'},
  {'FeatureName': 'card1', 'ValueAsString': '4577'},
...
```

生成 Hive DDL 命令

SageMaker Python 开发工具包FeatureStore类还提供了生成 Hive DDL 命令的功能。表的架构是根据功能定义生成的。列以功能名称命名，数据类型根据要素类型推断。

```
print(feature_group.as_hive_ddl())
```

输出示例：

```
CREATE EXTERNAL TABLE IF NOT EXISTS sagemaker_featurestore.identity-feature-
group-27-19-33-00 (
  TransactionID INT
  id_01 FLOAT
  id_02 FLOAT
  id_03 FLOAT
  id_04 FLOAT
 ...
```

构建训练数据集

功能商店会自动构建Amazon Glue创建要素组时的数据目录，如果需要，可以将其关闭。下面介绍了如何使用本主题前面创建的身份和事务要素组的要素值创建单个训练数据集。此外，以下内容介绍了如何运行 Amazon Athena 查询来加入存储在离线商店中的身份和交易功能组中的数据。

首先，请使用创建 Athena 查询`athena_query()`适用于身份和交易功能组。`table_name` 是Amazon Glue由要素商店自动生成的表。

```
identity_query = identity_feature_group.athena_query()
transaction_query = transaction_feature_group.athena_query()

identity_table = identity_query.table_name
transaction_table = transaction_query.table_name
```

编写和执行 Athena 查询

您可以在这些要素组上使用 SQL 编写查询，然后使用`.run()`命令并指定要保存在该数据集的 S3 存储桶位置。

```
# Athena query
query_string = 'SELECT * FROM "'+transaction_table+'" LEFT JOIN "'+identity_table+'" ON
 "'+transaction_table+'".transactionid = "'+identity_table+'".transactionid'

# run Athena query. The output is loaded to a Pandas dataframe.
dataset = pd.DataFrame()
identity_query.run(query_string=query_string,
 output_location='s3://'+default_s3_bucket_name+'/query_results/')
identity_query.wait()
dataset = identity_query.as_dataframe()
```

从这里你可以使用此数据集训练模型，然后执行推理。

删除功能组

您可以使用删除功能组`delete`function.

```
feature_group.delete()
```

以下代码示例来自于欺诈检测示例。

```
identity_feature_group.delete()
transaction_feature_group.delete()
```

有关更多信息，请参阅 。删除功能组 API

向 IAM 角色添加所需的策略

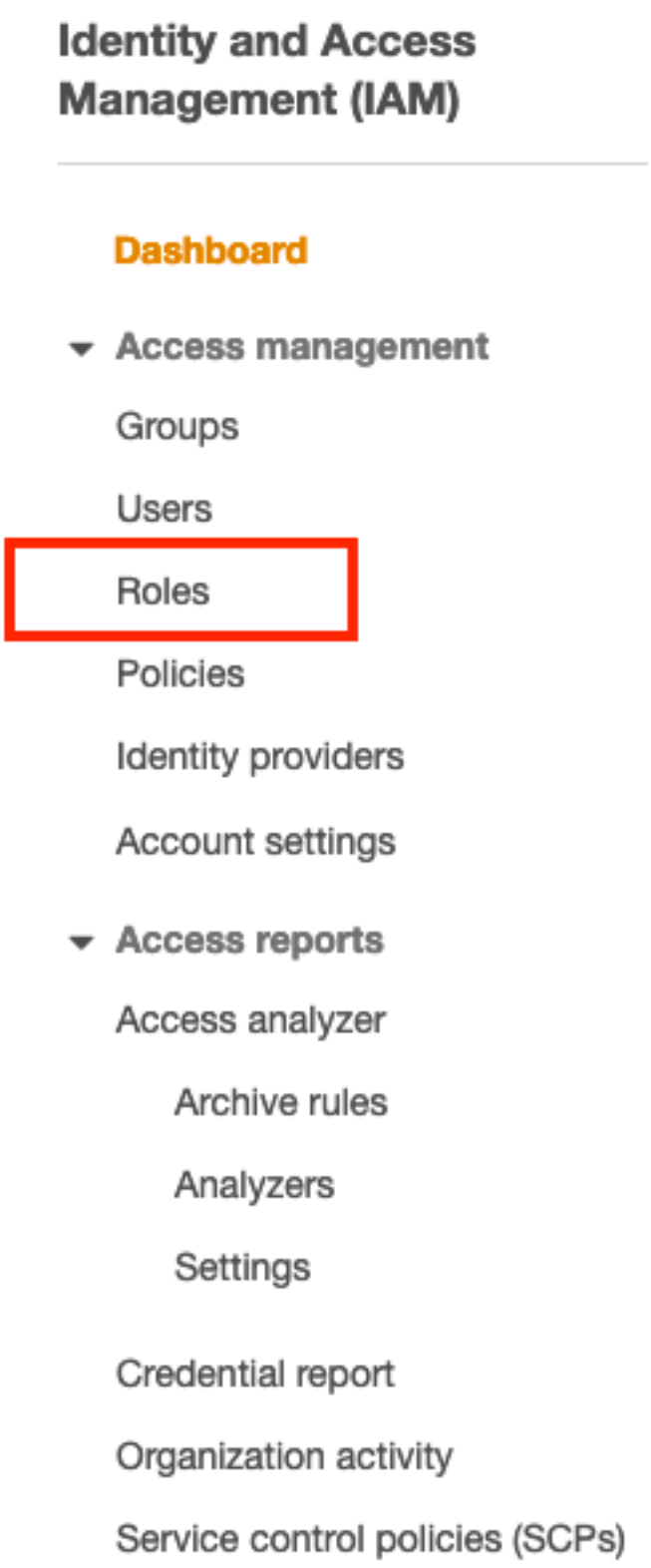
要开始使用 Amazon SageMaker 功能商店，您必须向角色添加所需的策略，`AmazonSageMakerFeatureStoreAccess`. 下面是介绍如何通过控制台将其添加到角色的演练。

第 1 步：访问Amazon管理控制台

登录到Amazon打开管理控制台，IAM 控制台.

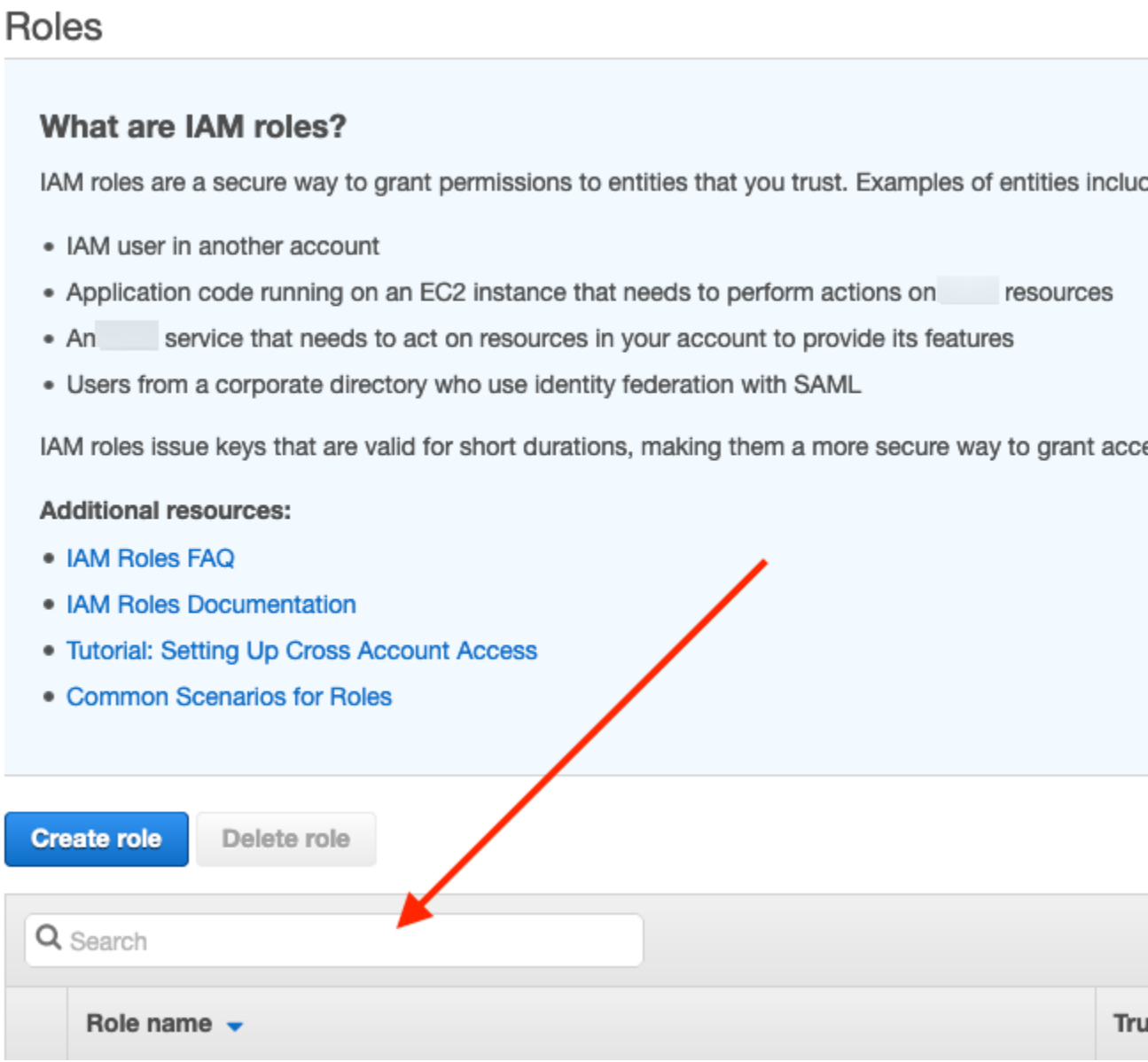
第 2 步：选择 Roles

在左侧的导航窗格中，选择角色。下面说明了导航窗格角色用红色概述。


Identity and Access Management (IAM)
Dashboard
Access management
Groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access analyzer
Archive rules
Analyzers
Settings
Credential report
Organization activity
Service control policies (SCPs)

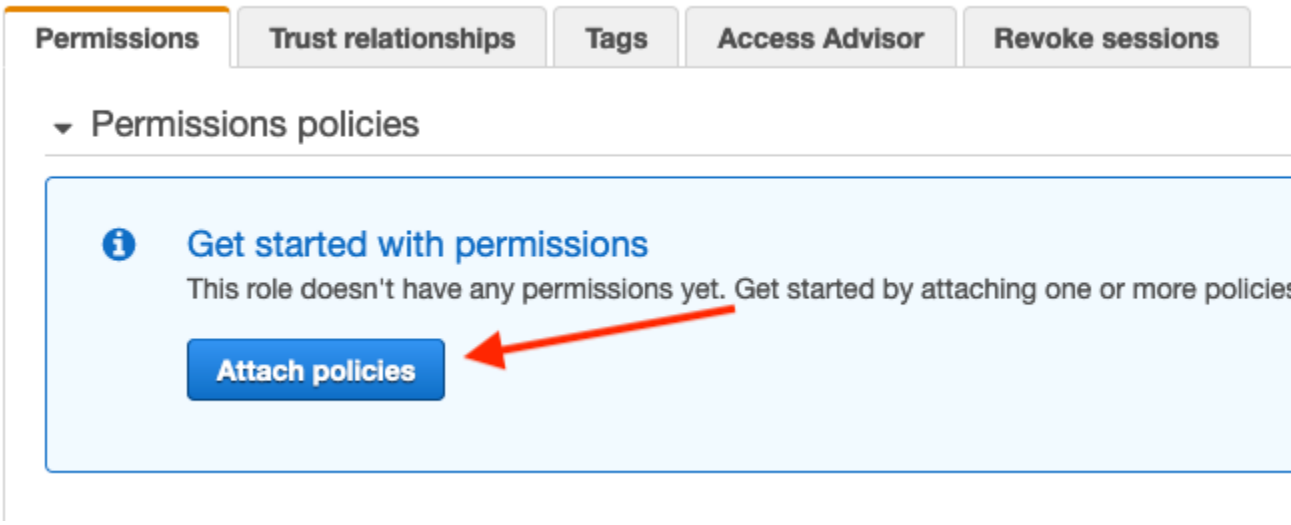
第 3 步：查找您的角色

在搜索栏中，输入您在 Amazon SageMaker 功能商店中使用的角色。下面用红色箭头说明了您进入角色的位置。

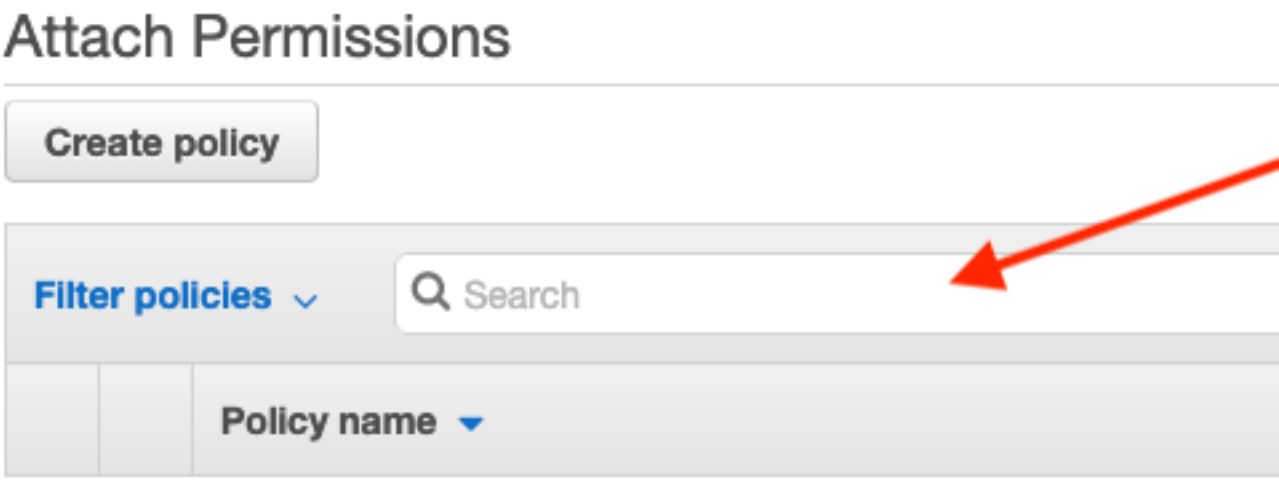


第 4 步：附加策略

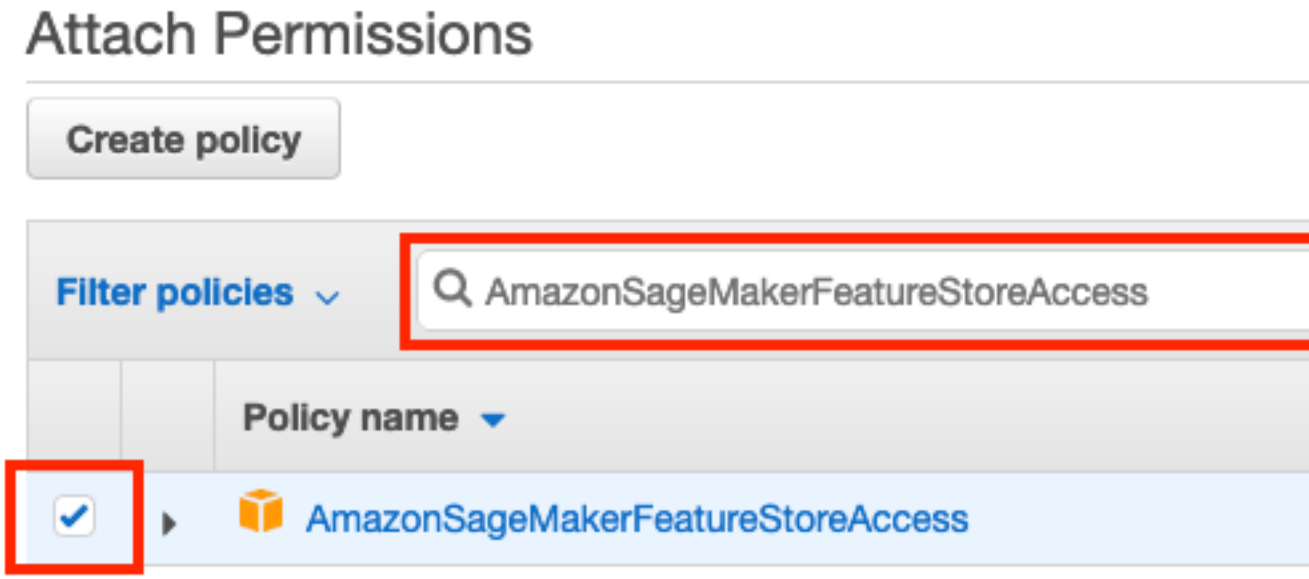
找到你的角色后，选择附加策略. 下面用红色箭头说明了这一点。



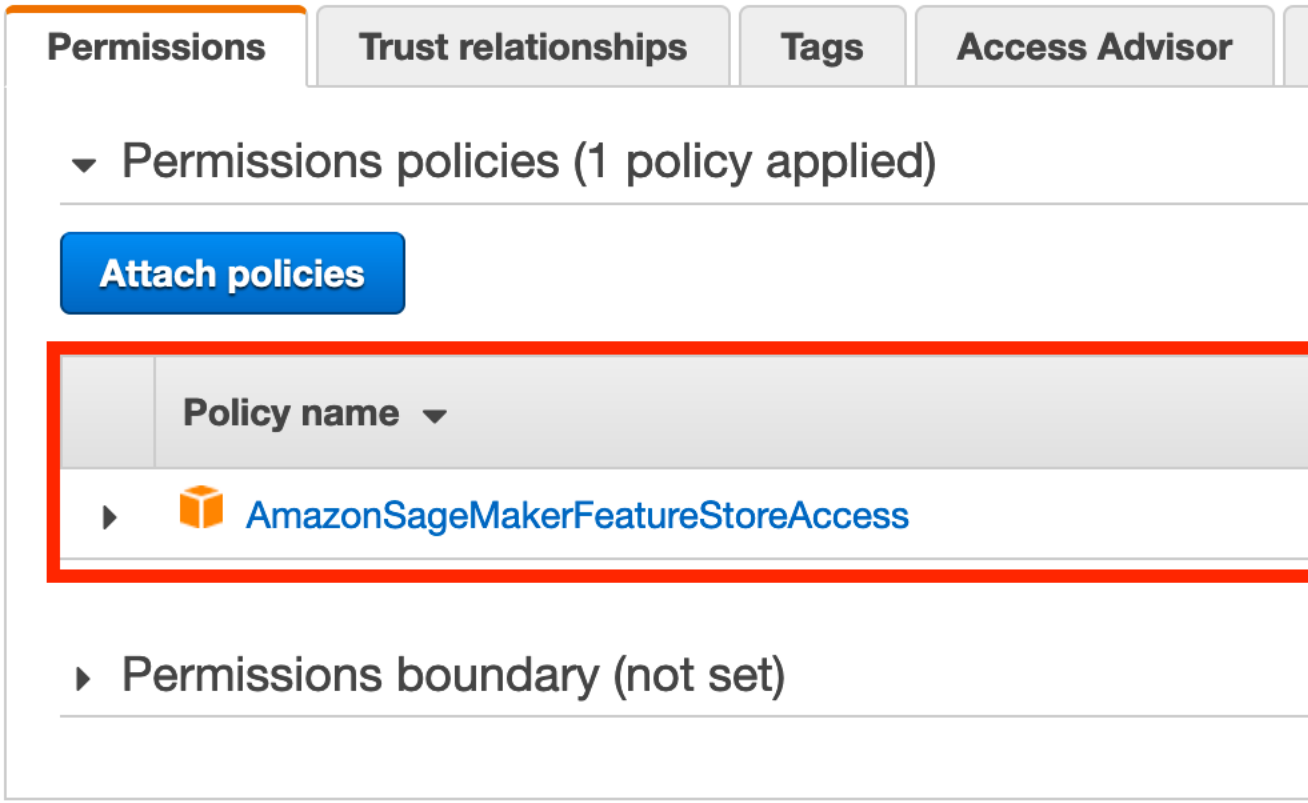
接下来，您将在搜索栏中输入所需的政策。下面说明了用于输入策略的搜索栏。



你需要添加的政策是`AmazonSageMakerFeatureStoreAccess`. 输入策略后，请选择复选框然后选择附加策略. 下面说明了这一步骤。



在您将两个策略附加到角色之后，该策略应显示在 IAM 角色下方。下面说明了它在 IAM 角色下的显示方式。



在 Amazon SageMaker Studio 中使用 Amazon SageMaker 功能商

您可以使用 Studio 创建和查看有关功能组的详细信息。

主题

- 在 Studio 中创建功能组 (p. 759)
- 在 Studio 中查看功能组详细信息 (p. 764)

在 Studio 中创建功能组

Studio 中的创建要素组过程有四个步骤：输入详细信息、定义、所需功能和标签。

在开始之前请考虑以下选项：

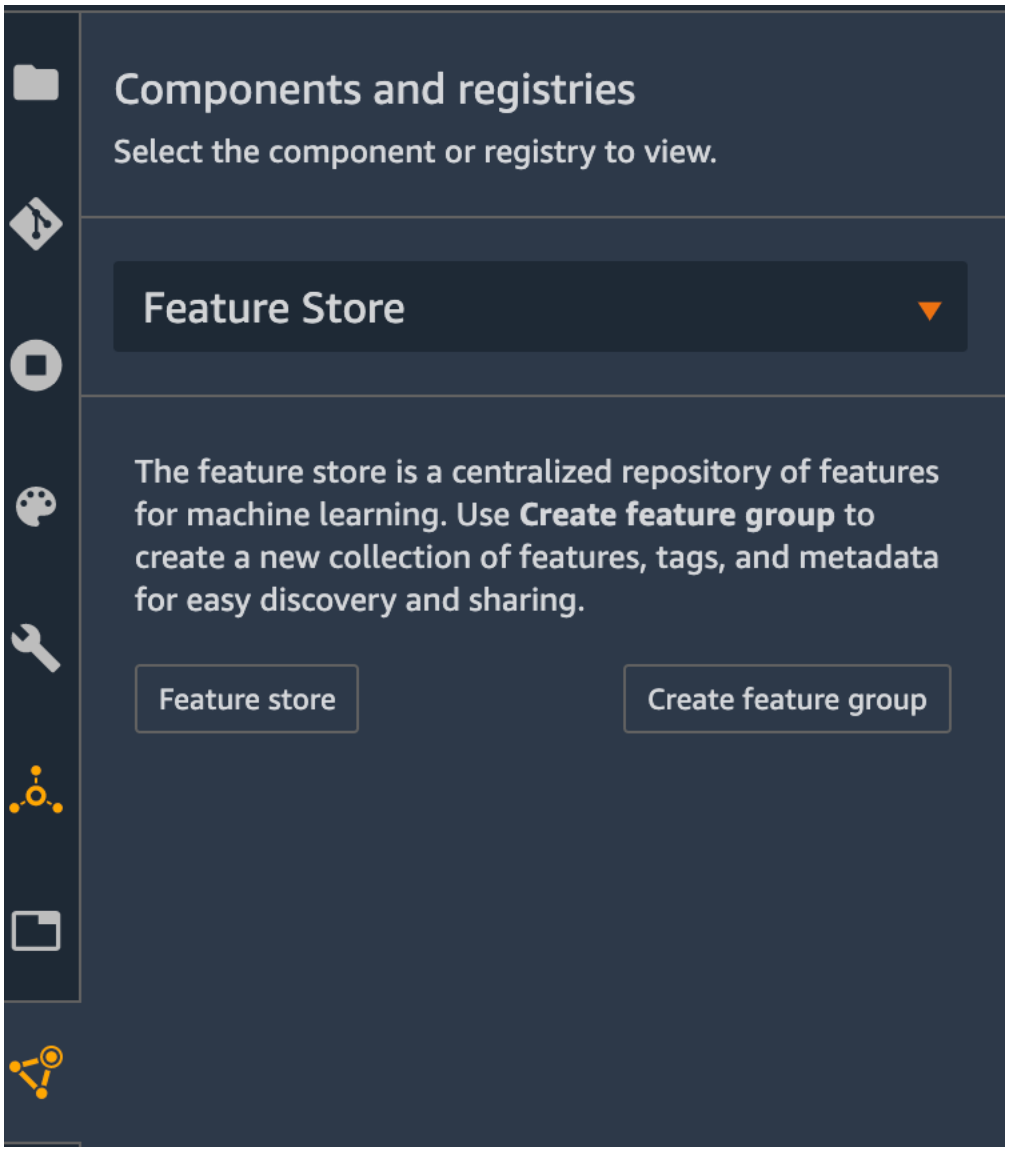
- 如果你打算只使用在线商店，你只需要功能的架构即可。这是你的列和每列的数据类型。
- 如果您计划使用离线商店，则需要 S3 存储桶 URI 和角色 ARN。
- 如果您计划使用加密功能，则需要 KMS 密钥。您可以对在线商店和离线商店使用相同的密钥，也可以为每个商店使用唯一的密钥。
- 如果您计划使用Amazon Glue集成时，请准备提供数据目录名称、数据库名称和表名。

在 Studio 中创建功能组

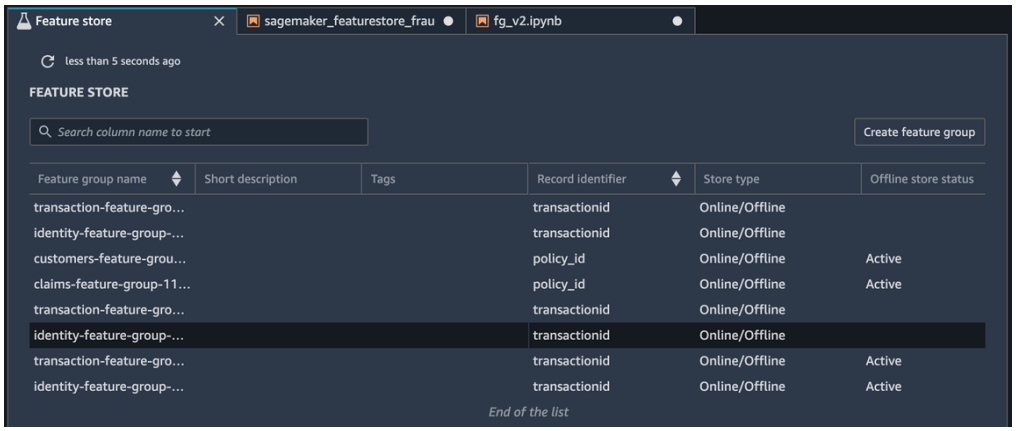
1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。

2. 在左侧导航窗格中，选择组件和注册表图标 ( )。

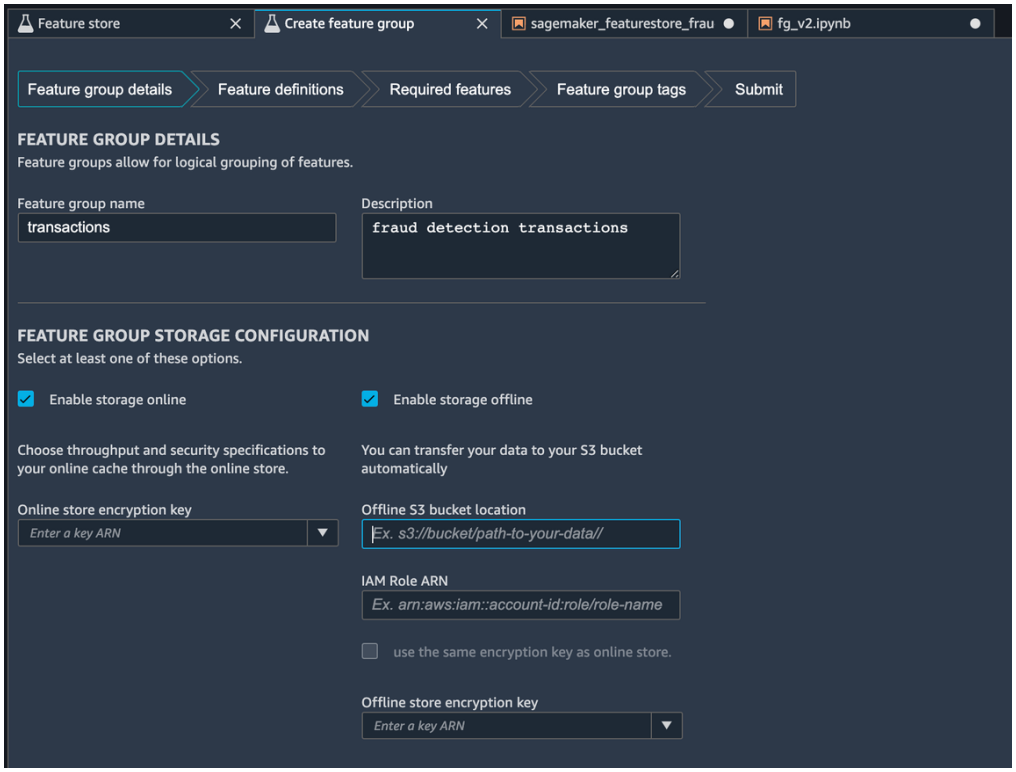
3. 在文件和资源浏览器中，选择功能商店.



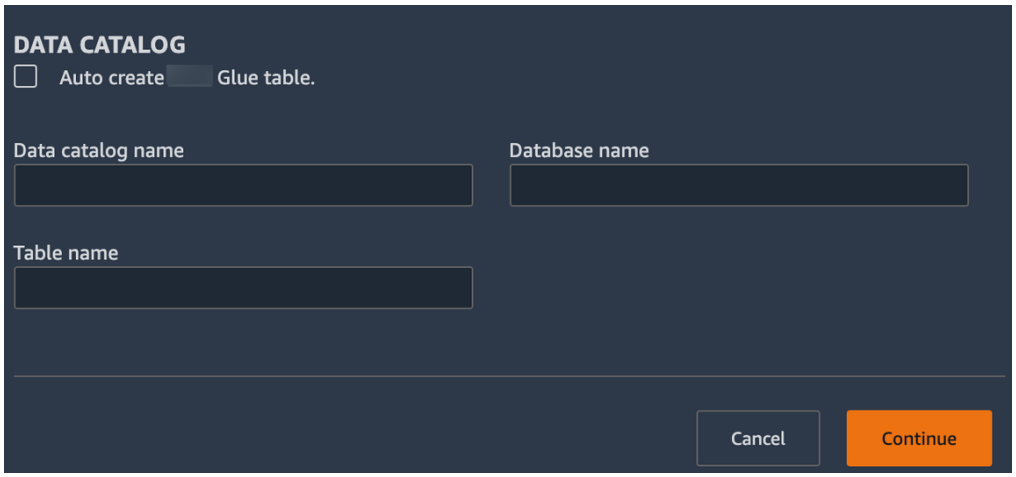
4. 这些区域有：功能商店选项卡上列出功能组。



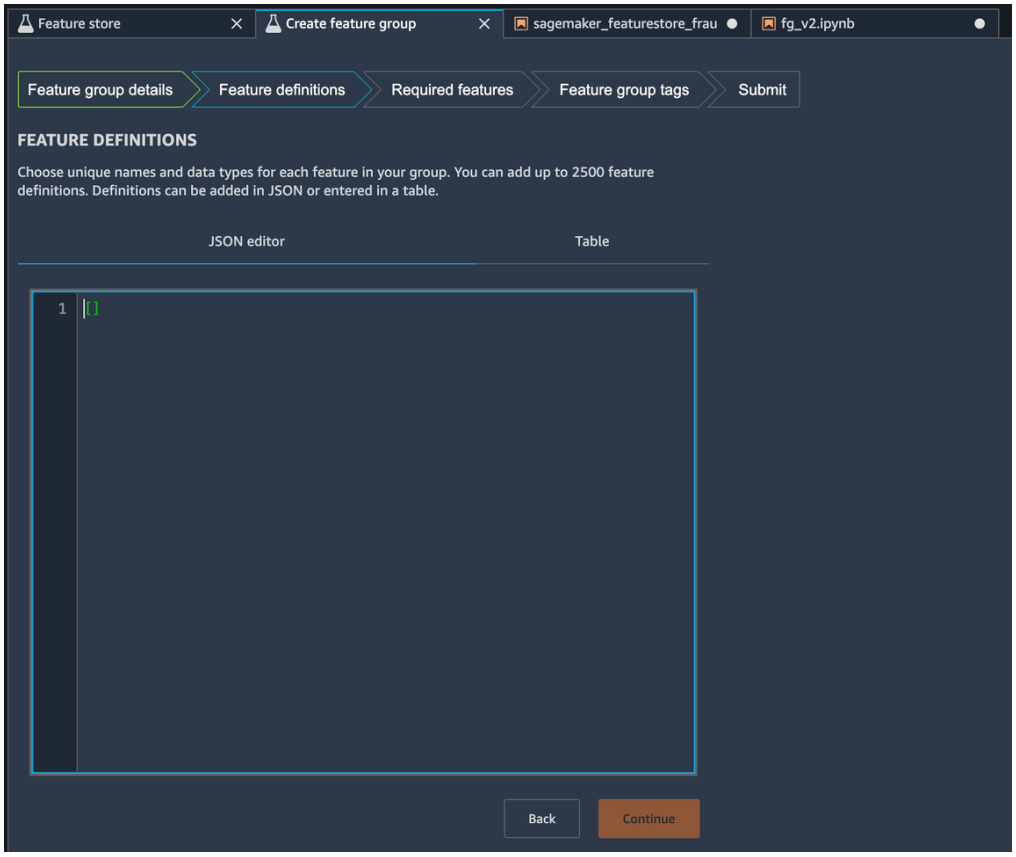
5. 这些区域有：功能商店选项卡上，选择创建功能组.
6. 输入要素组详细信息。



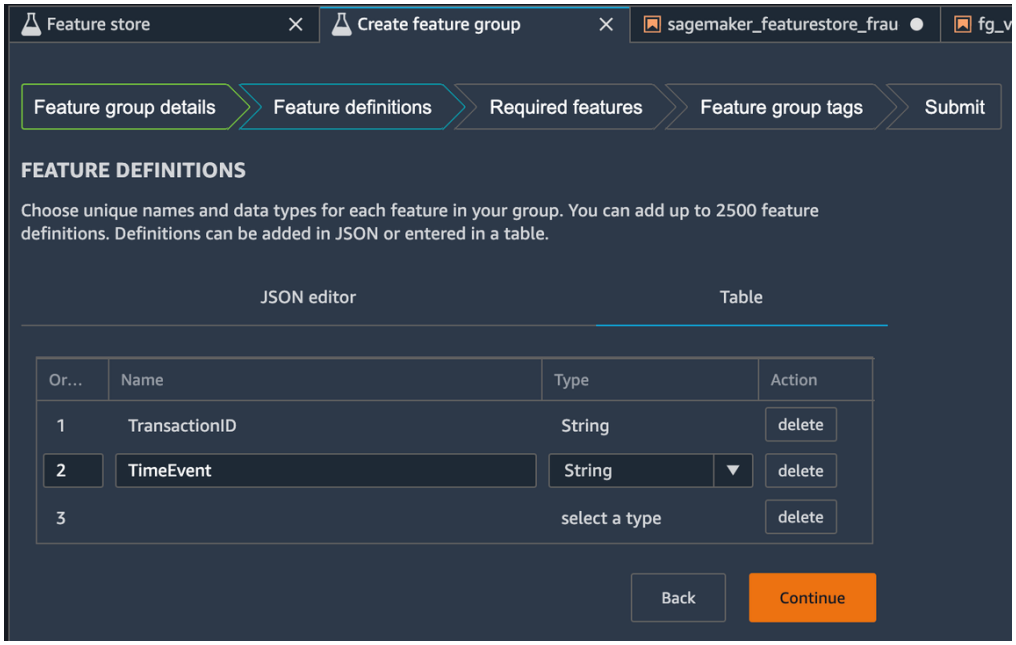
7. （可选）如果您使用的是 Glue，请输入数据目录详细信息。



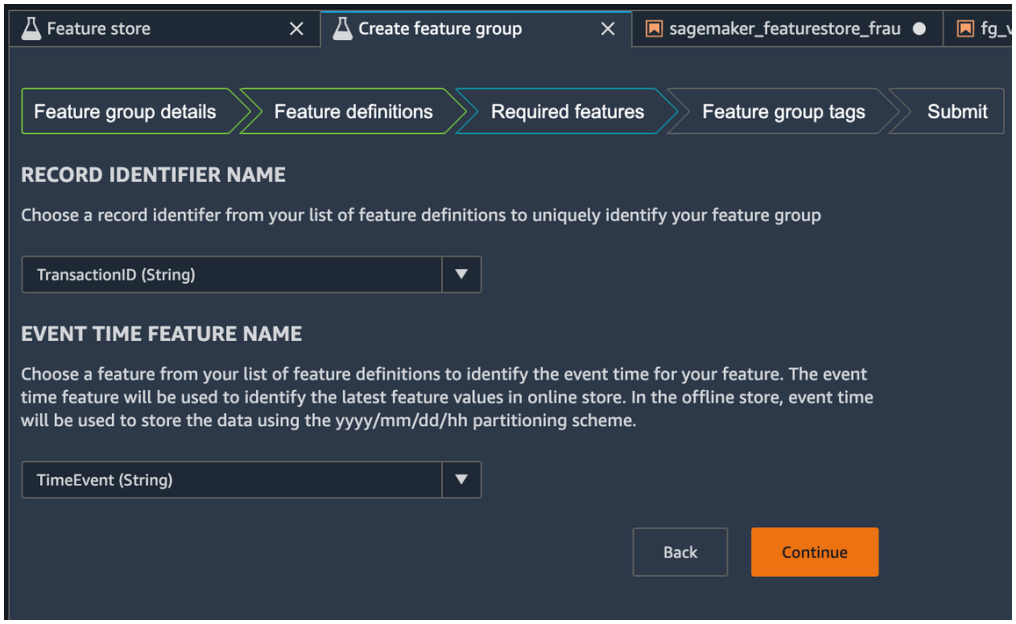
8. 输入功能定义。为功能提供架构有两种选择：JSON 编辑器或表编辑器。



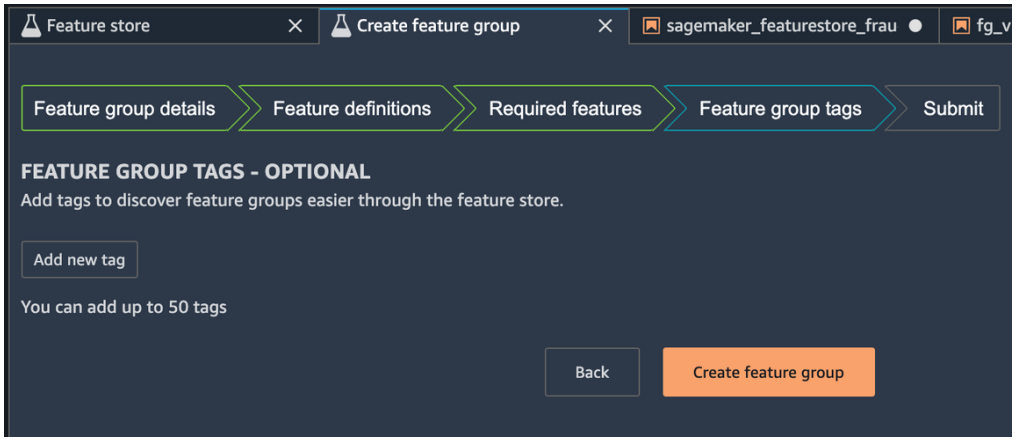
9. 表编辑器接受列名和数据类型。在最小的例子中，下一步至少需要两个：一个用于记录标识符，另一个用于时间事件功能。您最多可拥有 2,500 个功能定义。



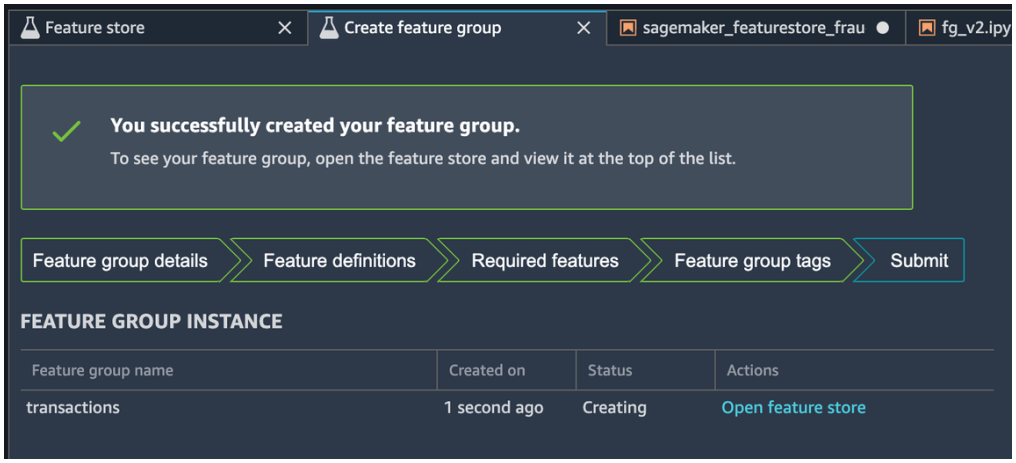
10. 设置要用于此功能组的记录标识符和时间要素。



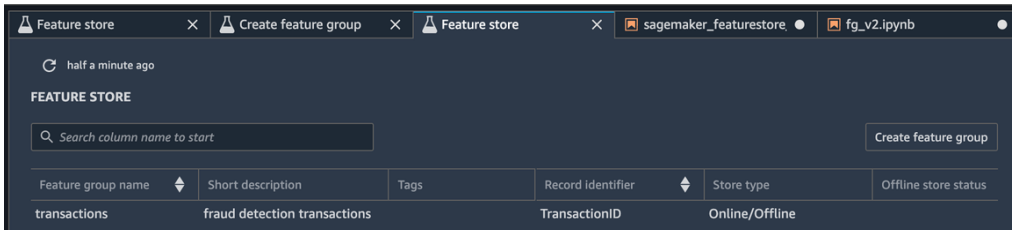
11. （可选）输入标签作为键值对。



12. 选择创建功能组.



13. 在操作列中，选择打开 feature store.



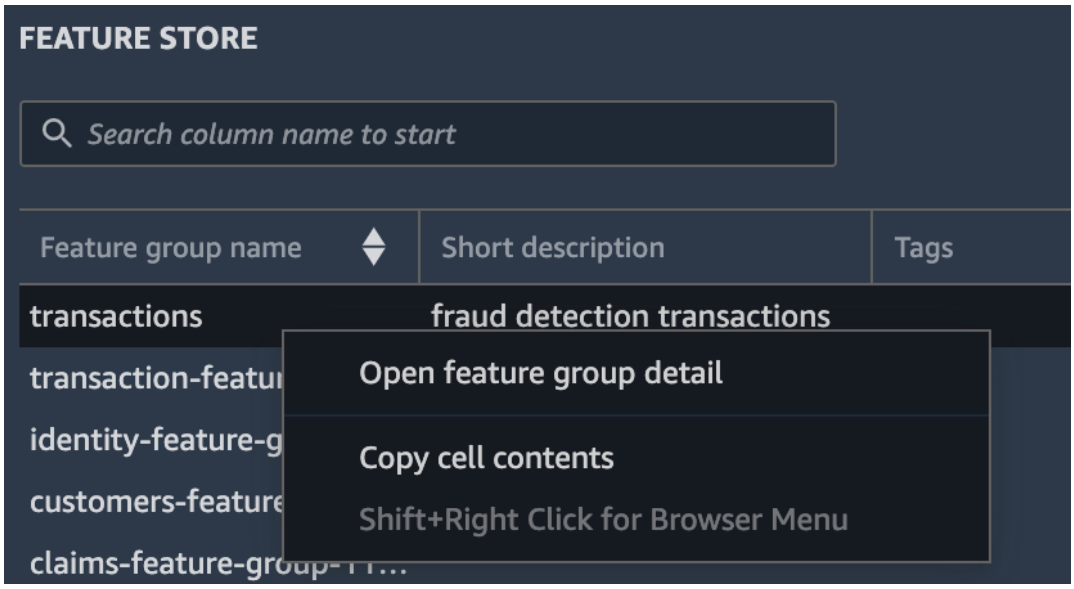
14. 完成创建要素组后，它将显示在要素组列表中。选择刷新按钮可刷新列表。

在 Studio 中查看功能组详细信息

您可以查看有关要素组的详细信息，并获取针对数据源运行的示例查询，以收集已定义要素的数据。

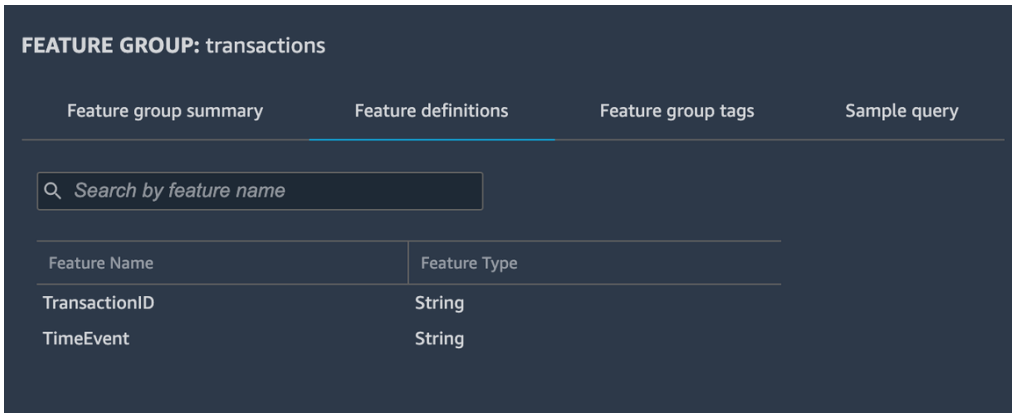
在 Studio 中查看功能组详细信息

1. 双击或右键单击列表中的要素组，然后选择打开功能组详细信息.

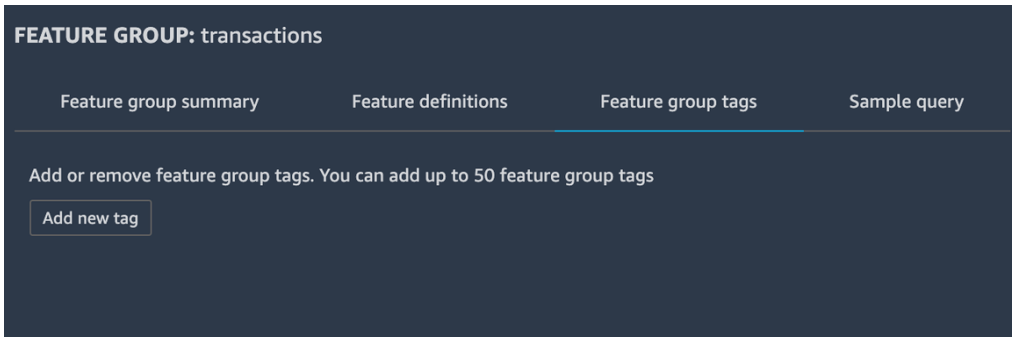


2. 在详细信息视图中，您可以查看功能组摘要、功能定义、功能组标签和示例查询。

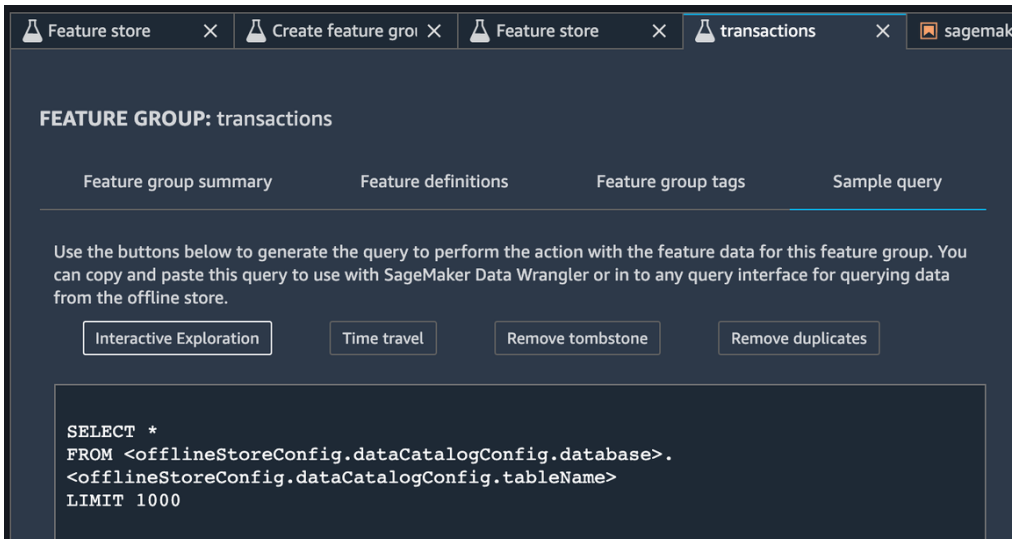
3. 在存储库的功能定义选项卡中，您可以按名称搜索要素。



4. 在存储库的功能组标签选项卡上，您可以添加和删除标签。



5. 在存储库的示例查询选项卡中，你可以看到各种示例查询：交互式探索、时间旅行、移除墓碑和删除重复项。



数据源和摄取

有多种方法可以将您的数据带入 Amazon SageMaker 功能商店。功能商店为数据摄取提供了一个名为`PutRecord`这使您能够批量或从流媒体源中提取数据。您可以使用 Amazon SageMaker Data Wrangler 来设计功能，然后将功能提取到功能商店。您还可以使用 Amazon EMR 通过 Spark 连接器批量提取数据。

主题

- 直播摄取 (p. 767)
- 使用功能商店的数据 Wrangler (p. 767)
- Batch 摄取 Spark 连接器设置 (p. 769)

直播摄取

您可以使用 Kafka 或 Kinesis 等流媒体源作为数据源，其中要素从那里提取并直接馈送到在线 feature store 进行训练、推理或功能创建。可以通过调用同步将记录推送到 feature store`PutRecord`API 调用。由于这是同步 API 调用，因此允许在单个 API 调用中推送小批量更新。这使您能够保持功能值的高新鲜度，并在检测到更新后立即发布值。这些查询也称为流功能。

使用功能商店的数据 Wrangler

Data Wrangler 是 Studio 的一项功能，它为导入、准备、转换、特征化和分析数据提供端到端解决方案。Data Wrangler 使您能够设计功能并将其提取到 feature store 中。

在 Studio 中，在与数据 Wrangler 互动后，选择Export选项卡上，选择导出步骤，然后选择功能商店，如以下屏幕截图所示。这将导出一个包含所有源代码的 Jupyter 笔记本，以创建 feature store 功能组，该功能组将 Data Wrangler 中的功能添加到离线或在线功能商店。

创建功能组后，您还可以选择并加入多个功能组之间的数据，以便在 Data Wrangler 中创建新的设计功能，然后将数据集导出到 S3 存储桶。

有关如何导出到功能 Store 的更多信息，请参阅导出到 SageMaker Feature Store.

Batch 摄取 Spark 连接器设置

介绍

亚马逊 SageMaker 功能商店支持使用现有 ETL 管道或 Amazon EMR 上的管道使用 Spark 批量提取数据。您还可以使用 Amazon 提供的此功能 SageMaker 笔记本实例。

为 Python 和 Scala 提供了安装和实施批量数据提取的方法。Python 开发人员可以使用`Amazon SageMaker-feature-store-pyspark`用于本地开发、在 Amazon EMR 上安装或从 Jupyter 笔记本电脑运行的 Python 库。Scala 开发人员可以使用 Maven 中提供的功能商店 Spark 连接器。

安装

Scala 用户

要求

- Spark >= 3.0.0
- 斯卡拉 >= 2.12.x
- 亚马逊 EMR > 6.x（仅当您使用亚马逊 EMR 时）

在 POM.xml 中声明依赖项

功能商店 Spark 连接器可在Maven 中央存储库. 在你的项目中声明以下内容`POM.xml`：

```
<dependency>
<groupId>software.amazon.sagemaker.featurestore</groupId>
<artifactId>sagemaker-feature-store-spark-sdk_2.12</artifactId>
<version>1.0.0</version>
</dependency>
```

Python 用户

要求

- PySpark >= 3.0.0
- Python >= 3.6
- 亚马逊 EMR > 6.x（仅当您使用亚马逊 EMR 时）

Python 开发人员可以使用库，sagemaker 功能商店 pyspark. 以下部分介绍如何在本地、Amazon EMR 上以及如何在 Amazon SageMaker 上安装库。

我们建议设置`$SPARK_HOME`到已安装 Spark 的目录中。在安装过程中，我们将一些必需的 jar 文件上传到`SPARK_HOME`，以便所有依赖项都将自动加载。需要 Spark 启动 JVM 才能做到这一点 PySpark 库工作。

本地安装

要查找有关安装的更多信息，请通过附加启用详细模式`--verbose`转到以下安装命令。

```
pip3 install Amazon SageMaker-feature-store-pyspark --no-binary :all:
```

在亚马逊 EMR 上安装

使用最新版本的容器（版本 6）创建集群并启用 SSH 以进行故障排除。

您可以创建自定义步骤以开始库安装，也可以通过 SSH 连接到集群以直接在控制台中安装库。

```
    export SPARK_HOME=/usr/lib/spark
    sudo -E pip3 install sagemaker-feature-store-pyspark --no-binary :all: --verbose
```

注意：如果您想自动安装依赖罐SPARK_HOME，请勿使用亚马逊 EMR 的引导步骤。

在亚马逊上安装 SageMaker 笔记本实例

亚马逊 SageMaker 笔记本实例使用的是与功能商店 Spark 连接器不兼容的旧版本的 Spark。你必须升级 Spark，然后安装sagemaker-feature-store-pyspark.

在笔记本中，添加并运行如下所示的单元格：

```
import os

original_spark_version = "2.4.0"
os.environ['SPARK_HOME'] = '/home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-
packages/pyspark'

# Install a newer versiion of Spark which is compatible with spark library
!pip3 install pyspark==3.1.1
!pip3 install sagemaker-feature-store-pyspark --no-binary :all:
```

示例实施

斯卡拉

功能存储批处理站 .Scala

```
import software.amazon.sagemaker.featurestore.sparksdk.FeatureStoreManager
import org.apache.spark.sql.types.{StringType, StructField, StructType}
import org.apache.spark.sql.{Row, SparkSession}

object ProgramOffline {
  def main(args: Array[String]): Unit = {

    val spark = SparkSession.builder().getOrCreate()

    // Construct test DataFrame
    val data = List(
      Row("1", "2021-07-01T12:20:12Z"),
      Row("2", "2021-07-02T12:20:13Z"),
      Row("3", "2021-07-03T12:20:14Z")
    )

    val schema = StructType(
      List(StructField("RecordIdentifier", StringType), StructField("EventTime",
 StringType))
    )

    val df = spark.createDataFrame(spark.sparkContext.parallelize(data), schema)
    val featureStoreManager = new FeatureStoreManager()
```

```
    // Load the feature definitions from input schema. The feature definitions can be used
 to create a feature group
    val featureDefinitions = featureStoreManager.loadFeatureDefinitionsFromSchema(df)

    val featureGroupArn = "arn:aws:Amazon SageMaker:us-west-2:<your-account-id>:feature-
group/<your-feature-group-name>"

    // Ingest by default
    featureStoreManager.ingestData(df, featureGroupArn)

    // Offline store direct ingestion, flip the flag of direct_offline_store
    featureStoreManager.ingestData(df, featureGroupArn, directOfflineStore = true)
  }
}
```

Python

FeatureStoreBatchIngestion.py

```
from pyspark.sql import SparkSession
from feature_store_pyspark.FeatureStoreManager import FeatureStoreManager
import feature_store_pyspark

extra_jars = ",".join(feature_store_pyspark.classpath_jars())
spark = SparkSession.builder \
.config("spark.jars", extra_jars) \
.getOrCreate()

# Construct test DataFrame
columns = ["RecordIdentifier", "EventTime"]
data = [("1","2021-03-02T12:20:12Z"), ("2", "2021-03-02T12:20:13Z"),
 ("3", "2021-03-02T12:20:14Z")]

df = spark.createDataFrame(data).toDF(*columns)
feature_store_manager= FeatureStoreManager()

# Load the feature definitions from input schema. The feature definitions can be used to
 create a feature group
feature_definitions = feature_store_manager.load_feature_definitions_from_schema(df)

feature_group_arn = "arn:aws:Amazon SageMaker:us-west-2:<your-account-id>:feature-group/
<your-feature-group-name>"

# Ingest by default
feature_store_manager.ingest_data(input_data_frame=df, feature_group_arn=feature_group_arn)

# Offline store direct ingestion, flip the flag of direct_offline_store
feature_store_manager.ingest_data(input_data_frame=df,
 feature_group_arn=feature_group_arn, direct_offline_store=true)
```

提交 Spark Job

斯卡拉

您应该可以使用 Spark 连接器作为普通依赖项。在所有平台上运行应用程序不需要额外的指令。

Python

这些区域有： PySpark 版本需要导入额外的依赖 jar，因此需要额外的步骤才能运行 Spark 应用程序。

如果你没指定SPARK_HOME在安装过程中，你必须在运行时在 JVM 中加载所需的罐子spark-submit.feature-store-pyspark-dependency-jars是 Spark 库安装的 Python 脚本，用于自动获取所有罐子的路径。

```
spark-submit --jars `feature-store-pyspark-dependency-jars` FeatureStoreBatchIngestion.py
```

如果您在 Amazon EMR 上运行此应用程序，建议在客户端模式下运行应用程序，这样您就不需要将依赖罐子分发到其他任务节点。使用 Spark 参数在 Amazon EMR 集群中再添加一个步骤，如下所示：

```
spark-submit --deploy-mode client --master yarn s3://<path-to-script>/
FeatureStoreBatchIngestion.py
```

使用 Athena 和查询功能商店Amazon Glue

在离线 feature store 中创建功能组后，您可以选择使用 Amazon Athena 在Amazon Glue目录。这要求在数据目录中注册数据，并在功能商店中自动注册的其他目录详细信息。换言之，功能存储会自动构建Amazon Glue创建要素组时的数据目录，您可以将其关闭。当你想通过执行 SQL 查询来构建数据集然后训练模型进行推理时，这特别有用。

在FeatureStore已创建并使用离线存储中的数据填充，您可以编写 SQL 查询来加入存储在离线存储中的来自不同的数据FeatureGroups. 为此，可以使用 Amazon Athena 编写和执行 SQL 查询。可以设置 Amazon GlueCrawler 将按计划运行，以确保您的目录始终保持最新状态。

如果你想这样做，请定义一个角色，该角色可以由Amazon Glue用于访问离线商店的 S3 存储桶的 Crawler。有关更多信息，请参阅 。创建 IAM 角色.

有关如何使用的更多信息Amazon Glue和 Athena 来构建用于模型训练和推理的训练数据集，请参阅构建训练数据集：创建功能组.

示例 Athena 查询

下面我们提供了一些示例查询，作为模板，供您使用 Athena 快速编写查询。

互动探索

此查询选择前 1000 条记录。

```
SELECT *
FROM <FeatureGroup.DataCatalogConfig.DatabaseName>.<FeatureGroup.DataCatalogConfig.TableName>
LIMIT 1000
```

没有重复项的最新快照

此查询选择最新的非重复记录。

```
SELECT *
FROM
    (SELECT *,
         row_number()
        OVER (PARTITION BY <RecordIdentiferFeatureName>
    ORDER BY  <EventTimeFeatureName> desc, Api_Invocation_Time DESC, write_time DESC) AS
 row_num
```

```
    FROM
 <FeatureGroup.DataCatalogConfig.DatabaseName>.<FeatureGroup.DataCatalogConfig.TableName>)
WHERE row_num = 1;
```

离线商店中没有重复和删除记录的最新快照

此查询过滤掉所有已删除的记录并从脱机商店中选择非重复记录。

```
SELECT *
FROM
    (SELECT *,
         row_number()
        OVER (PARTITION BY <RecordIdentiferFeatureName>
    ORDER BY  <EventTimeFeatureName> desc, Api_Invocation_Time DESC, write_time DESC) AS
 row_num
    FROM
 <FeatureGroup.DataCatalogConfig.DatabaseName>.<FeatureGroup.DataCatalogConfig.TableName>)
WHERE row_num = 1 and
NOT is_deleted;
```

在离线商店中没有重复和删除记录的 Time Travel

此查询过滤掉所有已删除的记录，然后从特定时间点选择非重复记录。

```
SELECT *
FROM
    (SELECT *,
         row_number()
        OVER (PARTITION BY <RecordIdentiferFeatureName>
    ORDER BY  <EventTimeFeatureName> desc, Api_Invocation_Time DESC, write_time DESC) AS
 row_num
    FROM
 <FeatureGroup.DataCatalogConfig.DatabaseName>.<FeatureGroup.DataCatalogConfig.TableName>
    where <EventTimeFeatureName> <= timestamp '<timestamp>')
    -- replace timestamp '<timestamp>' with just <timestamp>  if EventTimeFeature is of
 type fractional
WHERE row_num = 1 and
NOT is_deleted
```

跨账户离线商店访问

Amazon SageMaker 功能商店允许用户在一个账户（账户 A）中创建功能组，并使用另一个账户（账户 B）中的 Amazon S3 存储桶使用离线商店对其进行配置。可以使用以下部分中的步骤进行设置。

主题

- 第 1 步：在账户 A 中设置离线商店访问角色 (p. 773)
- 第 2 步：在账户 B 中设置离线存储 S3 存储桶 (p. 774)
- 第 3 步：在账户 A 中设置离线商店 KMS 加密密钥 (p. 775)
- 第 4 步：在账号 A 中创建功能组 (p. 776)

第 1 步：在账户 A 中设置离线商店访问角色

首先，为 Amazon SageMaker 功能商店设置一个角色，以便将数据写入离线商店。达到这一目的最简单的方式是使用`AmazonSageMakerFeatureStoreAccess`策略或使用已经具

有AmazonSageMakerFeatureStoreAccess附加策略。本文件将本政策称为Account-A-Offline-Feature-Store-Role-ARN.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetBucketAcl",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::*SageMaker*",
                "arn:aws:s3:::*Sagemaker*",
                "arn:aws:s3:::*sagemaker*"
            ]
        }
    ]
}
```

前面的代码片段显示AmazonSageMakerFeatureStoreAccess政策。这些区域有：Resource默认情况下，策略的部分范围缩小为名称包含的 S3 存储桶SageMaker、Sagemaker，或者sagemaker. 这意味着正在使用的离线存储 S3 存储桶必须遵循此命名约定。如果情况并非如此，或者如果您想进一步缩小资源范围，则可以在控制台中将策略复制并粘贴到 S3 存储桶策略中，请自定义Resource“将” 部分arn:aws:s3:::your-offline-store-bucket-name，然后将其附加到角色。

此外，此角色必须附加 KMS 权限。至少，它需要kms:GenerateDataKey允许使用客户管理的密钥写入离线商店。请参阅步骤 3 了解为什么跨账户场景需要客户管理的密钥以及如何设置它。以下示例显示了一个内联策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:*:Account-A-Account-Id:key/*"
        }
    ]
}
```

这些区域有：Resource此策略的部分范围限定为账户 A 中的任何密钥。为了进一步缩小此范围，请在步骤 3 中设置脱机存储 KMS 密钥后，返回到此策略并将其替换为密钥 ARN。

第 2 步：在账户 B 中设置离线存储 S3 存储桶

在账户 B 中创建 S3 存储桶如果您使用的是默认值AmazonSageMakerFeatureStoreAccess策略，存储桶名称必须包括SageMaker、Sagemaker，或者sagemaker. 编辑存储桶策略，如下面的示例所示，以允许账户 A 读取和写入对象。

本文档将以下示例存储桶策略引用为Account-B-Offline-Feature-Store-Bucket.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "S3CrossAccountBucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetBucketAcl"
            ],
            "Principal": {
                "AWS": [
                    "*Account-A-Offline-Feature-Store-Role-ARN*"
                ],
            },
            "Resource": [
                "arn:aws:s3:::offline-store-bucket-name/*",
                "arn:aws:s3:::offline-store-bucket-name"
            ]
        }
    ]
}
```

在上一个策略中，委托人是Account-A-Offline-Feature-Store-Role-ARN，该角色是在步骤 1 的账户 A 中创建的角色，并提供给 Amazon SageMaker 功能商店以写入离线商店。您可以在下面提供多个 ARN 角色Principal.

第 3 步：在账户 A 中设置离线商店 KMS 加密密钥

Amazon SageMaker 功能商店确保始终为离线存储中的 S3 对象启用服务器端加密。对于跨账户使用案例，您必须提供客户管理的密钥，以便您可以控制谁可以向离线商店写信（在这种情况下，Account-A-Offline-Feature-Store-Role-ARN来自账户 A）以及谁可以从离线商店中读取（在本例中是账户 B 的身份）。

本文将以下示例密钥策略引用为Account-A-Offline-Feature-Store-KMS-Key-ARN.

```
{
    "Version": "2012-10-17",
    "Id": "key-consolepolicy-3",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::Account-A-Account-Id:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow access for Key Administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::Account-A-Account-Id:role/Administrator",
                ]
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms:List*",
                "kms:Put*",
```

```
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow Feature Store to get information about the customer managed key",
            "Effect": "Allow",
            "Principal": {
                "Service": "sagemaker.amazonaws.com"
            },
            "Action": [
                "kms:Describe*",
                "kms:Get*",
                "kms:List*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "*Account-A-Offline-Feature-Store-Role-ARN*",
                    "*arn:aws:iam::Account-B-Account-Id:root*"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:RetireGrant",
                "kms:ReEncryptFrom",
                "kms:ReEncryptTo",
                "kms:GenerateDataKey",
                "kms:ListAliases",
                "kms:ListGrants"
            ],
            "Resource": "*",
        }
    ]
}
```

第 4 步：在账号 A 中创建功能组

接下来，在账户 A 中创建功能组，并在账户 B 中使用离线存储 S3 存储桶。要执行此操作，请提供以下参数RoleArn、OfflineStoreConfig.S3StorageConfig.KmsKeyId和OfflineStoreConfig.S3StorageConfig下方式：

- 提供的Account-A-Offline-Feature-Store-Role-ARN作为RoleArn.
- 提供的Account-A-Offline-Feature-Store-KMS-Key-ARN为了OfflineStoreConfig.S3StorageConfig.KmsKeyId.
- 提供的Account-B-Offline-Feature-Store-Bucket为了OfflineStoreConfig.S3StorageConfig.S3Uri.

安全性和访问控制

借助 Amazon SageMaker 功能商店，您可以创建两种类型的商店：在线商店或离线商店。在线商店用于低延迟的实时推理用例，而离线商店则用于训练和批量推理使用案例。当您创建供在线或离线使用的要素组时，您可以提供Amazon Key Management Service用于加密所有静态数据的客户托管密钥。如果您不提供Amazon KMSkey 然后我们确保您的数据在服务器端使用Amazon拥有的Amazon KMS键或Amazon管理Amazon KMS键。创建功能组时，您可以选择存储类型并可以选择提供Amazon KMS密钥用于加密数据，然后你可以调用各种 API 进行数据管理，例如`PutRecord`、`GetRecord`、`DeleteRecord`.

功能商店允许您在功能组级别授予或拒绝个人的访问权限，并启用对功能商店的跨账户访问权限。例如，您可以设置开发人员账户以访问离线商店，以进行模型训练和探索，而这些账户没有对生产账户的写入权限。您可以设置生产账户以访问线上和线下商店。功能商店使用独特的客户Amazon KMS用于离线和在线存储静态数据加密的密钥。访问控制可通过 API 和Amazon KMS密钥访问。您还可以创建要素组级访问控制。

有关客户托管密钥的更多信息，请参阅客户托管密钥. 有关 Amazon KMS 的更多信息，请参阅Amazon KMS。

使用Amazon KMSAmazon SageMaker Feature Store 的权限

静态加密保护下的功能商店Amazon KMS客户托管密钥。默认情况下，它将Amazon拥有的在线商店的客户管理密钥和AmazonOblineStore 的客户托管密钥. 功能商店支持在下面加密在线或离线商店的选项客户托管密钥. 创建在线或离线商店时，您可以为功能商店选择客户管理的密钥，而且每个商店可能会有所不同。

仅支持要素商店对称客户托管密钥. 您无法使用非对称客户托管密钥加密在线或离线商店中的数据。要获取确定客户托管密钥是对称还是非对称的帮助，请参阅识别对称密钥和非对称客户托管密钥.

当您使用客户托管密钥时，您可以利用以下功能：

- 您创建和管理客户管理的密钥，包括设置密钥策略、IAM 策略和GRANTS以控制对客户托管密钥的访问。您可以启用和禁用客户管理的密钥，启用和禁用自动密钥轮换, 和删除客户托管密钥当它不再使用时。
- 您可以使用具有导入的密钥材料的客户托管密钥，或者您拥有和管理的自定义密钥存储中的客户托管密钥。
- 您可以通过检查对以下的 API 调用来审核您的在线或离线商店的加密和解密。Amazon KMS在AmazonCloudTrail 日志.

您无需支付的月费Amazon拥有客户托管密钥。客户托管密钥将产生费用对于每次 API 调用和Amazon Key Management Service配额适用于每个客户托管密钥。

授权为您的在线商店使用客户管理密钥

如果您使用客户托管密钥为保护您的在线商店，该客户托管密钥的策略必须赋予功能商店代表您使用该功能商店的权限。您可以完全控制客户托管密钥的策略和授权。

要素商店无需额外授权即可使用默认设置。Amazon拥有 KMS 密钥保护您的在线或线下商店Amazonaccount.

客户托管密钥策略

当你选择客户托管密钥为保护您的在线商店，功能商店必须具有代表做出选择的委托人使用客户托管密钥的权限。该委托人（用户或角色）必须具有功能商店所需的客户托管密钥权限。您可以在密钥策略、IAM 策略或授权中提供这些权限。要素 Store 至少需要对客户托管密钥具备以下权限：

- “KMS：加密”、“KMS: Decrypt”、“KMS: describe Key”、“KMS: 创建格兰特”、“KMS: 退休补助金”、“KMS: 重新加密”、“KMS: RecryptTo”、“KMS: ListGrant”、“KMS: ListGrant”、“KMS: ListGrant”、“KMS: ListGrant””

例如，以下示例密钥策略仅提供所需的权限。该策略具有以下效果：

- 允许功能商店在加密操作中使用客户托管密钥并创建授权，但仅当它代表账户中具备功能商店使用权限的委托人行事时才可如此。如果策略语句中指定的委托人无权使用您的功能商店，调用将失败，即使调用来自功能商店服务也是如此。
- 这些区域有：kms:ViaService条件键仅当 FeatuRestore 代表策略语句中所列委托人发出请求时，才允许权限。这些委托人不能直接调用这些操作。的价值`kms:ViaService`应为`sagemaker.*.amazonaws.com`.

  Note

  这些区域有：`kms:ViaService`状况密钥只能用于客户管理的在线商店Amazon KMSkey，且不能用于离线商店。如果您将此特殊条件添加到客户管理的密钥中，并使用相同的Amazon KMS在线和离线商店的 key，那么它将失败`CreateFeatureGroup`API 操作。
- 赋予客户托管密钥管理员对客户托管密钥的只读访问权限，以及撤销授权的权限，包括功能商店保护数据所使用的授权。

在使用示例密钥策略之前，请将示例委托人替换为 Amazon 账户中的实际委托人。

```
{"Id": "key-policy-feature-store",
   "Version":"2012-10-17",
   "Statement": [
     {"Sid" : "Allow access through Amazon SageMaker Feature Store for all principals in
 the account that are authorized to use  Amazon SageMaker Feature Store",
       "Effect": "Allow",
       "Principal": {"AWS": "arn:aws:iam::111122223333:user/featurestore-user"},
       "Action": [
         "kms:Encrypt",
         "kms:Decrypt",
         "kms:DescribeKey",
         "kms:CreateGrant",
         "kms:RetireGrant",
         "kms:ReEncryptFrom",
         "kms:ReEncryptTo",
         "kms:GenerateDataKey",
         "kms:ListAliases",
         "kms:ListGrants"
       ],
       "Resource": "*",
       "Condition": {"StringLike": {"kms:ViaService" : "sagemaker.*.amazonaws.com"
          }
       }
     },
     {"Sid":  "Allow administrators to view the customer managed key and revoke grants",
       "Effect": "Allow",
       "Principal": {"AWS": "arn:aws:iam::111122223333:role/featurestore-admin"
        },
       "Action": [
         "kms:Describe*",
         "kms:Get*",
         "kms:List*",
         "kms:RevokeGrant"
       ],
       "Resource": "*"
     },
     {"Sid": "Enable IAM User Permissions",
       "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::123456789:root"
        },
        "Action": "kms:*",
        "Resource": "*"
     }
```

```
    ]
}
```

使用授权来授权特色商店

除密钥策略以外，功能商店还使用授权来对客户托管密钥设置权限。要查看有关您的账户中客户托管密钥的授权，请使用`ListGrants`operation. 要素商店无需授权或任何额外权限即可使用Amazon客户拥有托管密钥以保护你的在线商店。

功能存储在执行后台系统维护和连续数据保护任务时使用授予权限。

每项授权都特定于在线商店。如果账户中包含多个使用同一客户管理密钥加密的商店，则每个账户将有唯一的授权`FeatureGroup`使用相同的客户托管密钥。

该密钥策略还可以允许账户撤销授予在客户托管密钥上。但是，如果您对某个活动加密在线商店撤销授权，功能商店将无法保护和维护该商店。

监控功能商店与之交互Amazon KMS

如果您使用客户托管密钥为了保护你的在线或线下商店，你可以使用Amazon CloudTrail用于跟踪功能商店发送到的请求的日志Amazon KMS代表您。

访问在线商店中的数据

这些区域有：调用者（IAM 用户或 IAM 角色）到所有数据平面操作（放置、获取、删除记录）必须具备客户托管密钥的以下权限：

```
"kms:Decrypt"
```

授权为离线商店使用客户管理密钥

这些区域有：roleArn这将作为参数传递到`createFeatureGroup`必须对离线商店 KmsKeyId 具有以下权限：

```
"kms:GenerateDataKey"
```

Note

在线商店的关键政策也适用于离线商店，只有在`kms:ViaService`未指定条件。

Important

您可以指定Amazon KMS加密密钥，用于在创建功能组时加密用于离线要 feature store 的 Amazon S3 位置。如果Amazon KMS未指定加密密钥，默认情况下，我们使用Amazon KMS键。通过定义存储桶级密钥对于上证，你可以减少Amazon KMS请求成本高达 99%。

配额、命名规则和数据类型

限制和配额

Note

可以根据您的需求提高软限制。

- 每个要素组的最大数目Amazon账户：软限制为 100。
- 每个要素组的功能定义的最大数量：2500。
- 每个 API 的每秒事务数 (TPS)Amazon账户：每个 API 的软限制为 10000 TPS，不包括`BatchGetRecord`API 调用，软限制为 500 TPS。
- 记录的最大大小：350KB。
- 记录标识符的最大大小：2KB。
- 要素值的最大大小：350KB。
- 并发要素组创建工作流的最大数目：4.
- BatchgetRecord API：最多可以包含 100 条记录，最多可以查询 10 个要素组。

命名规则

- 保留关键字：以下是保留字，不能用作要素定义中的功能名称：`is_deleted`、`write_time`, 和`api_invocation_time`.

数据类型

- 字符串功能类型：字符串是使用 UTF-8 二进制编码的 Unicode。字符串的最小长度可以为零，最大长度受记录的最大大小的约束。
- 小数功能类型：小数要素值必须符合双精度浮点数，如IEEE 754 标准.
- 积分功能类型：要素存储支持 64 位有符号整数范围内的整数。最小值为 -2^{63}和最大值：$2^{63}-1$。
- 活动时间特点：所有功能组都有活动时间功能。该功能可以具有字符串或分数的要素类型。接受 ISO-8601 格式的字符串事件时间，以 UTC 时间表示，符合模式：[yyyy-mm-dd'h: mm: ssz, yyy-mm-dd'hh: yyy-mm-dd'hh: mm: ss.sssz]。小数事件时间值被接受为从 unix epoch 开始的秒数，精度为毫秒。活动时间必须在 [0000-01-01T00:00:00 .000Z、9999-12-31T23:59:59 .999Z] 的范围内。

Amazon SageMaker 功能商店离线商店数据格式

Amazon SageMaker 功能商店离线存储数据存储在账户中的 Amazon S3 存储桶中。当你打电话`PutRecord`，您的数据将在 15 分钟内进行缓冲、批处理并写入 Amazon S3。功能商店仅支持 Pucia 文件格式。具体来说，当您的数据写入离线商店时，只能从 Amazon S3 存储桶中以 Pucia 格式从 Amazon S3 存储桶中检索数据。每个文件可以包含多个`Records`.

按以下命名约定组织文件：

```
s3://<bucket-name>/<customer-prefix>/<account-id>/sagemaker/<aws-region>/offline-store/
<feature-group-name>-<feature-group-creation-time>/data/year=<event-time-year>/
    month=<event-time-month>/day=<event-time-day>/hour=<event-time-hour>/
<timestamp_of_latest_event_time_in_file>_<16-random-alphanumeric-digits>.parquet
```

`Records`在离线商店中，按事件时间划分为每小时的分区，如上例所示。分区方案是不可配置的。下面的代码显示了 Puca 文件的示例：

```
s3://my-bucket/my-prefix/123456789012/sagemaker/us-east-1/offline-store/
      customer-purchase-history-patterns-1593511200/data/year=2020/month=06/day=31/
hour=00/20200631T064401Z_108934320012Az11.parquet
```

功能商店还公开了离线商店配置 .s3 存储配置。已解决输出 s3URI字段，可以在描述 FureGroupAPI 调用。这是写入特定功能组文件的 S3 路径。

示例值`ResolvedOutputS3Uri`：

```
s3://my-bucket/my-prefix/123456789012/sagemaker/us-east-1/offline-store/customer-purchase-history-patterns-1593511200/data
```

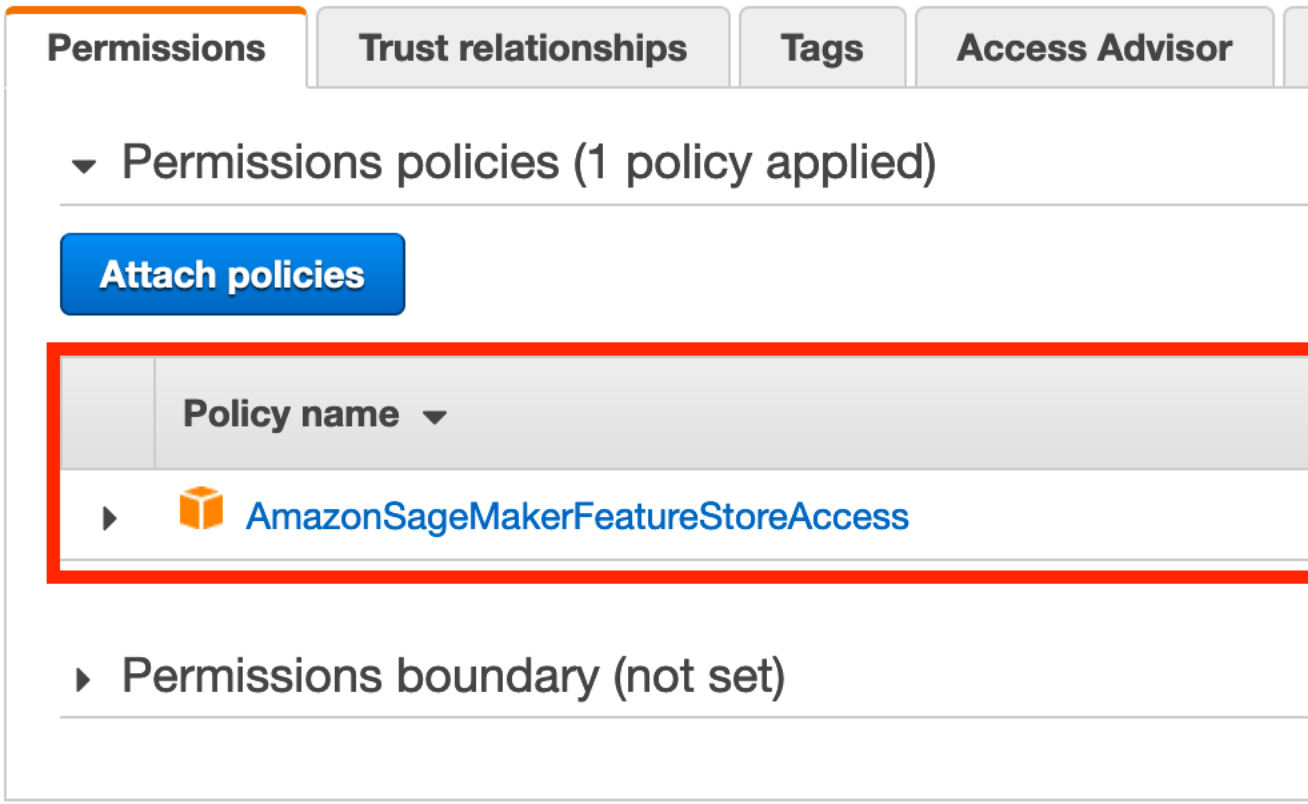
在离线商店中保留以下附加字段时，要素商店将其添加到每条记录中：

- api_调用 _ 时间— 服务收到的时间戳`PutRecord`要么`DeleteRecord`调用。如果使用托管摄取（例如 Data Wrangler），则这是将数据写入离线存储的时间戳。
- write e_time— 将数据写入离线商店的时间戳。可用于构建时间旅行相关的查询。
- is_已删除–`False`默认情况下。如果`DeleteRecord`被称为，一个新的`Record`被插入`RecordIdentifierValue`然后设置为`True`在离线商店里。

Amazon SageMaker Feature Store 笔记

要开始使用 Amazon SageMaker 功能商店，您可以从下表中的各种示例 Jupyter 笔记本中进行选择。如果这是你第一次使用功能商店，请尝试功能商店简介笔记本。要运行任何这些笔记本，您必须将此策略附加到您的 IAM 执行角色：`AmazonSageMakerFeatureStoreAccess`.

请参阅IAM 角色访问你的角色并附加此策略。有关演练，请参阅向 IAM 角色添加策略. 以下屏幕截图说明了附加策略后如何显示在您的 IAM 角色下。



Feature Store 样本

下表概述了解决 Amazon SageMaker 功能商店不同使用案例的各种样本笔记本。

笔记本标题	说明
Feature Store 简介	介绍关键要素存储功能，例如如何创建、配置功能组以及如何将数据提取到在线或离线 feature store。
使用功能商店检测欺诈	一个高级示例，介绍如何通过将数据输入要素商店、查询数据以形成训练数据集以及如何训练一个简单的推理模型来训练欺诈检测模型。
使用 KMS 密钥加密在线或离线功能商店中的数据	关于如何使用 KMS 密钥加密和解密在线或离线功能商店中的数据以及如何验证数据是否已加密的高级示例。请注意，这款笔记本处理静态加密。
使用 Feature Store 进行客户端加密Amazon加密 SDK	一个高级示例如何使用功能商店进行客户端加密Amazon加密 SDK 库在将您的数据提取到在线或离线功能商店之前对其进行加密。
如何使用 KMS 密钥安全地将影像数据集存储在要素存储中？	一个高级示例，演示如何使用 KMS 密钥进行服务器端加密将图像数据集安全存储到要素存储中。
创建从 Ground Truth 分类标签作业到功能商店的机器学习工作流	一种机器学习 (ML) 工作流程，演示如何提供图像或文本分类标签作业的输出Amazon要素商店的 Ground Truth。

训练模型

有关使用 Amazon SageMaker 训练模型的概述，请参阅使用 Amazon SageMaker 训练模型 (p. 7).

SageMaker 提供了功能用于监控和管理机器学习模型的训练和验证。有关可用指标、增量训练、自动模型优化以及使用增强清单文件来标记训练数据的指南，请参阅以下主题。

- 有关选择机器学习算法及其针对任务或问题的实施的指导，请参阅选择算法 (p. 783).
- 有关调试和分析机器学习模型训练的指导，请参阅亚马逊SageMakerDebugger (调试程序) (p. 1783).
- 有关分布式深度学习模型训练的指导，请参阅Amazon SageMaker 分布式培训库 (p. 1964).
- 有关用于监控和训练模型的指标，请参阅使用 Amazon CloudWatch 指标监控和分析训练作业 (p. 2093)。
- 有关用于检测模型后处理偏差的指标，请参阅使用亚马逊检测训练后数据和模型偏差 SageMaker 澄清 (p. 2058).
- 有关模型可解释性的指导，请参阅亚马逊 SageMaker 澄清模型的可解释性 (p. 2072).
- 有关 SageMaker 中增量训练的指南，请参阅。Amazon SageMaker 中的增量训练 (p. 2081).
- 有关在 SageMaker 中使用托管的现场训练的指导，请参阅Amazon SageMaker 中的托管的 Spot (p. 2085).
- 有关在 SageMaker 中使用训练检查点的指导，请参阅在 Amazon SageMaker 中使用检查点 (p. 2087).
- 有关自动模型优化（也称为超参数优化）的指南，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。
- 有关为标签训练数据使用增强清单文件的指南，请参阅通过增强清单文件，将数据集元数据提供给训练作业 (p. 2090)。

主题

- 选择算法 (p. 783)
- 使用亚马逊管理 Machine Learning SageMaker 实验 (p. 1758)
- 亚马逊SageMakerDebugger (调试程序) (p. 1783)
- 使用 SageMaker 执行自动模型优化 (p. 1939)
- Amazon SageMaker 分布式培训库 (p. 1964)
- 亚马逊 SageMaker 澄清偏差检测和模型可解释性 (p. 2045)
- Amazon SageMaker 中的增量训练 (p. 2081)
- 访问训练数据 (p. 2085)
- Amazon SageMaker 中的托管的 Spot (p. 2085)
- 在 Amazon SageMaker 中使用检查点 (p. 2087)
- 通过增强清单文件，将数据集元数据提供给训练作业 (p. 2090)
- 使用 Amazon CloudWatch 指标监控和分析训练作业 (p. 2093)

选择算法

机器学习可以帮助你完成需要某种感应性推断的经验任务。这项任务涉及上岗，因为它使用数据来训练算法来做出可概括的推论。这意味着，算法可以在统计上做出可靠的预测或决策，或者在应用于未用于训练它们的新数据时完成其他任务。

为了帮助您为任务选择最佳算法，我们将这些任务按不同的抽象级别对这些任务进行分类。在抽象的最高级别，机器学习尝试查找要素或结构性较低的项目（例如数据集中的文本）之间的模式或关系。模式识别技术可以分为不同的机器学习范式，每种模式都解决特定的问题类型。目前，机器学习有三种基本范例用于解决各种问题类型：

- 监督学习 (p. 786)
- 无监督的学习 (p. 786)
- 强化学习 (p. 787)

每种学习范式可以解决的问题类型是通过考虑你想从你拥有或可以收集的数据类型中做出的推断（或预测、决策或其他任务）来确定的。机器学习范例使用算法方法来解决各种问题类型。这些算法提供了解决这些问题的方法。

但是，许多算法，例如神经网络，可以用不同的学习范式和不同类型的问题来部署。多种算法也可以解决特定的问题类型。一些算法更普遍适用，而另一些算法则对于某些类型的目标和数据非常具体。因此，机器学习算法和问题类型之间的映射是多对多的。此外，还有各种实现选项可用于算法。

以下各节提供了有关实施选项、机器学习范例和适用于不同问题类型的算法的指导。

主题

- 选择算法 (p. 784)
- 基本机器学习范例的问题类型 (p. 786)
- 使用 Amazon SageMaker 内置算法或预训练模型 (p. 787)
- Amazon SageMaker 将强化学习结合使用 (p. 1757)

选择算法

选择算法后，必须决定要使用哪种实现算法。亚马逊 SageMaker 支持三种需要加大努力的实施方案。

- 预训练模型所需的努力最少，模型是否准备好部署或者微调和部署使用 SageMaker JumpStart。
- 内置算法如果数据集很大且需要大量资源来训练和部署模型，则需要更多的努力和规模。
- 如果没有可行的内置解决方案，请尝试开发一个使用用于机器和深度学习框架的预制图像适用于受支持的框架，例如 Scikit-Learn、TensorFlow、PyTorch、MXNet 或 Chainer。
- 如果你需要运行自定义软件包或使用任何不属于受支持框架的一部分或可通过 PyPi 获得的代码，那么你需要构建你自己的自定义 Docker 镜像配置为安装必要的软件包或软件。还必须将自定义映像推送到亚马逊弹性容器注册表等在线存储库。

主题

- 使用内置算法 (p. 785)
- 在受支持的框架中使用脚本模式 (p. 785)
- 使用自定义 Docker 镜像 (p. 785)

指导算法

实现	需要代码	预编码算法	Support 第三方套餐	Support 自定义代码	努力水平
内置	否	是	否	否	低
Scikit-learn	是	是	仅限 PyPi	是	中
Spark ML	是	是	仅限 PyPi	是	中
xgBoost（开源）	是	是	仅限 PyPi	是	中

实现	需要代码	预编码算法	Support 第三方套餐	Support 自定义代码	努力水平
TensorFlow	是	否	仅限 PyPi	是	中高
PyTorch	是	否	仅限 PyPi	是	中高
MXNet	是	否	仅限 PyPi	是	中高
Chainer	是	否	仅限 PyPi	是	中高
自定义映像	是	否	是，从任何源	是	高

使用内置算法

在为问题和数据类型选择算法时，最简单的选择是使用 Amazon SageMaker 的内置算法之一。这些内置算法有两个主要好处。

- 内置算法无需编码即可开始运行实验。您唯一需要提供的输入是数据、超参数和计算资源。这使您可以更快地运行实验，同时减少跟踪结果和代码更改的开销。
- 内置算法具有跨多个计算实例的并行化功能，对于所有适用算法都可以立即开箱即用的 GPU 支持（由于固有的限制，某些算法可能不包括在内）。如果你有大量数据来训练模型，那么大多数内置算法都可以轻松扩展以满足需求。即使你已经有了预训练的模型，也可能更容易使用它的必然结果 SageMaker 并在受支持的框架上使用脚本模式输入你已经知道的超参数，而不是移植它。

有关 SageMaker 提供的内置算法的更多信息，请参阅使用 Amazon SageMaker 内置算法或预训练模型 (p. 787).

有关 docker 注册表路径、数据格式、推荐的 EC2 实例类型的重要信息 CloudWatch SageMaker 提供的所有内置算法的常见日志，请参阅有关内置算法的常见信息 (p. 791).

在受支持的框架中使用脚本模式

如果您想用于模型的算法不受内置选择的支持，并且您喜欢对自己的解决方案进行编码，那么您应考虑使用 Amazon SageMaker 支持的框架。这被称为 “脚本模式”，因为您在文本文件中写入自定义代码（脚本）时使用 `.py`。扩展名。如上表所示， SageMaker 支持大多数流行的机器学习框架。这些框架预装了相应的框架和一些额外的 Python 软件包，例如 Pandas 和 NumPy，因此你可以编写自己的代码来训练算法。这些框架还允许你安装托管在上的任何 Python 软件包 PyPi 通过在训练代码中包含 requirements.txt 文件或包含自己的代码目录。R 在本地也受到支持 SageMaker 笔记本内核。一些框架，例如 cikit-learn 和 Spark ML，具有可以轻松使用的预编码算法，而其他框架如 TensorFlow 而 PyTorch 可能需要你自己实施算法。使用受支持的框架映像时唯一的限制是不能导入任何未托管在上的软件包 PyPi 或者框架的图像中尚未包含的那些内容。

有关 SageMaker 支持的框架的更多信息，请参阅将 Machine Learning 框架、Python 和 R 与 Amazon SageMaker 结合使用 (p. 11).

使用自定义 Docker 镜像

Amazon SageMaker 的内置算法和支持的框架应涵盖大多数使用案例，但有时您可能需要使用不包含在任何受支持框架中的软件包中的算法。您可能还会在需要部署的某个地方选择或保留预训练的模型。 SageMaker 使用 Docker 镜像托管所有模型的培训和服务，因此，如果所需的软件包或软件未包含在受支持的框架中，您可以提供自己的自定义 Docker 镜像。这可能是你自己的 Python 软件包，也可能是用 Stan 或 Julia 这样的语言编码的算法。对于这些图像，您还必须在 Dockerfile 中正确配置算法的训练和模型的服务。这需要对 Docker 的中级知识，除非你喜欢编写自己的机器学习算法，否则不推荐这样做。您的 Docker 映像必须先上传到在线存储库，例如 Amazon Elastic 容器注册表 (ECR)，然后才能正确训练和提供模型。

有关 SageMaker 中自定义 Docker 映像的更多信息，请参阅将 Docker 容器用于 SageMaker (p. 2400).

基本机器学习范例的问题类型

以下三节介绍了机器学习的三个基本范例所解决的主要问题类型。有关内置算法的列表 SageMaker 提供了解决这些问题类型，请参阅使用 Amazon SageMaker 内置算法或预训练模型 (p. 787).

主题

- 监督学习 (p. 786)
- 无监督的学习 (p. 786)
- 强化学习 (p. 787)

监督学习

如果您的数据集包含包含目标值（输出）的要素或属性（输入），则会出现监督学习问题。如果你的目标值是分类的（数学上离散的），那么你有分类问题. 区分二进制分类和多类分类是一种标准做法。

- 二进制分类是一种受监督的学习类型，可根据个体的属性将个体分配给两个预定义且互斥的类别之一。它受到监督，因为模型是使用示例进行训练的，其中为属性提供了正确标记的对象。基于诊断测试的结果对个人是否患有疾病的医学诊断是二元分类的一个示例。
- 多类别分类是一种受监督的学习类型，可根据个体的属性将个体分配给多个类别之一。它受到监督，因为模型是使用示例进行训练的，其中为属性提供了正确标记的对象。一个例子是预测与文本文档最相关的主题。文档可以分类为宗教、政治或金融，或者是其他若干个预定义的主题类别之一。

如果你试图预测的目标值在数学上是连续的，那么你有一个回归问题。回归根据一个或多个与其相关的其他变量或属性来估计因果目标变量的值。一个例子是预测房屋价格所使用的特征，如浴室和卧室的数量以及房屋和花园的平方英尺数。回归分析可以创建一个模型，该模型将其中一个或多个特征作为输入并预测房屋价格。

有关 SageMaker 提供的内置监督学习算法的更多信息，请参阅监督学习 (p. 790).

无监督的学习

如果您的数据集包含不包含标签或目标值（输出）的要素或属性（输入），则会出现无监督学习问题。在这种类型的问题中，必须根据输入数据中发现的模式来预测输出。无监督学习问题的目标是发现数据中的分组等模式。无监督学习可以应用于各种各样的任务或问题类型。主成分和聚类分析是常用于预处理数据的两种主要方法。以下是可以通过无监督学习解决的问题类型的简短列表：

- 维度缩减通常是用于确定用于模型构建的最相关特征的数据探索步骤的一部分。我们的想法是将数据从高维、人口稀少的空间转换为保留原始数据最重要属性的低维空间。这为减轻了人口稀少、高维数据可能产生的维度诅咒，统计分析成为问题。它还可以用来帮助理解数据，将高维数据减少到可视化的较低维度。
- 集群分析是一类用于将对象或案例分类为称为群集的组的技术。它尝试在数据中寻找离散组，其中一个组的成员尽可能彼此相似，而与其他组的成员尽可能互不相同。您可以定义希望算法用于确定相似性的要素或属性，选择一个距离函数来测量相似性，然后指定要在分析中使用的聚类数。
- 异常检测是识别数据集中的稀有物品、事件或观测值，这些物品、事件或观测值因为它们与其余数据有很大差异而引起怀疑。例如，可以使用识别异常物品来检测银行欺诈或医疗错误。异常也称为异常值、新奇、噪音、偏差和异常。
- 估算密度是根据观测到的数据来构建无法观察到的潜在概率密度函数的估计值。密度估计值的自然用途是进行数据探索。密度估计可以发现数据中的偏度和多模性等特征。密度估计的最基本形式是重新缩放的直方图。

SageMaker 提供了多种内置的机器学习算法，可供您用于这些无监督的学习任务。有关 SageMaker 提供的内置无监督算法的详细信息，请参阅无监督学习 (p. 790).

强化学习

强化学习是一种基于与环境互动的学习类型。这种学习被必须通过学习行为的代理人使用 trial-and-error 与动态环境的互动，其目标是代理通过其操作获得最大化的长期奖励。通过利用已知奖励的行为来取消探索有不确定奖励的行为，从而最大限度地提高奖励。

有关 SageMaker 的框架、工具包和强化学习环境的更多信息，请参阅Amazon SageMaker 将强化学习结合使用 (p. 1757).

使用 Amazon SageMaker 内置算法或预训练模型

亚马逊 SageMaker 提供了一套内置算法、预训练模型和预构建的解决方案模板，帮助数据科学家和机器学习从业人员快速开始培训和部署机器学习模型。对于 SageMaker 不熟悉的人来说，为您的特定用例选择正确的算法可能是一项具有挑战性的任务。下表提供了一份快速备忘单，其中显示了如何从示例问题或使用案例开始，以及如何找到适当的内置算法 SageMaker 这对于该问题类型有效。下表的章节提供了按学习范式（受监督和无监督）和重要数据域（文本和图像）组织的其他指导。

表：将用例映射到内置算法

示例问题和使用案例	学习范式或领域	问题类型	数据输入格式	内置算法
以下是 15 种问题类型中的几个示例，这些问题类型可以通过提供的预训练模型和预构建的解决方案模板来解决。 SageMaker JumpStart： 问题回答：输出给定问题答案的聊天机器人。 文本分析：分析特定于财务等行业领域的模型中的文本。	预训练模型和预构建的解决方案模板	图像分类 表格分类 表格回归 文本分类 对象检测 嵌入文本 解答问题 句子对分类 映像嵌入 命名实体识别 实例分段 文本生成 文本总结 语义分割 机器翻译	图片、文本、表格	热门机型，包括 Mobilenet、YOLO、更快的 R-CNN、BERT、LightGBM 和 catBoost 有关可用的预训练模型的列表，请参阅JumpStart 模型. 有关可用的预构建解决方案模板的列表，请参阅JumpStart Solutions.
预测商品是否属于某个类别：电子邮件垃圾邮件过滤器	监督学习 (p. 790)	二进制/多类别分类	表格	因子分解机算法 (p. 1611), K 最近邻 (k-NN) 算法 (p. 1644), 线性学习器算法 (p. 1657),

示例问题和使用案例	学习范式或领域	问题类型	数据输入格式	内置算法
				XGBoost 算法 (p. 1731)
预测数字/连续值：估计房屋的价值		回归	表格	因子分解机算法 (p. 1611), K 最近邻 (k-NN) 算法 (p. 1644), 线性学习器算法 (p. 1657), XGBoost 算法 (p. 1731)
根据行为的历史数据，预测 future 行为：根据以前的销售数据预测新产品的销售额。		时间序列预测	表格	DeepAR 预测算法 (p. 1598)
改进高维对象的数据嵌入：识别重复的支持票证或根据票证中文本的相似性找到正确的路线		嵌入：将高维物体转换为低维空间。	表格	Object2Vec 算法 (p. 1677)
从与标签/目标变量关系薄弱的数据集中删除那些列：预测其里程时汽车的颜色。	无监督学习 (p. 790)	功能工程：减少维度	表格	主成分分析 (PCA) 算法 (p. 1700)
检测应用中的异常行为：发现 IoT 传感器何时发送异常读数		异常检测	表格	随机森林砍伐 (RCF) 算法 (p. 1704)
保护您的应用程序免受可疑用户的侵害：检测访问服务的 IP 地址是否来自坏参与者		IP 异常检测	表格	IP 见解 (p. 1630)
将类似的对象/数据组合在一起：从交易历史中找到高、中和低支出客户		群集或分组	表格	K-Means 算法 (p. 1637)
将一组文档组织成主题（事先未知）：根据文档中使用的术语将文档标记为属于医疗类别。		主题建模	文本	潜在狄利克雷分配 (LDA) 算法 (p. 1652), 神经主题模型 (NTM) 算法 (p. 1671)

示例问题和使用案例	学习范式或领域	问题类型	数据输入格式	内置算法
为语料库中的文档分配预定义的类别：将图书馆中的图书分类为学术学科	文本分析 (p. 790)	文本分类	文本	BlazingText 算法 (p. 1589)
将文本从一种语言转换为其他语言：西班牙语到英语		机器翻译算法	文本	序列到序列算法 (p. 1720)
总结一篇长文本语料库：研究 paper 摘要		文本总结	文本	序列到序列算法 (p. 1720)
将音频文件转换为文本：转录呼叫中心对话以进一步分析		语音到文本转换	文本	序列到序列算法 (p. 1720)
根据图片的内容为图片添加标签/标记：关于图片中成人内容的警报	图像处理 (p. 790)	图片和多标签分类	映像	图像分类算法 (p. 1620)
检测图像中的人和物体：警方查看失踪人员的大型照片库		对象检测和分类	映像	对象检测算法 (p. 1691)
使用类别单独标记图像的每个像素：自动驾驶汽车准备识别其方式的物体		计算机视觉	映像	语义分割算法 (p. 1711)

有关 Docker 注册表路径、数据格式、重新推出的 Amazon EC2 实例类型以及 CloudWatch SageMaker 提供的所有内置算法的常见日志，请参阅有关内置算法的常见信息 (p. 791).

以下各节为亚马逊提供了更多指导。 SageMaker 内置算法按它们所属的监督和无监督学习范式分组。有关这些学习范式及其相关问题类型的描述，请参阅选择算法 (p. 783). 还提供了部分用于 SageMaker 内置算法可用于解决两个重要的机器学习领域：文本分析和图像处理。

- 预训练的模型和解决方案模板 (p. 789)
- 监督学习 (p. 790)
- 无监督学习 (p. 790)
- 文本分析 (p. 790)
- 图像处理 (p. 790)

预训练的模型和解决方案模板

SageMaker JumpStart 提供了广泛的预训练模型、预构建的解决方案模板以及使用 SageMaker SDK 以及工作室。有关这些模型、解决方案和示例笔记本的更多信息 SageMaker JumpStart，请参阅SageMaker JumpStart (p. 38).

监督学习

亚马逊 SageMaker 提供了几种内置的通用算法，可用于分类或回归问题。

- 线性学习器算法 (p. 1657)— 学习用于回归的线性函数或为分类学习线性阈值函数。
- 因子分解机算法 (p. 1611)— 线性模型的扩展，旨在经济地捕获高维稀疏数据集中的各特征之间的交互。
- XGBoost 算法 (p. 1731)— 梯度提升树算法的实现，它将一组较简单且较弱的模型的一系列估计值结合在一起。
- K 最近邻 (k-NN) 算法 (p. 1644)— 一种非参数化方法，该方法使用 k 个最近标注点将标注分配给新的数据点进行分类，或者使用用于回归的 k 个最近点的平均值分配一个预测目标值。

亚马逊 SageMaker 还提供了几种内置的监督学习算法，用于在特征工程和根据时间序列数据进行预测期间执行更专业的任务

- Object2Vec 算法 (p. 1677)— 用于特征工程的新型高度可定制的多用途算法。它可以学习高维对象的低维密集嵌入，从而提高下游模型的训练效率的特性。虽然这是一种监督算法，因为它需要标记数据进行训练，但在许多情况下，在许多情况下，可以纯粹从数据中的自然聚类中获取关系标签，而无需任何明确的人类注释。
- DeepAR 预测算法 (p. 1598)一种指导式学习算法，用于使用递归神经网络 (RNN) 来预测标量（一维）时间序列。

无监督学习

亚马逊 SageMaker 提供了几种内置算法，可用于各种无监督学习任务，例如聚类、维度缩减、模式识别和异常检测。

- 主成分分析 (PCA) 算法 (p. 1700)— 通过将数据点投影到前几个主组件，从而减少数据集中的维数（要素数）。目的是尽可能保留尽可能多的信息或变体。对于数学家来说，主要成分是数据协方差矩阵的特征向量。
- K-Means 算法 (p. 1637)— 在数据中寻找离散组，其中一个组的成员尽可能彼此相似，而与其他组的成员尽可能互不相同。
- IP 见解 (p. 1630)— 了解 IPv4 地址的使用模式。它旨在捕获 IPv4 地址与各种实体（例如用户 ID 或帐号）之间的关联。
- 随机森林砍伐 (RCF) 算法 (p. 1704)— 检测数据集中与良好结构或模式化数据存在偏差的异常数据点。

文本分析

SageMaker 提供的算法专为自然语言处理、文档分类或总结、主题建模或分类以及语言转录或翻译中使用的文本文档进行分析而定制。

- BlazingText 算法 (p. 1589)— 高度优化的 Word2vec 和文本分类算法的实现，可轻松扩展到大型数据集。它对于许多下游自然语言处理 (NLP) 任务是有用的。
- 序列到序列算法 (p. 1720)通常用于神经网络机器翻译的监督算法。
- 潜在狄利克雷分配 (LDA) 算法 (p. 1652)— 一种适用于确定一组文档中的主题的算法。它是一种自主算法，这意味着在训练期间不适用包含答案的示例数据。
- 神经主题模型 (NTM) 算法 (p. 1671)— 另一种自主技术，它通过一种神经网络方法来确定一组文档中的主题。

图像处理

SageMaker 还提供了用于图像分类、对象检测和计算机视觉的图像处理算法。

- 图像分类算法 (p. 1620)将示例数据与答案结合使用（称为监督算法)。 使用此算法为图像分类。
- 语义分割算法 (p. 1711)— 提供了一种细粒度的像素级方法来开发计算机视觉应用程序。
- 对象检测算法 (p. 1691)— 使用单个深度神经网络检测和分类图像中的对象。它是一种指导式学习算法，将图像作为输入并识别图像场景中的所有对象实例。

主题

- 有关内置算法的常见信息 (p. 791)
- BlazingText 算法 (p. 1589)
- DeepAR 预测算法 (p. 1598)
- 因子分解机算法 (p. 1611)
- 图像分类算法 (p. 1620)
- IP 见解 (p. 1630)
- K-Means 算法 (p. 1637)
- K 最近邻 (k-NN) 算法 (p. 1644)
- 潜在狄利克雷分配 (LDA) 算法 (p. 1652)
- 线性学习器算法 (p. 1657)
- 神经主题模型 (NTM) 算法 (p. 1671)
- Object2Vec 算法 (p. 1677)
- 对象检测算法 (p. 1691)
- 主成分分析 (PCA) 算法 (p. 1700)
- 随机森林砍伐 (RCF) 算法 (p. 1704)
- 语义分割算法 (p. 1711)
- 序列到序列算法 (p. 1720)
- XGBoost 算法 (p. 1731)

有关内置算法的常见信息

下表列出了 Amazon SageMaker 提供的每种算法的参数。

算法名称	渠道名称	训练输入模式	文件类型	实例类	可并行化	
BlazingText	训练	文件或管道	文本文件（每行一句，带空格分隔的标记）	GPU（仅单个实例）或 CPU	否	
DeepAR 预测	训练和 (可选) 测试	文件	JSON 行或 Parquet	GPU 或 CPU	是	
因子分解机	训练和 (可选) 测试	文件或管道	recordIO-protobuf	CPU（对密集数据使用 GPU）	是	
图像分类	训练和验证，（可选）train_lst、validation_lst 和模型	文件或管道	recordIO 或图像文件 (.jpg 或 .png)	GPU	是	

算法名称	渠道名称	训练输入模式	文件类型	实例类	可并行化	
IP 见解	训练和 (可选) 验证	文件	CSV	CPU 或 GPU	是	
k-means	训练和 (可选) 测试	文件或管道	recordIO-protobuf 或 CSV	CPU 或 GPUCommon（一个或多个实例上的单个 GPU 设备）	否	
k-nearest-neighbor (k-NN)	训练和 (可选) 测试	文件或管道	recordIO-protobuf 或 CSV	CPU 或 GPU（一个或多个实例上的单个 GPU 设备）	是	
LDA	训练和 (可选) 测试	文件或管道	recordIO-protobuf 或 CSV	CPU（仅单个实例）	否	
线性学习器	训练和 (可选) 验证和/或测试	文件或管道	recordIO-protobuf 或 CSV	CPU 或 GPU	是	
神经主题模型	训练和 (可选) 验证和/或测试	文件或管道	recordIO-protobuf 或 CSV	GPU 或 CPU	是	
Object2Vec	训练和 (可选) 验证和/或测试	文件	JSON 行	GPU 或 CPU（仅单个实例）	否	
对象检测	训练和验证，（可选）train_annotation、validation_annotation 和模型	文件或管道	recordIO 或图像文件（.jpg 或 .png）	GPU	是	
PCA	训练和 (可选) 测试	文件或管道	recordIO-protobuf 或 CSV	GPU 或 CPU	是	
随机森林砍伐	训练和 (可选) 测试	文件或管道	recordIO-protobuf 或 CSV	CPU	是	
语义分割	训练和验证、train_annotation、validation_annotation 以及（可选）label_map 和模型	文件或管道	图像文件	GPU（仅单个实例）	否	
Seq2Seq 建模	训练、验证和 vocab	文件	recordIO-protobuf	GPU（仅单个实例）	否	
XGBoost（0.90-1、0.90-2、1.0-1、1.2-1）	训练和 (可选) 验证	文件或管道	CSV、LibSVM 或实木复合地板	CPU（或者用于 1.2-1 的 GPU）	是	

可并行化 的算法可部署在多个计算实例上以进行分布式训练。

以下主题提供有关 Docker 注册表路径、数据格式、推荐的 Amazon EC2 实例类型的信息以及 CloudWatch Amazon SageMaker 提供的所有内置算法的常见日志。

主题

- Docker 注册表路径和示例代码 (p. 793)
- 内置算法的常见数据格式 (p. 1579)
- 内置算法的实例类型 (p. 1587)
- 内置算法的日志 (p. 1588)

Docker 注册表路径和示例代码

以下主题列出了每个亚马逊的 Docker 注册表路径和其他参数 SageMaker 提供的算法和 Deep Learning Containers (DLC)。

按如下方式使用路径：

- 要创建训练作业 (创建 _训练_job) 中，指定 Docker 注册表路径 (`TrainingImage`) 和训练输入模式 (`TrainingInputMode`) 用于训练图片。您可以创建训练作业来通过特定数据集训练模型。
- 要创建模型 (create_model) 中，指定 Docker 注册表路径 (`Image`) 对于推理图像 (`PrimaryContainer Image`)。 SageMaker 启动基于终端节点配置的机器学习计算实例并部署包含构件 (模型训练的结果) 的模型。

> **Note**
>
> 对于注册表路径，请使用`:1`版本标签，以确保您使用的是算法 /DLC 的稳定版本。您可以通过在具有 `:1` 标签的推理镜像上使用具有 `:1` 标签的镜像来可靠地托管已训练的模型。使用`:latest`注册表路径中的标签为您提供了最多 up-to-date 算法 /DLC 版本，但可能会导致向后兼容性问题。避免将 `:latest` 标签用于生产用途。
> 对于 xgBoost，请勿使用`:latest`要么`:1`. 使用所需的特定版本，例如`:0.90-1-cpu-py3`、`:0.90-2-cpu-py3`、`:1.0-1-cpu-py3`，或者`:1.2-1`.

要查找注册表路径，请选择Amazon区域，然后选择算法或 DLC。

主题

- 美国东部（俄亥俄）(us-east-2) 的 Docker 注册表路径和示例代码 (p. 794)
- 美国东部（弗吉尼亚北部）Docker 注册表路径和示例代码 (us-east-1) (p. 827)
- 美国西部（加利福尼亚北部）的 Docker 注册表路径和示例代码 (us-west-1) (p. 863)
- 美国西部（俄勒冈）(us-west-2) Docker 注册表路径和示例代码 (p. 897)
- 非洲 Docker 注册表路径和示例代码（开普敦）(af-south-1) (p. 930)
- 亚太地区注册管理机构路径和示例代码（香港）(ap-east-1) (p. 962)
- 亚太地区 Docker 注册表路径和示例代码（孟买）(ap-south-1) (p. 994)
- 亚太地区（大阪）Docker 注册表路径和示例代码 (ap-northeast-3) (p. 1028)
- 亚太地区 Docker 注册管理机构路径和示例代码 (ap-northeast-2) (p. 1052)
- 亚太地区 Docker 注册管理机构路径和示例代码（新加坡）(ap-southeast-1) (p. 1085)
- 亚太地区 Docker 注册表路径和示例代码（悉尼）(ap-southeast-2) (p. 1119)
- 亚太地区 Docker 注册表路径和示例代码 (ap-nornor-1) (p. 1152)
- 加拿大 Docker 注册表路径和示例代码 (ca-central-1) (p. 1186)
- 中国（北京）的 Docker 注册表路径和示例代码 (cn-north-1) (p. 1219)
- 中国（宁夏）的 Docker 注册表路径和示例代码 (cn-northwest-1) (p. 1251)
- 欧洲 Docker 注册表路径和示例代码（法兰克福）(eu-central-1) (p. 1282)

- 欧洲 Docker 注册表路径和示例代码（爱尔兰）(eu-west-1) (p. 1316)
- 欧洲 Docker 注册表路径和示例代码（伦敦）(eu-west-2) (p. 1350)
- 欧洲 Docker 注册表路径和示例代码（巴黎）(eu-west-3) (p. 1383)
- 欧洲 Docker 注册表路径和示例代码（斯德哥尔摩）(eu-north-1) (p. 1415)
- 欧洲 Docker 注册表路径和示例代码（米兰）(eu-south-1) (p. 1447)
- 中东（巴林）Docker 注册表路径和示例代码 (me-south-1) (p. 1479)
- 南美洲（圣保罗）Docker 注册表路径和示例代码 (sa-east-1) (p. 1511)
- Docker 注册表路径和示例代码Amazon GovCloud (美国西部) (us-gov-west-1) (p. 1543)

美国东部（俄亥俄）(us-east-2) 的 Docker 注册表路径和示例代码

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 795)
- BlazingText（算法）(p. 795)
- Chainer（DLC）(p. 796)
- 澄清（算法）(p. 796)
- 数据 Wrangler（算法）(p. 796)
- 调试器（算法）(p. 797)
- DeepAR 预测（算法）(p. 797)
- 分解机器（算法）(p. 797)
- 拥抱脸（算法）(p. 798)
- 知识产权见解（算法）(p. 800)
- 图像分类（算法）(p. 801)
- 推理 mXNet (DLC) (p. 801)
- 推理 PyTorch (DLC) (p. 801)
- K-Mean（算法）(p. 802)
- KNN（算法）(p. 802)
- LDA（算法）(p. 802)
- 线性学习者（算法）(p. 803)
- MxNet (DLC) (p. 803)
- MxNet 教练（DLC）(p. 806)
- 模型监视器（算法）(p. 806)
- NTM（算法）(p. 806)
- Neo 图像分类（算法）(p. 807)
- Neo MxNet (DLC) (p. 807)
- 新 PyTorch (DLC) (p. 807)
- Neo TensorFlow (DLC) (p. 808)
- Neo xgBoost（算法）(p. 808)
- 对象检测（算法）(p. 809)
- Object2VEC（算法）(p. 809)
- PCA（算法）(p. 809)
- pyTorch (DLC) (p. 810)
- 随机森林砍伐 (算法) (p. 813)
- 射线 PyTorch (DLC) (p. 813)
- Scikit-learn (算法) (p. 814)

- 语义分割（算法）(p. 814)
- Sq2Seq (算法) (p. 814)
- Spark（算法）(p. 815)
- SparkML 服务（算法）(p. 815)
- Tensorflow (DLC) (p. 815)
- Tensorflow 教练（DLC）(p. 824)
- 张量流推断 (DLC) (p. 825)
- Tensorflow Ray (DLC) (p. 825)
- 大众（算法）(p. 826)
- xgBoost（算法）(p. 826)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='us-
east-2',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-east-2.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
825641698319.dkr.ecr.us-east-2.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='us-
east-2',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='us-
east-2',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
211330385671.dkr.ecr.us-east-2.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
415577184552.dkr.ecr.us-east-2.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
915447279597.dkr.ecr.us-east-2.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
566113047672.dkr.ecr.us-east-2.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
404615174143.dkr.ecr.us-east-2.amazonaws.com/	1	训练、推理		

注册表路径	版本	Job 类型（图像范围）		
factorization-machines：<tag>				

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='us-
east-2',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-training：<tag>	4.17.0	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-training：<tag>	4.17.0	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-inference：<tag>	4.17.0	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.17.0	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/	4.11.0	培训		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-training：<tag>				
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-training：<tag>	4.11.0	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.us-east-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
404615174143.dkr.ecr.us-east-2.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
825641698319.dkr.ecr.us-east-2.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='us-
east-2',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='us-
east-2',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Mean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
404615174143.dkr.ecr.us-east-2.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
404615174143.dkr.ecr.us-east-2.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
999911452149.dkr.ecr.us-east-2.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
404615174143.dkr.ecr.us-east-2.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='us-
east-2',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-2.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/	1.7.0	eia	CPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：*<tag>*				
763104351884.dkr.ecr.us-east-2.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-east-2.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.us-east-2.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12.1	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2-1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='us-
east-2',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
777275614652.dkr.ecr.us-east-2.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
404615174143.dkr.ecr.us-east-2.amazonaws.com/ntm：<tag>	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
007439368137.dkr.ecr.us-east-2.amazonaws.com/image-classification-neo：<tag>	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='us-
east-2',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-inference-mxnet：<tag>	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='us-
east-2',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-	1.8	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
inference-pytorch：*<tag>*				
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='us-
east-2',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
007439368137.dkr.ecr.us-east-2.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
825641698319.dkr.ecr.us-east-2.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
404615174143.dkr.ecr.us-east-2.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
404615174143.dkr.ecr.us-east-2.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='us-
east-2',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.11.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.11.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/	1.9.0	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
pytorch-training：*<tag>*				
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference-eia：*<tag>*	1.5.1	eia	CPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
404615174143.dkr.ecr.us-east-2.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='us-
east-2',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-火炬*<tag>*	1.6	培训	CPU，GPU	py36
462105765813.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-ray-container:	0.8	培训	CPU，GPU	py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
ray-0.8.5-火炬-*<tag>*				

Scikit-learn (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='us-
east-2',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
825641698319.dkr.ecr.us-east-2.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Sq2Seq (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='us-east-2')
```

注册表路径	版本	Job 类型（图像范围）		
825641698319.dkr.ecr.us-east-2.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='us-
east-2',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
314815235551.dkr.ecr.us-east-2.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
314815235551.dkr.ecr.us-east-2.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
314815235551.dkr.ecr.us-east-2.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='us-east-2',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='us-
east-2',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.3.0	eia	CPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.1.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.0.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.0.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.0.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.0.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：*<tag>*	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：*<tag>*	2.0.2	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference-eia：<tag>	2.0.0	eia	CPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.13.0	eia	CPU	-
763104351884.dkr.ecr.us-east-2.amazonaws.com/tensorflow-inference：<tag>	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.12.0	eia	CPU	-
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.12.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.11.0	eia	CPU	-
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow-eia：<tag>	1.10.0	eia	CPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='us-
east-2',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-coach-container: 教练 -1.0.0-tf-*<tag>*	1.0.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-	0.10.1	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.10.1-*<tag>*				
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='us-
east-2',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
007439368137.dkr.ecr.us-east-2.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='us-
east-2',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-ray-container: ray-1.6.0-TF-*<tag>*	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-	0.8.5	培训	CPU，GPU	py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
ray-container: ray-0.8.5-tf-*<tag>*				
462105765813.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-ray-container: ray-0.8.2-TF-*<tag>*	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-tensorflow: ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-tensorflow: ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-tensorflow: ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-tensorflow: ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='us-
east-2',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.us-east-2.amazonaws.com/sagemaker-rl-vw-container: vw-8.7.0-*<tag>*	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='us-east-2',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-xgboost：<tag>	1.5-1	1.5.2	训练、推理	
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-xgboost：<tag>	1.3-1	1.3.3	训练、推理	
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-xgboost：<tag>	1.2	1.2.0	训练、推理	
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-xgboost：<tag>	1.2-1	1.2.0	训练、推理	
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-xgboost：<tag>	1.0-1	1.0	训练、推理	
825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost：<tag>		1.5.2	训练、推理	
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	训练、推理	
257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-xgboost：<tag>	0.90-1	0.90	训练、推理	

美国东部（弗吉尼亚北部）Docker 注册表路径和示例代码 (us-east-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 828)
- BlazingText（算法）(p. 829)
- Chainer（DLC）(p. 829)
- 澄清（算法）(p. 830)
- 数据 Wrangler（算法）(p. 830)

- 调试器（算法）(p. 831)
- DeepAR 预测（算法）(p. 831)
- 分解机器（算法）(p. 831)
- 拥抱脸（算法）(p. 832)
- 知识产权见解（算法）(p. 835)
- 图像分类（算法）(p. 835)
- 推理 mXNet (DLC) (p. 835)
- 推理 PyTorch (DLC) (p. 836)
- K-Meean（算法）(p. 836)
- KNN（算法）(p. 837)
- LDA（算法）(p. 837)
- 线性学习者（算法）(p. 837)
- MxNet (DLC) (p. 838)
- MxNet 教练（DLC）(p. 840)
- 模型监视器（算法）(p. 841)
- NTM（算法）(p. 841)
- Neo 图像分类（算法）(p. 841)
- Neo MxNet (DLC) (p. 842)
- 新 PyTorch (DLC) (p. 842)
- Neo TensorFlow (DLC) (p. 843)
- Neo xgBoost（算法）(p. 843)
- 对象检测（算法）(p. 844)
- Object2VEC（算法）(p. 844)
- PCA（算法）(p. 844)
- pyTorch (DLC) (p. 845)
- 随机森林砍伐 (算法) (p. 848)
- 射线 PyTorch (DLC) (p. 848)
- Scikit-learn (算法) (p. 849)
- 语义分割（算法）(p. 849)
- Seq2Seq（算法）(p. 850)
- Spark（算法）(p. 850)
- SparkML 服务（算法）(p. 850)
- Tensorflow (DLC) (p. 851)
- Tensorflow 教练（DLC）(p. 859)
- 张量流推断 (DLC) (p. 860)
- Tensorflow Ray (DLC) (p. 861)
- 大众（算法）(p. 862)
- xgBoost（算法）(p. 862)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='us-
east-1',image_scope='inference',version='0.4')
```

```
# Output path
'763104351884.dkr.ecr.us-east-1.amazonaws.com/autogluon-inference:0.4-cpu-py38'
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-east-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='us-east-1')

# Output path
'811284229777.dkr.ecr.us-east-1.amazonaws.com/blazingtext:1'
```

注册表路径	版本	Job 类型（图像范围）		
811284229777.dkr.ecr.us-east-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='us-
east-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')

# Output path
'520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-chainer:5.0.0-cpu-py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='us-
east-1',version='1.0',image_scope='processing')

# Output path
'205585389593.dkr.ecr.us-east-1.amazonaws.com/sagemaker-clarify-processing:1.0'
```

注册表路径	版本	Job 类型（图像范围）		
205585389593.dkr.ecr.us-east-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='us-east-1')

# Output path
'663277389841.dkr.ecr.us-east-1.amazonaws.com/sagemaker-data-wrangler-container:1.x'
```

注册表路径	版本	Job 类型（图像范围）		
663277389841.dkr.ecr.us-east-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='us-east-1')

# Output path
'503895931360.dkr.ecr.us-east-1.amazonaws.com/sagemaker-debugger-rules:latest'
```

注册表路径	版本	Job 类型（图像范围）		
503895931360.dkr.ecr.us-east-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='us-east-1')

# Output path
'522234722520.dkr.ecr.us-east-1.amazonaws.com/forecasting-deepar:1'
```

注册表路径	版本	Job 类型（图像范围）		
522234722520.dkr.ecr.us-east-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='us-east-1')

# Output path
'382416733822.dkr.ecr.us-east-1.amazonaws.com/factorization-machines:1'
```

注册表路径	版本	Job 类型（图像范围）		
382416733822.dkr.ecr.us-east-1.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='us-
east-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')

# Output path
'763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-training:2.4.1-
transformers4.4.2-gpu-py37'
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='us-east-1')

# Output path
'382416733822.dkr.ecr.us-east-1.amazonaws.com/ipinsights:1'
```

注册表路径	版本	Job 类型（图像范围）		
382416733822.dkr.ecr.us-east-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='us-east-1')

# Output path
'811284229777.dkr.ecr.us-east-1.amazonaws.com/image-classification:1'
```

注册表路径	版本	Job 类型（图像范围）		
811284229777.dkr.ecr.us-east-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='us-
east-1',version='1.5.1',instance_type='ml.inf1.6xlarge')

# Output path
'785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-neo-mxnet:1.5.1-inf-py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='us-
east-1',version='1.9',py_version='py3')

# Output path
'785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-neo-pytorch:1.9-inf-py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='us-east-1')

# Output path
'382416733822.dkr.ecr.us-east-1.amazonaws.com/kmeans:1'
```

注册表路径	版本	Job 类型（图像范围）		
382416733822.dkr.ecr.us-east-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='us-east-1')

# Output path
'382416733822.dkr.ecr.us-east-1.amazonaws.com/knn:1'
```

注册表路径	版本	Job 类型（图像范围）		
382416733822.dkr.ecr.us-east-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='us-east-1')

# Output path
'766337827248.dkr.ecr.us-east-1.amazonaws.com/lda:1'
```

注册表路径	版本	Job 类型（图像范围）		
766337827248.dkr.ecr.us-east-1.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='us-east-1')

# Output path
'382416733822.dkr.ecr.us-east-1.amazonaws.com/linear-learner:1'
```

注册表路径	版本	Job 类型（图像范围）		
382416733822.dkr.ecr.us-east-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='us-
east-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')

# Output path
'763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-inference:1.4.1-cpu-py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='us-
east-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')

# Output path
'520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-mxnet:coach0.11-cpu-py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='us-east-1')

# Output path
'156813124566.dkr.ecr.us-east-1.amazonaws.com/sagemaker-model-monitor-analyzer'
```

注册表路径	版本	Job 类型（图像范围）		
156813124566.dkr.ecr.us-east-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='us-east-1')

# Output path
'382416733822.dkr.ecr.us-east-1.amazonaws.com/ntm:1'
```

注册表路径	版本	Job 类型（图像范围）		
382416733822.dkr.ecr.us-east-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='us-east-1')

# Output path
'785573368785.dkr.ecr.us-east-1.amazonaws.com/image-classification-neo:latest'
```

注册表路径	版本	Job 类型（图像范围）		
785573368785.dkr.ecr.us-east-1.amazonaws.com/image-classification-neo：<tag>	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='us-
east-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')

# Output path
'785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-mxnet:1.8-cpu-py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-mxnet：<tag>	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='us-
east-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')

# Output path
'785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-pytorch:1.6-cpu-py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.6	推断	CPU，GPU	py3
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.7	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.6	推断	CPU，GPU	py3
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.5	推断	CPU，GPU	py3
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='us-
east-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')

# Output path
'785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-tensorflow:1.15.3-cpu-
py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-tensorflow：<tag>	2.4	推断	CPU，GPU	py3
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-inference-tensorflow：<tag>	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='us-east-1')
```

```
# Output path
'785573368785.dkr.ecr.us-east-1.amazonaws.com/xgboost-neo:latest'
```

注册表路径	版本	Job 类型（图像范围）		
785573368785.dkr.ecr.us-east-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='us-east-1')

# Output path
'811284229777.dkr.ecr.us-east-1.amazonaws.com/object-detection:1'
```

注册表路径	版本	Job 类型（图像范围）		
811284229777.dkr.ecr.us-east-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='us-east-1')

# Output path
'382416733822.dkr.ecr.us-east-1.amazonaws.com/object2vec:1'
```

注册表路径	版本	Job 类型（图像范围）		
382416733822.dkr.ecr.us-east-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='us-east-1')

# Output path
'382416733822.dkr.ecr.us-east-1.amazonaws.com/pca:1'
```

注册表路径	版本	Job 类型（图像范围）		
382416733822.dkr.ecr.us-east-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='us-
east-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')

# Output path
'763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference:1.8.0-cpu-py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.11.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.11.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.5.1	eia	CPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='us-east-1')

# Output path
'382416733822.dkr.ecr.us-east-1.amazonaws.com/randomcutforest:1'
```

注册表路径	版本	Job 类型（图像范围）		
382416733822.dkr.ecr.us-east-1.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='us-
east-1',version='0.8.5',instance_type='ml.c5.4xlarge')

# Output path
'462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-torch-
cpu-py36'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-ray-container训练:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='us-
east-1',version='0.23-1',image_scope='inference')

# Output path
'683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-scikit-learn:0.23-1-cpu-py3'
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='us-east-1')

# Output path
'811284229777.dkr.ecr.us-east-1.amazonaws.com/semantic-segmentation:1'
```

注册表路径	版本	Job 类型（图像范围）		
811284229777.dkr.ecr.us-east-1.amazonaws.com/	1	训练、推理		

注册表路径	版本	Job 类型（图像范围）		
semantic-segmentation：<tag>				

Seq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='us-east-1')

# Output path
'811284229777.dkr.ecr.us-east-1.amazonaws.com/seq2seq:1'
```

注册表路径	版本	Job 类型（图像范围）		
811284229777.dkr.ecr.us-east-1.amazonaws.com/seq2seq：<tag>	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='us-
east-1',version='3.0',image_scope='processing')

# Output path
'173754725891.dkr.ecr.us-east-1.amazonaws.com/sagemaker-spark-processing:3.0-cpu'
```

注册表路径	版本	Job 类型（图像范围）		
173754725891.dkr.ecr.us-east-1.amazonaws.com/sagemaker-spark-processing：<tag>	3.1	处理		
173754725891.dkr.ecr.us-east-1.amazonaws.com/sagemaker-spark-processing：<tag>	3.0	处理		
173754725891.dkr.ecr.us-east-1.amazonaws.com/sagemaker-spark-processing：<tag>	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='sparkml-serving',region='us-east-1',version='2.4')

# Output path
'683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-sparkml-serving:2.4'
```

注册表路径	版本	Job 类型（图像范围）		
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='us-
east-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')

# Output path
'520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving:1.12.0-cpu'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.3.0	eia	CPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.1	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.2	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.0.0	eia	CPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.14.0	eia	CPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13.0	eia	CPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='us-
east-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')

# Output path
'462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-coach-container:coach-1.0.0-tf-
cpu-py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-coach-container:	1.0.0	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
教练 -1.0.0-tf-*<tag>*				
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='us-
east-1',version='1.15.0',instance_type='ml.inf1.6xlarge')

# Output path
'785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-neo-tensorflow:1.15.0-inf-py3'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
785573368785.dkr.ecr.us-east-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
785573368785.dkr.ecr.us-east-1.amazonaws.com/	1.15.0	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
sagemaker-neo-tensorflow：*<tag>*				

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='us-
east-1',version='0.8.5',instance_type='ml.c5.4xlarge')

# Output path
'462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-cpu-
py36'
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-*<tag>*	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.2-*<tag>*	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-<tag>	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='us-
east-1',version='8.7.0',image_scope='training')

# Output path
'462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-cpu'
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.us-east-1.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-<tag>	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='us-east-1',version='1.2-1')

# Output path
'683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost:1.2-1'
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost：<tag>	1.5-1	1.5.2	训练、推理	
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost：<tag>	1.3-1	1.3.3	训练、推理	
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-2	1.2.0	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost：<tag>	us-2-1	1.2.0	训练、推理	
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost：<tag>	us-0-1	1.0	训练、推理	
811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost：<tag>	us-	1.5.2	训练、推理	
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost：<tag>	us-90-2	0.90	训练、推理	
683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost：<tag>	us-90-1	0.90	训练、推理	

美国西部（加利福尼亚北部）的 Docker 注册表路径和示例代码 (us-west-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 864)
- BlazingText（算法）(p. 865)
- Chainer（DLC）(p. 865)
- 澄清（算法）(p. 866)
- 数据 Wrangler（算法）(p. 866)
- 调试器（算法）(p. 866)
- DeepAR 预测（算法）(p. 866)
- 分解机器（算法）(p. 867)
- 拥抱脸（算法）(p. 867)
- 知识产权见解（算法）(p. 870)
- 图像分类（算法）(p. 870)
- 推理 mXNet (DLC) (p. 870)
- 推理 PyTorch (DLC) (p. 871)
- K-Mean (算法) (p. 871)
- KNN（算法）(p. 872)
- LDA（算法）(p. 872)
- 线性学习者（算法）(p. 872)
- MxNet (DLC) (p. 872)
- MxNet 教练（DLC）(p. 875)
- 模型监视器（算法）(p. 875)

- NTM（算法）(p. 876)
- Neo 图像分类 (算法) (p. 876)
- Neo MxNet (DLC) (p. 876)
- 新 PyTorch (DLC) (p. 877)
- Neo TensorFlow (DLC) (p. 877)
- Neo xgBoost（算法）(p. 878)
- 对象检测（算法）(p. 878)
- Object2VEC (算法) (p. 878)
- PCA（算法）(p. 879)
- pyTorch (DLC) (p. 879)
- 随机森林砍伐 (算法) (p. 882)
- 射线 PyTorch (DLC) (p. 882)
- Scikit-learn (算法) (p. 883)
- 语义分割（算法）(p. 883)
- Seq2Seq (算法) (p. 884)
- Spark（算法）(p. 884)
- SparkML 服务（算法）(p. 884)
- Tensorflow (DLC) (p. 885)
- Tensorflow 教练（DLC）(p. 893)
- 张量流推断 (DLC) (p. 894)
- Tensorflow Ray (DLC) (p. 894)
- 大众（算法）(p. 895)
- xgBoost（算法）(p. 896)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='us-
west-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-west-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/	0.3.2	推断		

注册表路径	版本	Job 类型（图像范围）		
autogluon-inference：*<tag>*				
763104351884.dkr.ecr.us-west-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='us-
west-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='us-
west-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
740489534195.dkr.ecr.us-west-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
926135532090.dkr.ecr.us-west-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
685455198987.dkr.ecr.us-west-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='us-
west-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/	4.12.3	培训		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-training：<tag>				
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.11.0	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.11.0	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.us-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='us-
west-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='us-
west-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Mean (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='us-
west-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12-1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12-1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='us-
west-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='model-monitor',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
890145073186.dkr.ecr.us-west-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
710691900526.dkr.ecr.us-west-1.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='us-
west-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
710691900526.dkr.ecr.us-west-1.amazonaws.com/	1.8	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
sagemaker-inference-mxnet：*<tag>*				

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='us-
west-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='us-
west-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4-2	推断	CPU，GPU	py3
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15-3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
710691900526.dkr.ecr.us-west-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='us-
west-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.11.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.11.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/	1.10.0	推断	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
pytorch-inference：*<tag>*				
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.10	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.10	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.09	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.09	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='us-
west-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='us-
west-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Seq2Seq (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='us-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
632365934929.dkr.ecr.us-west-1.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='us-
west-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
667973535471.dkr.ecr.us-west-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
667973535471.dkr.ecr.us-west-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
667973535471.dkr.ecr.us-west-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='us-west-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		

注册表路径	版本	Job 类型（图像范围）		
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='us-
west-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.3.0	eia	CPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.3	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.2	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.0.0	eia	CPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.14.0	eia	CPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13.0	eia	CPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.6.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='us-
west-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-coach-container:教练-1.0.0-TFF-<tag>	1.0.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-<tag>	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-	0.11.0	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.11.0-*<tag>*				
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='us-
west-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
710691900526.dkr.ecr.us-west-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='us-
west-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-TFF-*<tag>*	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-*<tag>*	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.2-TF-*<tag>*	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='us-
west-1',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.us-west-1.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-<tag>	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='us-west-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-xgboost：<tag>	1.5-1	1.5.2	训练、推理	
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-xgboost：<tag>	1.3-1	1.3.3	训练、推理	
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-2	1.2.0	训练、推理	
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-1	1.2.0	训练、推理	
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-xgboost：<tag>	1.0-1	1.0	训练、推理	
632365934929.dkr.ecr.us-west-1.amazonaws.com/xgboost：<tag>	1.5-2	1.5.2	训练、推理	
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	训练、推理	
746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-1	0.90	训练、推理	

美国西部（俄勒冈）(us-west-2) Docker 注册表路径和示例代码

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 898)
- BlazingText（算法）(p. 898)
- Chainer（DLC）(p. 899)
- 澄清（算法）(p. 899)
- 数据 Wrangler（算法）(p. 899)
- 调试器（算法）(p. 900)
- DeepAR 预测（算法）(p. 900)
- 分解机器（算法）(p. 900)
- 拥抱脸（算法）(p. 900)
- 知识产权见解（算法）(p. 903)
- 图像分类（算法）(p. 904)
- 推理 mXNet (DLC) (p. 904)
- 推理 PyTorch (DLC) (p. 904)
- K-Meean（算法）(p. 905)
- KNN（算法）(p. 905)
- LDA（算法）(p. 905)
- 线性学习者（算法）(p. 906)
- MxNet (DLC) (p. 906)
- MxNet 教练（DLC）(p. 908)
- 模型监视器（算法）(p. 909)
- NTM（算法）(p. 909)
- Neo 图像分类（算法）(p. 909)
- Neo MxNet (DLC) (p. 910)
- 新 PyTorch (DLC) (p. 910)
- Neo TensorFlow (DLC) (p. 911)
- Neo xgBoost（算法）(p. 911)
- 对象检测（算法）(p. 912)
- Object2VEC（算法）(p. 912)
- PCA（算法）(p. 912)
- pyTorch (DLC) (p. 912)
- 随机森林砍伐 (算法) (p. 916)
- 射线 PyTorch (DLC) (p. 916)
- Scikit-learn（算法）(p. 916)
- 语义分割（算法）(p. 917)
- Sq2Seq (算法) (p. 917)
- Spark（算法）(p. 917)
- SparkML 服务（算法）(p. 918)
- Tensorflow (DLC) (p. 918)
- Tensorflow 教练（DLC）(p. 927)
- 张量流推断 (DLC) (p. 928)
- Tensorflow Ray (DLC) (p. 928)
- 大众（算法）(p. 929)

- xgBoost（算法）(p. 929)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='us-
west-2',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-west-2.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
433757028032.dkr.ecr.us-west-2.amazonaws.com/blazingtext：*<tag>*	1	训练推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='us-
west-2',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='us-
west-2',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
306415355426.dkr.ecr.us-west-2.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
174368400705.dkr.ecr.us-west-2.amazonaws.com/sagemaker-	1.x	处理		

注册表路径	版本	Job 类型（图像范围）		
data-wrangler-container：*<tag>*				

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
895741380848.dkr.ecr.us-west-2.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
156387875391.dkr.ecr.us-west-2.amazonaws.com/forecasting-deepar：*<tag>*	1	训练推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
174872318107.dkr.ecr.us-west-2.amazonaws.com/factorization-machines：*<tag>*	1	训练推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='us-
west-2',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/	4.11.0	推断		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-inference：*<tag>*				
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.us-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
174872318107.dkr.ecr.us-west-2.amazonaws.com/ipinsights：*<tag>*	1	训练推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
433757028032.dkr.ecr.us-west-2.amazonaws.com/image-classification：*<tag>*	1	训练推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='us-
west-2',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='us-
west-2',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
174872318107.dkr.ecr.us-west-2.amazonaws.com/kmeans：*<tag>*	1	训练推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
174872318107.dkr.ecr.us-west-2.amazonaws.com/knn：*<tag>*	1	训练推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
266724342769.dkr.ecr.us-west-2.amazonaws.com/lda：*<tag>*	1	训练推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
174872318107.dkr.ecr.us-west-2.amazonaws.com/linear-learner：*<tag>*	1	训练推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='us-
west-2',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference：<tag>	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference-eia：<tag>	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-training：<tag>	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet-serving：<tag>	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference：<tag>	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference-eia：<tag>	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet-serving：<tag>	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet-serving-eia：<tag>	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet-eia：<tag>	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='coach-mxnet',region='us-
west-2',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
159807026194.dkr.ecr.us-west-2.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
174872318107.dkr.ecr.us-west-2.amazonaws.com/ntm：*<tag>*	1	训练推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='image-classification-neo',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
301217895009.dkr.ecr.us-west-2.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='us-
west-2',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='us-
west-2',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
inference-pytorch：*<tag>*				
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='us-
west-2',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
301217895009.dkr.ecr.us-west-2.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
433757028032.dkr.ecr.us-west-2.amazonaws.com/object-detection：*<tag>*	1	训练推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
174872318107.dkr.ecr.us-west-2.amazonaws.com/object2vec：*<tag>*	1	训练推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
174872318107.dkr.ecr.us-west-2.amazonaws.com/pca：*<tag>*	1	训练推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='us-
west-2',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：<tag>	1.11.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：<tag>	1.11.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：<tag>	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：<tag>	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：<tag>	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：<tag>	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：<tag>	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：<tag>	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：<tag>	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：<tag>	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：<tag>	1.8.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference-eia：*<tag>*	1.5.1	eia	CPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
174872318107.dkr.ecr.us-west-2.amazonaws.com/randomcutforest：*<tag>*	1	训练推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='us-
west-2',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-火炬*<tag>*	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='sklearn',region='us-
west-2',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
246618743249.dkr.ecr.us-west-2.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练推理	
246618743249.dkr.ecr.us-west-2.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
433757028032.dkr.ecr.us-west-2.amazonaws.com/semantic-segmentation：*<tag>*	1	训练推理		

Sq2Seq (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='us-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
433757028032.dkr.ecr.us-west-2.amazonaws.com/seq2seq：*<tag>*	1	训练推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='us-
west-2',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
153931337802.dkr.ecr.us-west-2.amazonaws.com/sagemaker-spark-processing：<tag>	3.1	处理		
153931337802.dkr.ecr.us-west-2.amazonaws.com/sagemaker-spark-processing：<tag>	3.0	处理		
153931337802.dkr.ecr.us-west-2.amazonaws.com/sagemaker-spark-processing：<tag>	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='us-west-2',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
246618743249.dkr.ecr.us-west-2.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.4	推断		
246618743249.dkr.ecr.us-west-2.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='us-
west-2',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/	2.8.0	培训	CPU，GPU	py39

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow-training：*<tag>*				
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.0.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	1.15.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.0.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.0.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.0.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.0.2	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.0.0	eia	CPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.13.0	eia	CPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	1.13.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.12.0	eia	CPU	-
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.12.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.11.0	eia	CPU	-
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-eia：<tag>	1.10.0	eia	CPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	培训	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='us-
west-2',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-coach-container: 教练 -1.0.0-TF-*<tag>*	1.0.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='us-
west-2',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
301217895009.dkr.ecr.us-west-2.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='us-
west-2',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-ray-containerTFay-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-*<tag>*	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-ray-containerTFay-0.8.2-*<tag>*	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='us-
west-2',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.us-west-2.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-*<tag>*	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='us-west-2',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
246618743249.dkr.ecr.us-west-2.amazonaws.com/sagemaker-xgboost：*<tag>*	1.5-1	1.5.2	训练推理	
246618743249.dkr.ecr.us-west-2.amazonaws.com/	1.3-1	1.3.3	训练推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
sagemaker-xgboost：<tag>				
246618743249.dkr.ecr.us-west-2.amazonaws.com/sagemaker-xgboost：<tag>	us-west-2	1.2.0	训练推理	
246618743249.dkr.ecr.us-west-2.amazonaws.com/sagemaker-xgboost：<tag>	us-west-1	1.2.0	训练推理	
246618743249.dkr.ecr.us-west-2.amazonaws.com/sagemaker-xgboost：<tag>	us-west-1	1.0	训练推理	
433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost：<tag>	us-west-2	1.5.2	训练推理	
246618743249.dkr.ecr.us-west-2.amazonaws.com/sagemaker-xgboost：<tag>	us-west-2	0.90	训练推理	
246618743249.dkr.ecr.us-west-2.amazonaws.com/sagemaker-xgboost：<tag>	us-west-1	0.90	训练推理	

非洲 Docker 注册表路径和示例代码（开普敦）(af-south-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 931)
- BlazingText（算法）(p. 932)
- Chainer（DLC）(p. 932)
- 澄清（算法）(p. 933)
- 数据 Wrangler（算法）(p. 933)
- 调试器（算法）(p. 933)
- DeepAR 预测（算法）(p. 934)
- 分解机器（算法）(p. 934)
- 拥抱脸（算法）(p. 934)
- 知识产权见解（算法）(p. 937)
- 图像分类（算法）(p. 937)
- 推理 mXNet (DLC) (p. 938)
- 推理 PyTorch (DLC) (p. 938)
- K-Meensor（算法）(p. 938)

- KNN（算法）(p. 939)
- 线性学习者（算法）(p. 939)
- MxNet (DLC) (p. 939)
- MxNet 教练（DLC）(p. 942)
- 模型监视器（算法）(p. 942)
- NTM（算法）(p. 943)
- Neo 图像分类（算法）(p. 943)
- Neo MxNet (DLC) (p. 943)
- 新 PyTorch (DLC) (p. 944)
- Neo TensorFlow (DLC) (p. 944)
- Neo xgBoost（算法）(p. 945)
- 对象检测（算法）(p. 945)
- Object2VEC（算法）(p. 945)
- PCA（算法）(p. 946)
- pyTorch (DLC) (p. 946)
- 随机森林砍伐 (算法) (p. 949)
- Scikit-learn（算法）(p. 949)
- 语义分割（算法）(p. 950)
- Sq2Seq（算法）(p. 950)
- Spark（算法）(p. 950)
- SparkML 服务（算法）(p. 951)
- Tensorflow (DLC) (p. 951)
- Tensorflow 教练（DLC）(p. 959)
- 张量流推断 (DLC) (p. 960)
- Tensorflow Ray (DLC) (p. 961)
- xgBoost（算法）(p. 961)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='af-
south-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
626614931356.dkr.ecr.af-south-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/	0.3.2	培训		

注册表路径	版本	Job 类型（图像范围）		
autogluon-training：<tag>				
626614931356.dkr.ecr.af-south-1.amazonaws.com/autogluon-inference：<tag>	0.3.2	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/autogluon-training：<tag>	0.3.1	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/autogluon-inference：<tag>	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/blazingtext：<tag>	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='af-
south-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-chainer：<tag>	5.0.0	训练、推理	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-chainer：<tag>	4.1.0	训练、推理	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='af-
south-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
811711786498.dkr.ecr.af-south-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
143210264188.dkr.ecr.af-south-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
314341159256.dkr.ecr.af-south-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/forecasting-deepar：*<tag>*	1	推理、训练		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/factorization-machines：*<tag>*	1	推理、训练		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='af-
south-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		

注册表路径	版本	Job 类型（图像范围）		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		

注册表路径	版本	Job 类型（图像范围）		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		

注册表路径	版本	Job 类型（图像范围）		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
626614931356.dkr.ecr.af-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='af-
south-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='af-
south-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meensor（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='af-
south-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.af-south-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
626614931356.dkr.ecr.af-south-1.amazonaws.com/	1.8.0	推断	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference：*<tag>*				
626614931356.dkr.ecr.af-south-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
626614931356.dkr.ecr.af-south-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
626614931356.dkr.ecr.af-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.af-south-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.8.0	培训	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.8.0	推断	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12.1	培训	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='af-
south-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
875698925577.dkr.ecr.af-south-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
774647643957.dkr.ecr.af-south-1.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='af-
south-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='af-
south-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='af-
south-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-	2.4-2	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
inference-tensorflow：<tag>				
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-inference-tensorflow：<tag>	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
774647643957.dkr.ecr.af-south-1.amazonaws.com/xgboost-neo：<tag>	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/object-detection：<tag>	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/object2vec：<tag>	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='af-
south-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.af-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py38
626614931356.dkr.ecr.af-south-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py38
626614931356.dkr.ecr.af-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
626614931356.dkr.ecr.af-south-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
626614931356.dkr.ecr.af-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
626614931356.dkr.ecr.af-south-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-inference：<tag>	1.9.1	推断	CPU，GPU	py38
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-training：<tag>	1.9.1	培训	CPU，GPU	py38
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-inference：<tag>	1.9.0	推断	CPU，GPU	py38
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-training：<tag>	1.9.0	培训	CPU，GPU	py38
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-inference：<tag>	1.8.1	推断	CPU，GPU	py3、py36
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-training：<tag>	1.8.1	培训	CPU，GPU	py3、py36
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-inference：<tag>	1.8.0	推断	CPU，GPU	py3、py36
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-training：<tag>	1.8.0	培训	CPU，GPU	py3、py36
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-inference：<tag>	1.7.1	推断	CPU，GPU	py3、py36
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-training：<tag>	1.7.1	培训	CPU，GPU	py3、py36
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-inference：<tag>	1.6.0	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
626614931356.dkr.ecr.eu-south-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
313743910680.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
313743910680.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.8.0	推断	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.8.0	培训	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/randomcutforest：*<tag>*	1	推理、训练		

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='af-
south-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
510948584623.dkr.ecr.af-south-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	推理、训练	
510948584623.dkr.ecr.af-south-1.amazonaws.com/	0.20.0	0.20.0	推理、训练	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
sagemaker-scikit-learn：*<tag>*				

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/semantic-segmentation：*<tag>*	1	推理、训练		

Sq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='af-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
455444449433.dkr.ecr.af-south-1.amazonaws.com/seq2seq：*<tag>*	1	推理、训练		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='af-
south-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
309385258863.dkr.ecr.af-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
309385258863.dkr.ecr.af-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		

注册表路径	版本	Job 类型（图像范围）		
309385258863.dkr.ecr.af-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='af-south-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
510948584623.dkr.ecr.af-south-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
510948584623.dkr.ecr.af-south-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='af-
south-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.8.0	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.8.0	培训	CPU，GPU	py39
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
626614931356.dkr.ecr.af-south-1.amazonaws.com/	2.7.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow-inference：*<tag>*				
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.3	推断	CPU，GPU	-
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.3	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py37
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.2	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.2	培训	CPU，GPU	py37
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.1	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.3.0	eia	CPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.0	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.0	培训	CPU，GPU	py37
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.2	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.2	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.1	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.1	培训	CPU，GPU	py37
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.0	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.0	培训	CPU，GPU	py37
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.3	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.3	培训	CPU，GPU	py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.2	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.2	培训	CPU，GPU	py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.1	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.1	培训	CPU，GPU	py2、py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.0	培训	CPU，GPU	py2、py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.4	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.4	培训	CPU，GPU	py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.3	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.3	培训	CPU，GPU	py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.2	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.2	培训	CPU，GPU	py2、py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.1	推断	CPU，GPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.1	培训	CPU，GPU	py2、py3
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.0.0	eia	CPU	-
626614931356.dkr.ecr.af-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
626614931356.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
626614931356.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py2、py3
626614931356.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.14.0	eia	CPU	-
626614931356.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
626614931356.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py2
626614931356.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-serving-eia：*<tag>*	1.13.0	eia	CPU	-
626614931356.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
313743910680.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
313743910680.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
313743910680.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
313743910680.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
313743910680.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
313743910680.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
313743910680.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
313743910680.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
313743910680.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
313743910680.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='af-
south-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='af-
south-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
774647643957.dkr.ecr.af-south-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='af-
south-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
313743910680.dkr.ecr.af-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='af-south-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
510948584623.dkr.ecr.af-south-1.amazonaws.com/sagemaker-xgboost：*<tag>*	af-5-1	1.5.2	推理、训练	
510948584623.dkr.ecr.af-south-1.amazonaws.com/sagemaker-xgboost：*<tag>*	af-3-1	1.3.3	推理、训练	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
510948584623.dkr.ecr.af-south-1.amazonaws.com/sagemaker-xgboost：<tag>	af-2-2	1.2.0	推理、训练	
510948584623.dkr.ecr.af-south-1.amazonaws.com/sagemaker-xgboost：<tag>	af-2-1	1.2.0	推理、训练	
510948584623.dkr.ecr.af-south-1.amazonaws.com/sagemaker-xgboost：<tag>	af-0-1	1.0	推理、训练	
455444449433.dkr.ecr.af-south-1.amazonaws.com/xgboost：<tag>	af-	1.5.2	推理、训练	
510948584623.dkr.ecr.af-south-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	推理、训练	
510948584623.dkr.ecr.af-south-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-1	0.90	推理、训练	

亚太地区注册管理机构路径和示例代码（香港）(ap-east-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 963)
- BlazingText（算法）(p. 964)
- Chainer（DLC）(p. 964)
- 澄清（算法）(p. 965)
- 数据 Wrangler（算法）(p. 965)
- 调试器（算法）(p. 965)
- DeepAR 预测（算法）(p. 966)
- 分解机器（算法）(p. 966)
- 拥抱脸（算法）(p. 966)
- 知识产权见解（算法）(p. 969)
- 图像分类（算法）(p. 969)
- 推理 mXNet (DLC) (p. 970)
- 推理 PyTorch (DLC) (p. 970)
- K-Meensor（算法）(p. 970)
- KNN（算法）(p. 971)

- 线性学习者（算法）(p. 971)
- MxNet (DLC) (p. 971)
- MxNet 教练（DLC）(p. 974)
- 模型监视器（算法）(p. 974)
- NTM（算法）(p. 975)
- Neo 图像分类（算法）(p. 975)
- Neo MxNet (DLC) (p. 975)
- 新 PyTorch (DLC) (p. 976)
- Neo TensorFlow (DLC) (p. 976)
- Neo xgBoost（算法）(p. 977)
- 对象检测（算法）(p. 977)
- Object2VEC（算法）(p. 977)
- PCA（算法）(p. 978)
- pyTorch (DLC) (p. 978)
- 随机森林砍伐 (算法) (p. 981)
- Scikit-learn（算法）(p. 981)
- 语义分割（算法）(p. 982)
- Sq2Seq（算法）(p. 982)
- Spark（算法）(p. 982)
- SparkML 服务（算法）(p. 983)
- Tensorflow (DLC) (p. 983)
- Tensorflow 教练（DLC）(p. 991)
- 张量流推断 (DLC) (p. 992)
- Tensorflow Ray (DLC) (p. 993)
- xgBoost（算法）(p. 993)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='ap-
east-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/	0.3.2	培训		

注册表路径	版本	Job 类型（图像范围）		
autogluon-training：*<tag>*				
871362719292.dkr.ecr.ap-east-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='ap-
east-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='ap-
east-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
098760798382.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
707077482487.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
199566480951.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='ap-
east-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		

注册表路径	版本	Job 类型（图像范围）		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		

注册表路径	版本	Job 类型（图像范围）		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		

注册表路径	版本	Job 类型（图像范围）		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
871362719292.dkr.ecr.ap-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='ap-
east-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='ap-
east-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meensor（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='ap-
east-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.ap-east-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/	1.8.0	推断	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference：*<tag>*				
871362719292.dkr.ecr.ap-east-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
871362719292.dkr.ecr.ap-east-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
871362719292.dkr.ecr.ap-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.ap-east-1.amazonaws.com/mxnet-inference-eia：<tag>	1.4.1	eia	CPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet-serving：<tag>	1.4.0	推断	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet-serving-eia：<tag>	1.4.0	eia	CPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	推断	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet-eia：<tag>	1.3.0	eia	CPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.8.0	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.8.0	推断	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='ap-
east-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
001633400207.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
110948597952.dkr.ecr.ap-east-1.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='ap-
east-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='ap-
east-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='ap-
east-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-	2.4.2	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
inference-tensorflow：*<tag>*				
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15-3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
110948597952.dkr.ecr.ap-east-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='ap-
east-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0	推断	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4	推断	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='ap-
east-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
651117190479.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
651117190479.dkr.ecr.ap-east-1.amazonaws.com/	0.20.0	0.20.0	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
sagemaker-scikit-learn：*<tag>*				

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Sq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='ap-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
286214385809.dkr.ecr.ap-east-1.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='ap-
east-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
732049463269.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
732049463269.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		

注册表路径	版本	Job 类型（图像范围）		
732049463269.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-spark-processing：<tag>	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='ap-east-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
651117190479.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.4	推断		
651117190479.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='ap-
east-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/	2.7.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow-inference：*<tag>*				
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.3	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.3	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.2	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.2	培训	CPU，GPU	py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.1	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.3.0	eia	CPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.0	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.0	培训	CPU，GPU	py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.2	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.2	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.0	培训	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13.1	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.12.0	eia	CPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.12.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
871362719292.dkr.ecr.ap-east-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
871362719292.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
871362719292.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
871362719292.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
871362719292.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.13.1	培训	CPU，GPU	py2
871362719292.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3
057415533634.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.13.0	eia	CPU	-
871362719292.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	1.13.0	推断	CPU，GPU	-
057415533634.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.12.0	eia	CPU	-
057415533634.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.12.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='ap-
east-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='ap-
east-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
110948597952.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='ap-
east-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
057415533634.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='ap-east-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
651117190479.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.5-1	1.5.2	训练、推理	
651117190479.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.3-1	1.3.3	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
651117190479.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-xgboost：<tag>	ap2-2	1.2.0	训练、推理	
651117190479.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-xgboost：<tag>	ap2-1	1.2.0	训练、推理	
651117190479.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-xgboost：<tag>	ap0-1	1.0	训练、推理	
286214385809.dkr.ecr.ap-east-1.amazonaws.com/xgboost：<tag>	ap-	1.5.2	训练、推理	
651117190479.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-xgboost：<tag>	ap90-2	0.90	训练、推理	
651117190479.dkr.ecr.ap-east-1.amazonaws.com/sagemaker-xgboost：<tag>	ap90-1	0.90	训练、推理	

亚太地区 Docker 注册表路径和示例代码（孟买）(ap-south-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 995)
- BlazingText（算法）(p. 996)
- Chainer（DLC）(p. 996)
- 澄清（算法）(p. 997)
- 数据 Wrangler（算法）(p. 997)
- 调试器（算法）(p. 997)
- DeepAR 预测（算法）(p. 998)
- 分解机器（算法）(p. 998)
- 拥抱脸（算法）(p. 998)
- 知识产权见解（算法）(p. 1001)
- 图像分类（算法）(p. 1001)
- 推理 mXNet (DLC) (p. 1002)
- 推理 PyTorch (DLC) (p. 1002)
- K-Mean（算法）(p. 1002)
- KNN（算法）(p. 1003)
- LDA（算法）(p. 1003)

- 线性学习者（算法）(p. 1003)
- MxNet (DLC) (p. 1003)
- MxNet 教练（DLC）(p. 1006)
- 模型监视器（算法）(p. 1007)
- NTM（算法）(p. 1007)
- Neo 图像分类（算法）(p. 1007)
- Neo MxNet (DLC) (p. 1007)
- 新 PyTorch (DLC) (p. 1008)
- Neo TensorFlow (DLC) (p. 1008)
- Neo xgBoost（算法）(p. 1009)
- 对象检测（算法）(p. 1009)
- Object2VEC（算法）(p. 1009)
- PCA（算法）(p. 1010)
- pyTorch (DLC) (p. 1010)
- 随机砍伐森林（算法）(p. 1013)
- 射线 PyTorch (DLC) (p. 1013)
- Scikit-learn（算法）(p. 1014)
- 语义分割（算法）(p. 1014)
- Sq2Seq（算法）(p. 1015)
- Spark（算法）(p. 1015)
- SparkML 服务（算法）(p. 1015)
- Tensorflow (DLC) (p. 1016)
- Tensorflow 教练（DLC）(p. 1024)
- 张量流推断 (DLC) (p. 1025)
- Tensorflow Ray (DLC) (p. 1026)
- 大众（算法）(p. 1026)
- xgBoost（算法）(p. 1027)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='ap-
south-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/	0.3.2	培训		

注册表路径	版本	Job 类型（图像范围）		
autogluon-training：*<tag>*				
763104351884.dkr.ecr.ap-south-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/blazingtext：*<tag>*	1	训练推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='ap-
south-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练推理	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='ap-
south-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
452307495513.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
089933028263.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
904829902805.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/factorization-machines：*<tag>*	1	训练推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='ap-
south-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.ap-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/ipinsights：*<tag>*	1	训练推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/image-classification：*<tag>*	1	训练推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='ap-
south-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='ap-
south-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Mean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/kmeans：*<tag>*	1	训练推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/knn：*<tag>*	1	训练推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/lda：*<tag>*	1	训练推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/linear-learner：*<tag>*	1	训练推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='mxnet',region='ap-
south-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/	1.4.1	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
sagemaker-mxnet-serving：*<tag>*				
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='ap-
south-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-<tag>	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
126357580389.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/ntm：*<tag>*	1	训练推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
763008648453.dkr.ecr.ap-south-1.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='ap-
south-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-inference-mxnet：<tag>	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='ap-
south-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.8	推断	CPU，GPU	py3
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.7	推断	CPU，GPU	py3
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.6	推断	CPU，GPU	py3
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.5	推断	CPU，GPU	py3
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='neo-tensorflow',region='ap-
south-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
763008648453.dkr.ecr.ap-south-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/object-detection：*<tag>*	1	训练推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/object2vec：*<tag>*	1	训练推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/pca：*<tag>*	1	训练推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='ap-
south-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/	1.10.0	推断	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
pytorch-inference：*<tag>*				
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机砍伐森林（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/randomcutforest：*<tag>*	1	训练推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='ray-pytorch',region='ap-
south-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-火炬*<tag>*	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='ap-
south-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练推理	
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/	1	训练推理		

注册表路径	版本	Job 类型（图像范围）		
semantic-segmentation：*<tag>*				

Sq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='ap-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
991648021394.dkr.ecr.ap-south-1.amazonaws.com/seq2seq：*<tag>*	1	训练推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='ap-
south-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
105495057255.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
105495057255.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
105495057255.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='ap-south-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.4	推断		
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='ap-
south-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/	2.6.2	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow-training：*<tag>*				
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.2	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.3.0	eia	CPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-
763104351884.dkr.ec2.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.0.4	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.0.0	eia	CPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py3、py36、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13.0	eia	CPU	-
763104351884.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='ap-
south-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-coach-container:教练 -1.0.0-TF-<tag>	1.0.0	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='ap-
south-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
763008648453.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='ap-
south-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-ray-containerProcess-1.6.0-TF-*<tag>*	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-*<tag>*	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-ray-containerProcess-0.8.2-TF-*<tag>*	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='ap-
south-1',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-<tag>	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='ap-south-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-xgboost：<tag>	ap5-1	1.5.2	训练推理	
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-xgboost：<tag>	ap3-1	1.3.3	训练推理	
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-xgboost：<tag>	ap2-2	1.2.0	训练推理	
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-xgboost：<tag>	ap2-1	1.2.0	训练推理	
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-xgboost：<tag>	ap0-1	1.0	训练推理	
991648021394.dkr.ecr.ap-south-1.amazonaws.com/xgboost：<tag>	ap-	1.5.2	训练推理	
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-xgboost：<tag>	ap90-2	0.90	训练推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-xgboost：*<tag>*	0.90-1	0.90	训练推理	

亚太地区（大阪）Docker 注册表路径和示例代码 (ap-northeast-3)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1029)
- BlazingText（算法）(p. 1029)
- 澄清（算法）(p. 1030)
- 调试器（算法）(p. 1030)
- 分解机器（算法）(p. 1030)
- 拥抱脸（算法）(p. 1030)
- 知识产权见解（算法）(p. 1033)
- 图像分类（算法）(p. 1033)
- 推理 mXNet (DLC) (p. 1034)
- 推理 PyTorch (DLC) (p. 1034)
- K-Meean（算法）(p. 1035)
- KNN（算法）(p. 1035)
- 线性学习者（算法）(p. 1035)
- MxNet (DLC) (p. 1035)
- 模型监视器（算法）(p. 1037)
- NTM（算法）(p. 1037)
- Neo 图像分类（算法）(p. 1037)
- Neo MxNet (DLC) (p. 1038)
- 新 PyTorch (DLC) (p. 1038)
- Neo TensorFlow (DLC) (p. 1039)
- Neo xgBoost（算法）(p. 1039)
- 对象检测（算法）(p. 1040)
- Object2VEC（算法）(p. 1040)
- PCA（算法）(p. 1040)
- pyTorch (DLC) (p. 1040)
- 随机森林砍伐 (算法) (p. 1043)
- Scikit-learn（算法）(p. 1043)
- 语义分割（算法）(p. 1044)
- Seq2Seq（算法）(p. 1044)
- Tensorflow (DLC) (p. 1044)
- 张量流推断 (DLC) (p. 1050)

- xgBoost（算法）(p. 1051)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='ap-
northeast-3',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='ap-
northeast-3',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
912233562940.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
479947661362.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='ap-
northeast-3',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/	4.11.0	推断		

注册表路径	版本	Job 类型（图像范围）		
huggingface-tensorflow-inference：*<tag>*				
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	版本 4.10.2	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	版本 4.10.2	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	版本 4.10.2	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-inference：*<tag>*	版本 4.10.2	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-inference：*<tag>*	版本 4.10.2	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	版本 4.10.2	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	版本 4.10.2	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		

注册表路径	版本	Job 类型（图像范围）		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='image-classification',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/image-classification：<tag>	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='ap-
northeast-3',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-neo-mxnet：<tag>	1.8	推断	inf	py3
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-neo-mxnet：<tag>	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='ap-
northeast-3',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-neo-pytorch：<tag>	1.9	推断	inf	py3
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-neo-pytorch：<tag>	1.8	推断	inf	py3
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/	1.7	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
sagemaker-neo-pytorch：*<tag>*				

K-Meean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/knn：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='mxnet',region='ap-
northeast-3',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/	1.4.1	eia	CPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：<tag>				
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-mxnet-serving-eia：<tag>	1.4.0	eia	CPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-mxnet-eia：<tag>	1.3.0	eia	CPU	py2、py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
990339680094.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-model-monitor-analyzer：<tag>		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/ntm：<tag>	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='ap-
northeast-3',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='ap-
northeast-3',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='ap-
northeast-3',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='ap-
northeast-3',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：<tag>	1.4.0	推断	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：<tag>	1.4.0	培训	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：<tag>	1.3.2	推断	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：<tag>	1.3.2	培训	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：<tag>	1.3.0	推断	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：<tag>	1.3.0	培训	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：<tag>	1.2.0	推断	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：<tag>	1.2.0	培训	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：<tag>	1.1.0	推断	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：<tag>	1.1.0	培训	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：<tag>	1.0.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference-eia：<tag>	1.3.1	eia	CPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：<tag>	1.3.1	推断	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：<tag>	1.3.1	培训	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-inference：<tag>	1.2.0	推断	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/pytorch-training：<tag>	1.2.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/randomcutforest：<tag>	1	训练、推理		

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='ap-
northeast-3',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/	0.23-1	0.23.2	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
sagemaker-scikit-learn：*<tag>*				
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Seq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='ap-northeast-3')
```

注册表路径	版本	Job 类型（图像范围）		
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='ap-
northeast-3',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/	2.8.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow-inference：*<tag>*				
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：*<tag>*	2.8.0	培训	CPU，GPU	py39
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.0.4	培训	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.0.3	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.0.3	培训	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.0.2	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.0.2	培训	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference-eia：<tag>	2.0.0	eia	CPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	1.15.5	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	1.15.5	培训	CPU，GPU	py3、py36、py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	1.15.4	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	1.15.4	培训	CPU，GPU	py3、py36、py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.14.0	eia	CPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py3
364406365360.dkr.ecr.ap-northeast-3.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='ap-
northeast-3',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-neo-tensorflow：<tag>	2.5.0	推断	inf	py3
925152966179.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-neo-tensorflow：<tag>	1.15.0	推断	inf	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='ap-northeast-3',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-xgboost：<tag>	1.5-1	1.5.2	训练、推理	
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-xgboost：<tag>	1.3-1	1.3.3	训练、推理	
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-xgboost：<tag>	1.2-2	1.2.0	训练、推理	
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-xgboost：<tag>	1.2-1	1.2.0	训练、推理	
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-xgboost：<tag>	1.0-1	1.0	训练、推理	
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/xgboost：<tag>	1.5-2	1.5.2	训练、推理	
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
867004704886.dkr.ecr.ap-northeast-3.amazonaws.com/sagemaker-xgboost：*<tag>*	0.90-1	0.90	训练、推理	

亚太地区 Docker 注册管理机构路径和示例代码 (ap-northeast-2)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1053)
- BlazingText（算法）(p. 1053)
- Chainer（DLC）(p. 1054)
- 澄清（算法）(p. 1054)
- 数据 Wrangler（算法）(p. 1054)
- 调试器（算法）(p. 1055)
- DeepAR 预测（算法）(p. 1055)
- 分解机器（算法）(p. 1055)
- 拥抱脸（算法）(p. 1056)
- 知识产权见解（算法）(p. 1058)
- 图像分类（算法）(p. 1059)
- 推理 mXNet (DLC) (p. 1059)
- 推理 PyTorch (DLC) (p. 1059)
- K-Meean（算法）(p. 1060)
- KNN（算法）(p. 1060)
- LDA（算法）(p. 1060)
- 线性学习者（算法）(p. 1061)
- MxNet (DLC) (p. 1061)
- MxNet 教练（DLC）(p. 1064)
- 模型监视器（算法）(p. 1064)
- NTM（算法）(p. 1064)
- Neo 图像分类（算法）(p. 1065)
- Neo MxNet (DLC) (p. 1065)
- 新 PyTorch (DLC) (p. 1065)
- Neo TensorFlow (DLC) (p. 1066)
- Neo xgBoost（算法）(p. 1066)
- 对象检测（算法）(p. 1067)
- Object2VEC（算法）(p. 1067)
- PCA（算法）(p. 1067)
- pyTorch (DLC) (p. 1068)
- 随机砍伐森林（算法）(p. 1071)
- 射线 PyTorch (DLC) (p. 1071)
- Scikit-learn（算法）(p. 1072)
- 语义分割（算法）(p. 1072)

- Seq2Seq（算法）(p. 1072)
- Spark（算法）(p. 1072)
- SparkML 服务（算法）(p. 1073)
- Tensorflow (DLC) (p. 1073)
- Tensorflow 教练（DLC）(p. 1082)
- 张量流推断 (DLC) (p. 1083)
- Tensorflow Ray (DLC) (p. 1083)
- 大众（算法）(p. 1084)
- xgBoost（算法）(p. 1084)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='ap-
northeast-2',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
306986355934.dkr.ecr.ap-northeast-2.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='ap-
northeast-2',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='ap-
northeast-2',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
263625296855.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
131546521161.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
578805364391.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
204372634319.dkr.ecr.ap-northeast-2.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
835164637446.dkr.ecr.ap-northeast-2.amazonaws.com/	1	训练、推理		

注册表路径	版本	Job 类型（图像范围）		
factorization-machines：<tag>				

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='ap-
northeast-2',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-training：<tag>	4.17.0	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-training：<tag>	4.17.0	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-inference：<tag>	4.17.0	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.17.0	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/	4.11.0	培训		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-training：<tag>				
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-training：<tag>	4.11.0	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
835164637446.dkr.ecr.ap-northeast-2.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
306986355934.dkr.ecr.ap-northeast-2.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='ap-
northeast-2',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='ap-
northeast-2',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
835164637446.dkr.ecr.ap-northeast-2.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
835164637446.dkr.ecr.ap-northeast-2.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
293181348795.dkr.ecr.ap-northeast-2.amazonaws.com/lda：<tag>	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
835164637446.dkr.ecr.ap-northeast-2.amazonaws.com/linear-learner：<tag>	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='ap-
northeast-2',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/mxnet-training：<tag>	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/mxnet-inference：<tag>	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/mxnet-training：<tag>	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/mxnet-inference：<tag>	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/	1.7.0	eia	CPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：*<tag>*				
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12-1	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2-1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='ap-
northeast-2',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
709848358524.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
835164637446.dkr.ecr.ap-northeast-2.amazonaws.com/ntm：<tag>	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/image-classification-neo：<tag>	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='ap-
northeast-2',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-inference-mxnet：<tag>	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='ap-
northeast-2',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-	1.8	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
inference-pytorch：*<tag>*				
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='ap-
northeast-2',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
306986355934.dkr.ecr.ap-northeast-2.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
835164637446.dkr.ecr.ap-northeast-2.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
835164637446.dkr.ecr.ap-northeast-2.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='ap-
northeast-2',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference：*<tag>*	1.10.2 1.10.0 1.9.1 1.9.0 1.8.1 1.8.0 1.7.1 1.6.0	推断	CPU，GPU	py38 py38 py38 py38 py3 py3 py3 py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-training：*<tag>*	1.10.2 1.10.0 1.9.1 1.9.0 1.8.1 1.8.0 1.7.1 1.6.0	培训	CPU，GPU	py38 py38 py38 py38 py3 py3 py3 py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
pytorch-training：*<tag>*				
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference-eia：*<tag>*	1.5.1	eia	CPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.9	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4	培训	CPU，GPU	py2、py3

随机砍伐森林（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
835164637446.dkr.ecr.ap-northeast-2.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='ap-
northeast-2',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-*<tag>*	1.6	培训	CPU，GPU	py36
462105765813.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='ap-
northeast-2',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
306986355934.dkr.ecr.ap-northeast-2.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Seq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='ap-northeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
306986355934.dkr.ecr.ap-northeast-2.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='spark',region='ap-
northeast-2',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
860869212795.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
860869212795.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
860869212795.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='ap-northeast-2',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='ap-
northeast-2',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：*<tag>*	2.8.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ec2.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.0.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.0.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.0.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.0.2	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.0.0	eia	CPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.13.0	eia	CPU	-
763104351884.dkr.ecr.ap-northeast-2.amazonaws.com/tensorflow-inference：<tag>	1.13.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.4	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='ap-
northeast-2',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-coach-container:Coach-1.0.0-TF-<tag>	1.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-<tag>	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-<tag>	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-<tag>	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-<tag>	0.10	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-<tag>	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='ap-
northeast-2',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
151534178276.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='ap-
northeast-2',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-TF-*<tag>*	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-*<tag>*	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.2-*<tag>*	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.8.5	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='ap-
northeast-2',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-*<tag>*	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='ap-northeast-2',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-xgboost：*<tag>*	1.5-1	1.5.2	训练、推理	
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/	1.3-1	1.3.3	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
sagemaker-xgboost：<tag>				
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-xgboost：<tag>	ap2-2	1.2.0	训练、推理	
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-xgboost：<tag>	ap2-1	1.2.0	训练、推理	
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-xgboost：<tag>	ap0-1	1.0	训练、推理	
306986355934.dkr.ecr.ap-northeast-2.amazonaws.com/xgboost：<tag>	ap-	1.5.2	训练、推理	
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-xgboost：<tag>	ap90-2	0.90	训练、推理	
366743142698.dkr.ecr.ap-northeast-2.amazonaws.com/sagemaker-xgboost：<tag>	ap90-1	0.90	训练、推理	

亚太地区 Docker 注册管理机构路径和示例代码（新加坡）(ap-southeast-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1086)
- BlazingText（算法）(p. 1087)
- Chainer（DLC）(p. 1087)
- 澄清（算法）(p. 1088)
- 数据 Wrangler（算法）(p. 1088)
- 调试器（算法）(p. 1088)
- DeepAR 预测（算法）(p. 1089)
- 分解机器（算法）(p. 1089)
- 拥抱脸（算法）(p. 1089)
- 知识产权见解（算法）(p. 1092)
- 图像分类（算法）(p. 1092)
- 推理 mXNet (DLC) (p. 1093)
- Inferenia PyTorch (DLC) (p. 1093)
- K-Mean（算法）(p. 1094)

- KNN（算法）(p. 1094)
- LDA（算法）(p. 1094)
- 线性学习者（算法）(p. 1094)
- MxNet (DLC) (p. 1095)
- MxNet 教练（DLC）(p. 1097)
- 模型监视器（算法）(p. 1098)
- NTM（算法）(p. 1098)
- Neo 图像分类（算法）(p. 1098)
- Neo MxNet (DLC) (p. 1098)
- 新 PyTorch (DLC) (p. 1099)
- Neo TensorFlow (DLC) (p. 1100)
- Neo xgBoost（算法）(p. 1100)
- 对象检测（算法）(p. 1100)
- Object2VEC（算法）(p. 1100)
- PCA（算法）(p. 1101)
- pyTorch (DLC) (p. 1101)
- 随机森林砍伐 (算法) (p. 1104)
- 射线 PyTorch (DLC) (p. 1104)
- Scikit-learn（算法）(p. 1105)
- 语义分割（算法）(p. 1105)
- Sq2Seq（算法）(p. 1106)
- Spark（算法）(p. 1106)
- SparkML 服务（算法）(p. 1106)
- Tensorflow (DLC) (p. 1107)
- Tensorflow 教练（DLC）(p. 1115)
- 张量流推断 (DLC) (p. 1116)
- Tensorflow Ray (DLC) (p. 1116)
- 大众（算法）(p. 1117)
- xgBoost（算法）(p. 1118)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='ap-
southeast-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='ap-
southeast-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/	4.1.0	训练、推理	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
sagemaker-chainer：*<tag>*				
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='ap-
southeast-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
834264404009.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
119527597002.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
972752614525.dkr.ecr.ap-southeast-1.amazonaws.com/	latest	调试器		

注册表路径	版本	Job 类型（图像范围）		
sagemaker-debugger-rules：*<tag>*				

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='ap-
southeast-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/image-classification：<tag>	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='ap-
southeast-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-neo-mxnet：<tag>	1.8	推断	inf	py3
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-neo-mxnet：<tag>	1.5	推断	inf	py3

Inferenia PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='ap-
southeast-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-neo-pytorch：<tag>	1.9	推断	inf	py3
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-neo-pytorch：<tag>	1.8	推断	inf	py3
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-neo-pytorch：<tag>	1.7	推断	inf	py3

K-Mean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='ap-
southeast-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/	1.4.1	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-training：*<tag>*				
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：<tag>	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：<tag>	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12-1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12-1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='ap-
southeast-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-<tag>	0.11-0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/	0.11-	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
sagemaker-rl-mxnet: 教练 0.11-*<tag>*				

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
245545462676.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='ap-
southeast-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-inference-mxnet：<tag>	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='ap-
southeast-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.8	推断	CPU，GPU	py3
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.7	推断	CPU，GPU	py3
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.6	推断	CPU，GPU	py3
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.5	推断	CPU，GPU	py3
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='ap-
southeast-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='ap-
southeast-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/pytorch-training：<tag>	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-pytorch：<tag>	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-pytorch：<tag>	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-pytorch：<tag>	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-pytorch：<tag>	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-pytorch：<tag>	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-pytorch：<tag>	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/randomcutforest：<tag>	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='ray-pytorch',region='ap-
southeast-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-ray-container训练:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='ap-
southeast-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Sq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='ap-southeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='ap-
southeast-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
759080221371.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
759080221371.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
759080221371.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='ap-southeast-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		

注册表路径	版本	Job 类型（图像范围）		
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='ap-
southeast-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.0.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	2.0.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	2.0.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.2	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.0.0	eia	CPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13-1	培训	CPU，GPU	py2
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-training：*<tag>*	1.13-1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13-0	eia	CPU	-
763104351884.dkr.ecr.ap-southeast-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13-0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12-0	eia	CPU	-
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12-0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12-0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11-0	eia	CPU	-
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11-0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.6.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='ap-
southeast-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-coach-container: 教练 -1.0.0-tf-<tag>	1.0.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-<tag>	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-	0.11.0	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.11.0-*<tag>*				
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='ap-
southeast-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
324986816169.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='ap-
southeast-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-TF-*<tag>*	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-*<tag>*	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.2-TF-*<tag>*	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='ap-
southeast-1',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-<tag>	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='ap-southeast-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-xgboost：<tag>	1.5-1	1.5.2	训练、推理	
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-xgboost：<tag>	1.3-1	1.3.3	训练、推理	
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-2	1.2.0	训练、推理	
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-1	1.2.0	训练、推理	
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-xgboost：<tag>	1.0-1	1.0	训练、推理	
475088953585.dkr.ecr.ap-southeast-1.amazonaws.com/xgboost：<tag>		1.5.2	训练、推理	
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	训练、推理	
121021644041.dkr.ecr.ap-southeast-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-1	0.90	训练、推理	

亚太地区 Docker 注册表路径和示例代码（悉尼）(ap-southeast-2)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1120)
- BlazingText（算法）(p. 1120)
- Chainer（DLC）(p. 1121)
- 澄清（算法）(p. 1121)
- 数据 Wrangler（算法）(p. 1121)
- 调试器（算法）(p. 1122)
- DeepAR 预测（算法）(p. 1122)
- 分解机器（算法）(p. 1122)
- 拥抱脸（算法）(p. 1122)
- 知识产权见解（算法）(p. 1125)
- 图像分类（算法）(p. 1126)
- 推理 mXNet (DLC) (p. 1126)
- 推理 PyTorch (DLC) (p. 1126)
- K-Mean（算法）(p. 1127)
- KNN（算法）(p. 1127)
- LDA（算法）(p. 1127)
- 线性学习者（算法）(p. 1128)
- MxNet (DLC) (p. 1128)
- MxNet 教练（DLC）(p. 1130)
- 模型监视器（算法）(p. 1131)
- NTM（算法）(p. 1131)
- Neo 图像分类（算法）(p. 1131)
- Neo MxNet (DLC) (p. 1132)
- 新 PyTorch (DLC) (p. 1132)
- Neo TensorFlow (DLC) (p. 1133)
- Neo xgBoost（算法）(p. 1133)
- 对象检测（算法）(p. 1134)
- Object2VEC（算法）(p. 1134)
- PCA（算法）(p. 1134)
- pyTorch (DLC) (p. 1134)
- 随机森林砍伐 (算法) (p. 1138)
- 射线 PyTorch (DLC) (p. 1138)
- Scikit-learn（算法）(p. 1138)
- 语义分割（算法）(p. 1139)
- Sq2Seq（算法）(p. 1139)
- Spark（算法）(p. 1139)
- SparkML 服务（算法）(p. 1140)
- Tensorflow (DLC) (p. 1140)
- Tensorflow 教练（DLC）(p. 1148)
- 张量流推断 (DLC) (p. 1149)
- Tensorflow Ray (DLC) (p. 1150)
- 大众（算法）(p. 1151)

- xgBoost（算法）(p. 1151)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='ap-
southeast-2',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
544295431143.dkr.ecr.ap-southeast-2.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='ap-
southeast-2',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='ap-
southeast-2',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
007051062584.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
422173101802.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-	1.x	处理		

注册表路径	版本	Job 类型（图像范围）		
data-wrangler-container：*<tag>*				

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
184798709955.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
514117268639.dkr.ecr.ap-southeast-2.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
712309505854.dkr.ecr.ap-southeast-2.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='ap-
southeast-2',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/	4.11.0	推断		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-inference：*<tag>*				
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
712309505854.dkr.ecr.ap-southeast-2.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
544295431143.dkr.ecr.ap-southeast-2.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='ap-
southeast-2',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='ap-
southeast-2',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Mean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
712309505854.dkr.ecr.ap-southeast-2.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
712309505854.dkr.ecr.ap-southeast-2.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
297031611018.dkr.ecr.ap-southeast-2.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
712309505854.dkr.ecr.ap-southeast-2.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='ap-
southeast-2',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet-eia：<tag>	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：<tag>	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：<tag>	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：<tag>	0.12-1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-mxnet：<tag>	0.12-1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='coach-mxnet',region='ap-
southeast-2',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11.1	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
563025443158.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
712309505854.dkr.ecr.ap-southeast-2.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='image-classification-neo',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/image-classification-neo：<tag>	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='ap-
southeast-2',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-inference-mxnet：<tag>	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='ap-
southeast-2',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.8	推断	CPU，GPU	py3
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.7	推断	CPU，GPU	py3
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
inference-pytorch：*<tag>*				
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='ap-
southeast-2',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
544295431143.dkr.ecr.ap-southeast-2.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
712309505854.dkr.ecr.ap-southeast-2.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
712309505854.dkr.ecr.ap-southeast-2.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='ap-
southeast-2',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.4.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.4.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.3.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.3.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.3.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.3.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.2.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.2.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.1.0	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference-eia：<tag>	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-inference：<tag>	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/pytorch-training：<tag>	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-pytorch：<tag>	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-pytorch：<tag>	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-pytorch：<tag>	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-pytorch：<tag>	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-pytorch：<tag>	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-pytorch：<tag>	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
712309505854.dkr.ecr.ap-southeast-2.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='ap-
southeast-2',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='ap-
southeast-2',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
544295431143.dkr.ecr.ap-southeast-2.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Sq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='ap-southeast-2')
```

注册表路径	版本	Job 类型（图像范围）		
544295431143.dkr.ecr.ap-southeast-2.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='ap-
southeast-2',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
440695851116.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
440695851116.dkr.ecr.ap-southeast-2.amazonaws.com/	3.0	处理		

注册表路径	版本	Job 类型（图像范围）		
sagemaker-spark-processing：<tag>				
440695851116.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-spark-processing：<tag>	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='ap-southeast-2',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.4	推断		
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='ap-
southeast-2',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	1.13.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference-eia：<tag>	1.12.0	eia	CPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	1.12.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.13.0	eia	CPU	-
763104351884.dkr.ecr.ap-southeast-2.amazonaws.com/tensorflow-inference：<tag>	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.12.0	eia	CPU	-
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.12.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='ap-
southeast-2',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-coach-container: 教练 -0.0-tf-*<tag>*	1.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='ap-
southeast-2',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
355873309152.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-neo-tensorflow：<tag>	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='ap-
southeast-2',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-ray-containerPay-1.6.t-<tag>	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-<tag>	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-ray-containerPay-0.8.TF-<tag>	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-<tag>	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-<tag>	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-<tag>	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-southeast-2.amazonaws.com/	0.5	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
sagemaker-rl-tensorflow:ray0.5-*<tag>*				

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='ap-
southeast-2',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-*<tag>*	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='ap-southeast-2',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-xgboost：*<tag>*	1.5-1	1.5.2	训练、推理	
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-xgboost：*<tag>*	1.3-1	1.3.3	训练、推理	
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-xgboost：*<tag>*	1.2-2	1.2.0	训练、推理	
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-xgboost：*<tag>*	1.2-1	1.2.0	训练、推理	
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/	1.0-1	1.0	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
sagemaker-xgboost：<tag>				
544295431143.dkr.ecr.ap-southeast-2.amazonaws.com/xgboost：<tag>	ap-	1.5.2	训练、推理	
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-xgboost：<tag>	ap-2	0.90	训练、推理	
783357654285.dkr.ecr.ap-southeast-2.amazonaws.com/sagemaker-xgboost：<tag>	ap-1	0.90	训练、推理	

亚太地区 Docker 注册表路径和示例代码 (ap-nornor-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1153)
- BlazingText（算法）(p. 1154)
- Chainer（DLC）(p. 1154)
- 澄清（算法）(p. 1154)
- 数据 Wrangler（算法）(p. 1155)
- 调试器（算法）(p. 1155)
- DeepAR 预测（算法）(p. 1155)
- 分解机器（算法）(p. 1156)
- 拥抱脸（算法）(p. 1156)
- 知识产权见解（算法）(p. 1159)
- 图像分类（算法）(p. 1159)
- 推理 mXNet (DLC) (p. 1159)
- 推理 PyTorch (DLC) (p. 1160)
- K-Meean（算法）(p. 1160)
- KNN（算法）(p. 1160)
- LDA（算法）(p. 1161)
- 线性学习者（算法）(p. 1161)
- MxNet (DLC) (p. 1161)
- MxNet 教练（DLC）(p. 1164)
- 模型监视器（算法）(p. 1164)
- NTM（算法）(p. 1165)
- Neo 图像分类 (算法) (p. 1165)
- Neo MxNet (DLC) (p. 1165)
- 新 PyTorch (DLC) (p. 1166)
- Neo TensorFlow (DLC) (p. 1166)
- Neo xgBoost（算法）(p. 1167)

- 对象检测（算法）(p. 1167)
- Object2VEC（算法）(p. 1167)
- PCA（算法）(p. 1168)
- pyTorch (DLC) (p. 1168)
- 随机森林砍伐 (算法) (p. 1171)
- 射线 PyTorch (DLC) (p. 1171)
- Scikit-learn（算法）(p. 1172)
- 语义分割（算法）(p. 1172)
- Sq2Seq (算法) (p. 1173)
- Spark（算法）(p. 1173)
- SparkML 服务（算法）(p. 1173)
- Tensorflow (DLC) (p. 1174)
- Tensorflow 教练（DLC）(p. 1182)
- 张量流推断 (DLC) (p. 1183)
- Tensorflow Ray (DLC) (p. 1183)
- 大众（算法）(p. 1184)
- xgBoost（算法）(p. 1185)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='ap-
northeast-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
501404015308.dkr.ecr.ap-northeast-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='ap-
northeast-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='ap-
northeast-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
377024640650.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
649008135260.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
430734990657.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
633353088612.dkr.ecr.ap-northeast-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
351501993468.dkr.ecr.ap-northeast-1.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='ap-
northeast-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/	4.12.3	推断		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-inference：*<tag>*				
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
351501993468.dkr.ecr.ap-northeast-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
501404015308.dkr.ecr.ap-northeast-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='ap-
northeast-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='ap-
northeast-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
351501993468.dkr.ecr.ap-northeast-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
351501993468.dkr.ecr.ap-northeast-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
258307448986.dkr.ecr.ap-northeast-1.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
351501993468.dkr.ecr.ap-northeast-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='ap-
northeast-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/	1.8.0	推断	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference：*<tag>*				
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：<tag>	1.8.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：<tag>	1.8.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='ap-
northeast-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-<tag>	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
574779866223.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
351501993468.dkr.ecr.ap-northeast-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='ap-
northeast-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='ap-
northeast-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='ap-
northeast-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-	2.4.2	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
inference-tensorflow：<tag>				
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-inference-tensorflow：<tag>	1.15-3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/xgboost-neo：<tag>	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
501404015308.dkr.ecr.ap-northeast-1.amazonaws.com/object-detection：<tag>	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
351501993468.dkr.ecr.ap-northeast-1.amazonaws.com/object2vec：<tag>	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
351501993468.dkr.ecr.ap-northeast-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='ap-
northeast-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.5.1	eia	CPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
351501993468.dkr.ecr.ap-northeast-1.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='ap-
northeast-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='ap-
northeast-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
501404015308.dkr.ecr.ap-northeast-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Sq2Seq (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='ap-northeast-1')
```

注册表路径	版本	Job 类型（图像范围）		
501404015308.dkr.ecr.ap-northeast-1.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='ap-
northeast-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
411782140378.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
411782140378.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
411782140378.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='ap-northeast-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		

注册表路径	版本	Job 类型（图像范围）		
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='ap-
northeast-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.3.0	eia	CPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.3	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.0.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	2.0.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	2.0.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.2	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.0.0	eia	CPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13-1	培训	CPU，GPU	py2
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	1.13-1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13-0	eia	CPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13-0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12-0	eia	CPU	-
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12-0	推断	CPU，GPU	-
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12-0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11-0	eia	CPU	-
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11-0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.6.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='ap-
northeast-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-coach-container: 教练 -1.0.0-TF-<tag>	1.0.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-<tag>	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-	0.11.0	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.11.0-*<tag>*				
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='ap-
northeast-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
941853720454.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='ap-
northeast-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-*<tag>*	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.2-*<tag>*	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='ap-
northeast-1',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-<tag>	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='ap-northeast-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-xgboost：<tag>	1.5-1	1.5.2	训练、推理	
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-xgboost：<tag>	1.3-1	1.3.3	训练、推理	
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-2	1.2.0	训练、推理	
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-1	1.2.0	训练、推理	
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-xgboost：<tag>	1.0-1	1.0	训练、推理	
501404015308.dkr.ecr.ap-northeast-1.amazonaws.com/xgboost：<tag>	1.5.2	1.5.2	训练、推理	
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	训练、推理	
354813040037.dkr.ecr.ap-northeast-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-1	0.90	训练、推理	

加拿大 Docker 注册表路径和示例代码 (ca-central-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1187)
- BlazingText（算法）(p. 1187)
- Chainer（DLC）(p. 1188)
- 澄清（算法）(p. 1188)
- 数据 Wrangler（算法）(p. 1188)
- 调试器（算法）(p. 1189)
- DeepAR 预测（算法）(p. 1189)
- 分解机器（算法）(p. 1189)
- 拥抱脸（算法）(p. 1189)
- 知识产权见解（算法）(p. 1192)
- 图像分类（算法）(p. 1193)
- 推理 mXNet (DLC) (p. 1193)
- 推断 PyTorch (DLC) (p. 1193)
- K-Mean（算法）(p. 1194)
- KNN（算法）(p. 1194)
- LDA（算法）(p. 1194)
- 线性学习者（算法）(p. 1195)
- MxNet (DLC) (p. 1195)
- MxNet 教练（DLC）(p. 1197)
- 模型监视器（算法）(p. 1198)
- NTM（算法）(p. 1198)
- Neo 图像分类（算法）(p. 1198)
- Neo MxNet (DLC) (p. 1199)
- 新 PyTorch (DLC) (p. 1199)
- Neo TensorFlow (DLC) (p. 1200)
- Neo xgBoost（算法）(p. 1200)
- 对象检测（算法）(p. 1201)
- Object2Vc（算法）(p. 1201)
- PCA（算法）(p. 1201)
- pyTorch (DLC) (p. 1201)
- 随机砍伐森林（算法）(p. 1205)
- 射线 PyTorch (DLC) (p. 1205)
- Scikit-learn（算法）(p. 1205)
- 语义分割（算法）(p. 1206)
- Seeq2Seq（算法）(p. 1206)
- Spark（算法）(p. 1206)
- SparkML 服务（算法）(p. 1207)
- Tensorflow (DLC) (p. 1207)
- Tensorflow 教练（DLC）(p. 1215)
- 张量流推断 (DLC) (p. 1216)
- Tensorflow Ray (DLC) (p. 1217)
- 大众（算法）(p. 1218)

- xgBoost（算法）(p. 1218)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='ca-
central-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='ca-
central-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='ca-
central-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
675030665977.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
557239378090.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-	1.x	处理		

注册表路径	版本	Job 类型（图像范围）		
data-wrangler-container：*<tag>*				

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
519511493484.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='ca-
central-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/	4.11.0	推断		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-inference：*<tag>*				
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.ca-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='ca-
central-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推断 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='ca-
central-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Mean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/linear-learner：<tag>	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='ca-
central-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ca-central-1.amazonaws.com/mxnet-training：<tag>	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ca-central-1.amazonaws.com/mxnet-inference：<tag>	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.ca-central-1.amazonaws.com/mxnet-training：<tag>	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.ca-central-1.amazonaws.com/mxnet-inference：<tag>	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.ca-central-1.amazonaws.com/mxnet-inference-eia：<tag>	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.ca-central-1.amazonaws.com/mxnet-training：<tag>	1.6.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-inference：<tag>	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-inference-eia：<tag>	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-training：<tag>	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet-serving：<tag>	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-inference：<tag>	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-inference-eia：<tag>	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet-serving：<tag>	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet-serving-eia：<tag>	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet-eia：<tag>	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='coach-mxnet',region='ca-
central-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
536280801234.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='image-classification-neo',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
464438896020.dkr.ecr.ca-central-1.amazonaws.com/image-classification-neo：<tag>	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='ca-
central-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-inference-mxnet：<tag>	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='ca-
central-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.8	推断	CPU，GPU	py3
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.7	推断	CPU，GPU	py3
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
inference-pytorch：*<tag>*				
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='ca-
central-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.1.2	推断	CPU，GPU	py3
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
464438896020.dkr.ecr.ca-central-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2Vc（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='ca-
central-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.11.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.11.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机砍伐森林（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='ca-
central-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='ca-
central-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
341280168497.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
341280168497.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Seeq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='ca-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
469771592824.dkr.ecr.ca-central-1.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='ca-
central-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
446299261295.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
446299261295.dkr.ecr.ca-central-1.amazonaws.com/	3.0	处理		

注册表路径	版本	Job 类型（图像范围）		
sagemaker-spark-processing：<tag>				
446299261295.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-spark-processing：<tag>	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='ca-central-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
341280168497.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.4	推断		
341280168497.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='ca-
central-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ap-northeast-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.0.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.0.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.0.2	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.0.0	eia	CPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
763104351884.dkr.ecr.ca-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.14.0	eia	CPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13.0	eia	CPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='ca-
central-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-coach-container: 教练 -1.0.0-TF-*<tag>*	1.0.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='ca-
central-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
464438896020.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-neo-tensorflow：<tag>	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='ca-
central-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-TF-<tag>	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-<tag>	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.2-<tag>	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-<tag>	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-<tag>	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-<tag>	0.5.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-<tag>	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='ca-
central-1',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-<tag>	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='ca-central-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
341280168497.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-xgboost：<tag>	1.5-1	1.5.2	训练、推理	
341280168497.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-xgboost：<tag>	1.3-1	1.3.3	训练、推理	
341280168497.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-2	1.2.0	训练、推理	
341280168497.dkr.ecr.ca-central-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-1	1.2.0	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
341280168497.dkr.ecr.cn-central-1.amazonaws.com/sagemaker-xgboost：<tag>	1.0-1	1.0	训练、推理	
469771592824.dkr.ecr.cn-central-1.amazonaws.com/xgboost：<tag>	1.5-1	1.5.2	训练、推理	
341280168497.dkr.ecr.cn-central-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	训练、推理	
341280168497.dkr.ecr.cn-central-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-1	0.90	训练、推理	

中国（北京）的 Docker 注册表路径和示例代码 (cn-north-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1220)
- BlazingText（算法）(p. 1220)
- Chainer（DLC）(p. 1221)
- 澄清（算法）(p. 1221)
- 数据 Wrangler（算法）(p. 1221)
- 调试器（算法）(p. 1222)
- DeepAR 预测（算法）(p. 1222)
- 分解机器（算法）(p. 1222)
- 拥抱脸（算法）(p. 1223)
- 知识产权见解（算法）(p. 1225)
- 图像分类（算法）(p. 1226)
- 推理 mXNet (DLC) (p. 1226)
- 推理 PyTorch (DLC) (p. 1226)
- K-Meard（算法）(p. 1227)
- KNN（算法）(p. 1227)
- 线性学习者（算法）(p. 1227)
- MxNet (DLC) (p. 1228)
- MxNet 教练（DLC）(p. 1230)
- 模型监视器（算法）(p. 1231)
- NTM（算法）(p. 1231)
- Neo 图像分类（算法）(p. 1231)
- Neo MxNet (DLC) (p. 1232)
- 新 PyTorch (DLC) (p. 1232)

- Neo TensorFlow (DLC) (p. 1233)
- Neo xgBoost (算法) (p. 1233)
- 对象检测 (算法) (p. 1234)
- Object2VEC (算法) (p. 1234)
- PCA (算法) (p. 1234)
- pyTorch (DLC) (p. 1234)
- 随机森林砍伐 (算法) (p. 1238)
- Scikit-learn (算法) (p. 1238)
- 语义分割 (算法) (p. 1238)
- Sq2Seq (算法) (p. 1238)
- Spark (算法) (p. 1239)
- SparkML 服务 (算法) (p. 1239)
- Tensorflow (DLC) (p. 1240)
- Tensorflow 教练 (DLC) (p. 1248)
- 张量流推断 (DLC) (p. 1249)
- Tensorflow Ray (DLC) (p. 1249)
- xgBoost (算法) (p. 1250)

AutoGluon (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='cn-
north-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型 (图像范围)		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/autogluon-inference：*<tag>*	0.4.0	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/autogluon-inference：*<tag>*	0.3.2	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='cn-
north-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='cn-
north-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
122526803553.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='data-wrangler',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
245909111842.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
618459771430.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/forecasting-deepar：*<tag>*	1	训练推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/	1	训练推理		

注册表路径	版本	Job 类型（图像范围）		
factorization-machines：<tag>				

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='cn-
north-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-training：<tag>	4.17.0	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-training：<tag>	4.17.0	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-inference：<tag>	4.17.0	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-inference：<tag>	4.17.0	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-training：<tag>	4.12.3	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-training：<tag>	4.12.3	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-inference：<tag>	4.12.3	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-inference：<tag>	4.12.3	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/	4.11.0	培训		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-training：*<tag>*				
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='cn-
north-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='cn-
north-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meard（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/knn：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='cn-
north-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/	1.5.1	eia	CPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：*<tag>*				
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	0.12.1	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='cn-
north-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11.1	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
453000072557.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='cn-
north-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='cn-
north-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='cn-
north-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='cn-
north-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	1.0.1-0	推断	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	1.0.1-0	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	1.0.0-2	推断	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	1.0.0-2	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	1.0.0-0	推断	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	1.0.0-0	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	0.4.1	推断	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	0.4.1	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	0.4.0	推断	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	0.4.0	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	0.3.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	1.8.1	培训	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	1.8.0	推断	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	1.8.0	培训	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	1.7.1	推断	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	1.7.1	培训	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	1.6.0	推断	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	1.6.0	培训	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	1.5.0	推断	CPU，GPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	1.5.0	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：<tag>	1.4.0	推断	CPU，GPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：<tag>	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-pytorch：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-pytorch：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/randomcutforest：*<tag>*	1	训练、推理		

Scikit-learn (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='cn-
north-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
450853457545.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练推理	
450853457545.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/semantic-segmentation：*<tag>*	1	训练推理		

Sq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='cn-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
390948362332.dkr.ecr.cn-north-1.amazonaws.com.cn/seq2seq：*<tag>*	1	训练推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='cn-
north-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
671472414489.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-spark-processing：*<tag>*	3.1	处理		
671472414489.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-spark-processing：*<tag>*	3.0	处理		
671472414489.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='cn-north-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
450853457545.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
450853457545.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='cn-
north-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.8.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.8.0	培训	CPU，GPU	py39
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/	2.5.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow-inference：*<tag>*				
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.4.3	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.4.3	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.3.2	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.3.2	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.3.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.4	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.3	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.3	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.2	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.2	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference-eia：<tag>	2.0.0	eia	CPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.15.5	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	1.15.5	培训	CPU，GPU	py3、py36、py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.15.4	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：<tag>	1.15.4	培训	CPU，GPU	py3、py36、py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference-eia：*<tag>*	1.14.0	eia	CPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow-scriptmode：*<tag>*	1.13.1	培训	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-training：*<tag>*	1.13-1	培训	CPU，GPU	py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow-serving-eia：*<tag>*	1.13-0	eia	CPU	-
727897471807.dkr.ecr.cn-north-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	1.13-0	推断	CPU，GPU	-
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow-serving-eia：*<tag>*	1.12-0	eia	CPU	-
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow-serving：*<tag>*	1.12-0	推断	CPU，GPU	-
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow-scriptmode：*<tag>*	1.12-0	培训	CPU，GPU	py2、py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow-serving-eia：*<tag>*	1.11-0	eia	CPU	-
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow-serving：*<tag>*	1.11-0	推断	CPU，GPU	-
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow-scriptmode：*<tag>*	1.11-0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.8.0	培训	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.7.0	推断	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.7.0	培训	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.6.0	推断	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.6.0	培训	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='cn-
north-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-rl-tensorflow: 教练 0.11.1-<tag>	0.11.1	培训	CPU，GPU	py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-rl-tensorflow: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-rl-tensorflow: 教练 0.11-<tag>	0.11	培训	CPU，GPU	py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-rl-	0.10.1	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.10.1-*<tag>*				
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='cn-
north-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
472730292857.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='cn-
north-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-rl-tensorflow: ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
422961961927.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-	0.6	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
rl-tensorflow: ray0.6-<tag>				
422961961927.dkr.ecr.cn- north-1.amazonaws.com.cn/ sagemaker- rl-tensorflow: ray0.5.3-<tag>	0.5.3	培训	CPU，GPU	py3
422961961927.dkr.ecr.cn- north-1.amazonaws.com.cn/ sagemaker- rl-tensorflow: ray0.5-<tag>	0.5	培训	CPU，GPU	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='cn-north-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
450853457545.dkr.ecr.cn- north-1.amazonaws.com.cn/ sagemaker- xgboost：<tag>	1.5-1	1.5.2	训练推理	
450853457545.dkr.ecr.cn- north-1.amazonaws.com.cn/ sagemaker- xgboost：<tag>	1.3-1	1.3.3	训练推理	
450853457545.dkr.ecr.cn- north-1.amazonaws.com.cn/ sagemaker- xgboost：<tag>	1.2-2	1.2.0	训练推理	
450853457545.dkr.ecr.cn- north-1.amazonaws.com.cn/ sagemaker- xgboost：<tag>	1.2-1	1.2.0	训练推理	
450853457545.dkr.ecr.cn- north-1.amazonaws.com.cn/ sagemaker- xgboost：<tag>	1-1	1.0	训练推理	
390948362332.dkr.ecr.cn- north-1.amazonaws.com.cn/ xgboost：<tag>		1.5.2	训练推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
450853457545.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-xgboost：<tag>	0.90-2	0.90	训练推理	
450853457545.dkr.ecr.cn-north-1.amazonaws.com.cn/sagemaker-xgboost：<tag>	0.90-1	0.90	训练推理	

中国（宁夏）的 Docker 注册表路径和示例代码 (cn-northwest-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1252)
- BlazingText（算法）(p. 1252)
- Chainer（DLC）(p. 1253)
- 澄清（算法）(p. 1253)
- 数据 Wrangler（算法）(p. 1253)
- 调试器（算法）(p. 1254)
- DeepAR 预测（算法）(p. 1254)
- 分解机器（算法）(p. 1254)
- 拥抱脸（算法）(p. 1254)
- 知识产权见解（算法）(p. 1257)
- 图像分类（算法）(p. 1258)
- 推理 mXNet (DLC) (p. 1258)
- 推理 PyTorch (DLC) (p. 1258)
- K-Meensor（算法）(p. 1259)
- KNN（算法）(p. 1259)
- 线性学习者（算法）(p. 1259)
- MxNet (DLC) (p. 1260)
- MxNet 教练（DLC）(p. 1262)
- 模型监视器（算法）(p. 1263)
- NTM（算法）(p. 1263)
- Neo 图像分类（算法）(p. 1263)
- Neo MxNet (DLC) (p. 1263)
- 新 PyTorch (DLC) (p. 1264)
- Neo TensorFlow (DLC) (p. 1265)
- Neo xgBoost（算法）(p. 1265)
- 对象检测（算法）(p. 1265)
- Object2VEC（算法）(p. 1265)
- PCA（算法）(p. 1266)
- pyTorch (DLC) (p. 1266)

- 随机森林砍伐 (算法) (p. 1269)
- Scikit-learn (算法) (p. 1269)
- 语义分割 (算法) (p. 1270)
- Sq2Seq (算法) (p. 1270)
- Spark (算法) (p. 1270)
- SparkML 服务 (算法) (p. 1271)
- Tensorflow (DLC) (p. 1271)
- Tensorflow 教练 (DLC) (p. 1280)
- 张量流推断 (DLC) (p. 1280)
- Tensorflow Ray (DLC) (p. 1281)
- xgBoost (算法) (p. 1281)

AutoGluon (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='cn-
northwest-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型 (图像范围)		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/autogluon-inference：*<tag>*	0.4.0	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/autogluon-inference：*<tag>*	0.3.2	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='cn-northwest-1')
```

注册表路径	版本	Job 类型 (图像范围)		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/blazingtext：*<tag>*	1	训练推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='cn-
northwest-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-chainer：*<tag>*	5.0.0	训练推理	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-chainer：*<tag>*	4.1.0	训练推理	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-chainer：*<tag>*	4.0.0	训练推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='cn-
northwest-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
122578899357.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
249157047649.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-	1.x	处理		

注册表路径	版本	Job 类型（图像范围）		
data-wrangler-container：*<tag>*				

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
658757709296.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='cn-
northwest-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/	4.11.0	推断		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-inference：*<tag>*				
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.6.1	培训		

注册表路径	版本	Job 类型（图像范围）		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/ipinsights：*<tag>*	1	训练推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/image-classification：*<tag>*	1	训练推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='cn-
northwest-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='cn-
northwest-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meensor（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/kmeans：*<tag>*	1	训练推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/knn：*<tag>*	1	训练推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/linear-learner：*<tag>*	1	训练推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='cn-
northwest-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/	1.4.1	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-training：*<tag>*				
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：<tag>	1.1.0	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：<tag>	1.1.0	推断	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：<tag>	1.0.0	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：<tag>	1.0.0	推断	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：<tag>	0.12.1	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-mxnet：<tag>	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='cn-
northwest-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-rl-mxnet: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/	0.11-	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
sagemaker-rl-mxnet: 教练 0.11-*<tag>*				

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
453252182341.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/ntm：*<tag>*	1	训练推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='cn-
northwest-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='cn-
northwest-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='cn-
northwest-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/object-detection：*<tag>*	1	训练推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/object2vec：*<tag>*	1	训练推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pca：*<tag>*	1	训练推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='cn-
northwest-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.0-1	推断	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.0-1	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.10-2	推断	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.10-2	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/pytorch-training：<tag>	1.2.0	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-pytorch：<tag>	1.1.0	推断	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-pytorch：<tag>	1.1.0	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-pytorch：<tag>	1.0.0	推断	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-pytorch：<tag>	1.0.0	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-pytorch：<tag>	0.4.0	推断	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-pytorch：<tag>	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/randomcutforest：<tag>	1	训练推理		

Scikit-learn (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='sklearn',region='cn-
northwest-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/semantic-segmentation：*<tag>*	1	训练、推理		

Sq2Seq (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='cn-northwest-1')
```

注册表路径	版本	Job 类型（图像范围）		
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='cn-
northwest-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
844356804704.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-spark-processing：*<tag>*	3.1	处理		
844356804704.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-spark-processing：*<tag>*	3.0	处理		
844356804704.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='cn-northwest-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='cn-
northwest-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.8.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/	2.8.0	培训	CPU，GPU	py39

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow-training：*<tag>*				
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.2	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.2	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.13.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.12.2	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	1.12.2	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.12.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.4	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.3	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.3	培训	CPU，GPU	py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.2	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.2	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference-eia：<tag>	2.0.0	eia	CPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.15.5	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	1.15.5	培训	CPU，GPU	py3、py36、py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.15.4	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	1.15.4	培训	CPU，GPU	py3、py36、py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow-scriptmode：<tag>	1.13.1	培训	CPU，GPU	py2
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow-serving-eia：<tag>	1.13.0	eia	CPU	-
727897471807.dkr.ecr.cn-northwest-1.amazonaws.com.cn/tensorflow-inference：<tag>	1.13.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.9.0	推断	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.9.0	培训	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.8.0	推断	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-tensorflow：*<tag>*	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='cn-
northwest-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='inferentia-tensorflow',region='cn-
northwest-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
474822919863.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='cn-
northwest-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
423003514399.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-rl-tensorflow:ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='cn-northwest-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-xgboost：<tag>	1.5-1	1.5.2	训练、推理	
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-xgboost：<tag>	1.3-1	1.3.3	训练、推理	
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-xgboost：<tag>	1.2-2	1.2.0	训练、推理	
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-xgboost：<tag>	1.2-1	1.2.0	训练、推理	
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-xgboost：<tag>	1.0-1	1.0	训练、推理	
387376663083.dkr.ecr.cn-northwest-1.amazonaws.com.cn/xgboost：<tag>	1.5-2	1.5.2	训练、推理	
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-xgboost：<tag>	0.90-2	0.90	训练、推理	
451049120500.dkr.ecr.cn-northwest-1.amazonaws.com.cn/sagemaker-xgboost：<tag>	0.90-1	0.90	训练、推理	

欧洲 Docker 注册表路径和示例代码（法兰克福）(eu-central-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1283)
- BlazingText（算法）(p. 1284)
- Chainer（DLC）(p. 1284)
- 澄清（算法）(p. 1285)
- 数据 Wrangler（算法）(p. 1285)

- 调试器（算法）(p. 1285)
- DeepAR 预测（算法）(p. 1286)
- 分解机器（算法）(p. 1286)
- 拥抱脸（算法）(p. 1286)
- 知识产权见解（算法）(p. 1289)
- 图像分类（算法）(p. 1289)
- 推理 mXNet (DLC) (p. 1290)
- 推理 PyTorch (DLC) (p. 1290)
- K-Mean（算法）(p. 1291)
- KNN（算法）(p. 1291)
- LDA（算法）(p. 1291)
- 线性学习者（算法）(p. 1291)
- MxNet (DLC) (p. 1292)
- MxNet 教练（DLC）(p. 1294)
- 模型监视器（算法）(p. 1295)
- NTM（算法）(p. 1295)
- Neo 图像分类（算法）(p. 1295)
- Neo MxNet (DLC) (p. 1296)
- 新 PyTorch (DLC) (p. 1296)
- Neo TensorFlow (DLC) (p. 1297)
- Neo xgBoost（算法）(p. 1297)
- 对象检测（算法）(p. 1298)
- Object2VEC（算法）(p. 1298)
- PCA（算法）(p. 1298)
- pyTorch (DLC) (p. 1298)
- 随机砍伐森林（算法）(p. 1302)
- 射线 PyTorch (DLC) (p. 1302)
- Scikit-learn（算法）(p. 1302)
- 语义分割（算法）(p. 1303)
- Seq2Seq（算法）(p. 1303)
- Spark（算法）(p. 1303)
- SparkML 服务（算法）(p. 1304)
- Tensorflow (DLC) (p. 1304)
- Tensorflow 教练（DLC）(p. 1312)
- 张量流推断 (DLC) (p. 1313)
- Tensorflow Ray (DLC) (p. 1314)
- 大众（算法）(p. 1315)
- xgBoost（算法）(p. 1315)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='autogluon',region='eu-
central-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
813361260812.dkr.ecr.eu-central-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='eu-
central-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-chainer：<tag>	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-chainer：<tag>	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-chainer：<tag>	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='eu-
central-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
017069133835.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-clarify-processing：<tag>	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
024640144536.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-data-wrangler-container：<tag>	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='debugger',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
482524230118.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
495149712605.dkr.ecr.eu-central-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
664544806723.dkr.ecr.eu-central-1.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='eu-
central-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.eu-central-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
664544806723.dkr.ecr.eu-central-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='image-classification',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
813361260812.dkr.ecr.eu-central-1.amazonaws.com/image-classification：<tag>	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='eu-
central-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-neo-mxnet：<tag>	1.8	推断	inf	py3
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-neo-mxnet：<tag>	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='eu-
central-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-neo-pytorch：<tag>	1.9	推断	inf	py3
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-neo-pytorch：<tag>	1.8	推断	inf	py3
746233611703.dkr.ecr.eu-central-1.amazonaws.com/	1.7	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
sagemaker-neo-pytorch：*<tag>*				

K-Mean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
664544806723.dkr.ecr.eu-central-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
664544806723.dkr.ecr.eu-central-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
353608530281.dkr.ecr.eu-central-1.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
664544806723.dkr.ecr.eu-central-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='eu-
central-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/	1.5.1	eia	CPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：*<tag>*				
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12-1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12-1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='eu-
central-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
048819808253.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
664544806723.dkr.ecr.eu-central-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
746233611703.dkr.ecr.eu-central-1.amazonaws.com/image-classification-neo：<tag>	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='eu-
central-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-inference-mxnet：<tag>	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='eu-
central-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.8	推断	CPU，GPU	py3
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.7	推断	CPU，GPU	py3
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='eu-
central-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
746233611703.dkr.ecr.eu-central-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
813361260812.dkr.ecr.eu-central-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
664544806723.dkr.ecr.eu-central-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
664544806723.dkr.ecr.eu-central-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='eu-
central-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：<tag>	1.11.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：<tag>	1.11.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：<tag>	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：<tag>	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：<tag>	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：<tag>	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：<tag>	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：<tag>	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：<tag>	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：<tag>	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：<tag>	1.8.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机砍伐森林（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
664544806723.dkr.ecr.eu-central-1.amazonaws.com/randomcutforest：<tag>	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='eu-
central-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-火炬<tag>	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-<tag>	0.8.5	培训	CPU，GPU	py36

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='eu-
central-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
492215442770.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-scikit-learn：<tag>	0.23-1	0.23.2	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
492215442770.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
813361260812.dkr.ecr.eu-central-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Seq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='eu-central-1')
```

注册表路径	版本	Job 类型（图像范围）		
813361260812.dkr.ecr.eu-central-1.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='eu-
central-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
906073651304.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
906073651304.dkr.ecr.eu-central-1.amazonaws.com/	3.0	处理		

注册表路径	版本	Job 类型（图像范围）		
sagemaker-spark-processing：<tag>				
906073651304.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-spark-processing：<tag>	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='eu-central-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
492215442770.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.4	推断		
492215442770.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='eu-
central-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.1.0	eia	CPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.14.0	eia	CPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13.0	eia	CPU	-
763104351884.dkr.ecr.eu-central-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='eu-
central-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-coach-container: 教练 -1.0.0-TF-*<tag>*	1.0.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='eu-
central-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
746233611703.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='eu-
central-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-*<tag>*	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.TF-*<tag>*	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-central-1.amazonaws.com/	0.5	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
sagemaker-rl-tensorflow:ray0.5-*<tag>*				

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='eu-
central-1',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-*<tag>*	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='eu-central-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
492215442770.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.5-1	1.5.2	训练、推理	
492215442770.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.3-1	1.3.3	训练、推理	
492215442770.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.2-2	1.2.0	训练、推理	
492215442770.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.2-1	1.2.0	训练、推理	
492215442770.dkr.ecr.eu-central-1.amazonaws.com/	1.0-1	1.0	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
sagemaker-xgboost：<tag>				
813361260812.dkr.ecr.eu-central-1.amazonaws.com/xgboost：<tag>	eu-	1.5.2	训练、推理	
492215442770.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-xgboost：<tag>	eu-90-2	0.90	训练、推理	
492215442770.dkr.ecr.eu-central-1.amazonaws.com/sagemaker-xgboost：<tag>	eu-90-1	0.90	训练、推理	

欧洲 Docker 注册表路径和示例代码（爱尔兰）(eu-west-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1317)
- BlazingText（算法）(p. 1318)
- Chainer（DLC）(p. 1318)
- 澄清（算法）(p. 1318)
- 数据 Wrangler（算法）(p. 1319)
- 调试器（算法）(p. 1319)
- DeepAR 预测（算法）(p. 1319)
- 分解机器（算法）(p. 1320)
- 拥抱脸（算法）(p. 1320)
- 知识产权见解（算法）(p. 1323)
- 图像分类（算法）(p. 1323)
- 推理 mXNet (DLC) (p. 1323)
- 推理 PyTorch (DLC) (p. 1324)
- K-Meean（算法）(p. 1324)
- KNN（算法）(p. 1324)
- LDA（算法）(p. 1325)
- 线性学习者（算法）(p. 1325)
- MxNet (DLC) (p. 1325)
- MxNet 教练（DLC）(p. 1328)
- 模型监视器（算法）(p. 1328)
- NTM（算法）(p. 1329)
- Neo 图像分类（算法）(p. 1329)
- Neo MxNet (DLC) (p. 1329)
- 新 PyTorch (DLC) (p. 1330)
- Neo TensorFlow (DLC) (p. 1330)
- Neo xgBoost（算法）(p. 1331)

- 对象检测（算法）(p. 1331)
- Object2Vc（算法）(p. 1331)
- PCA（算法）(p. 1332)
- pyTorch (DLC) (p. 1332)
- 随机森林砍伐 (算法) (p. 1335)
- 射线 PyTorch (DLC) (p. 1335)
- Scikit-learn（算法）(p. 1336)
- 语义分割（算法）(p. 1336)
- Sq2Seq（算法）(p. 1337)
- Spark（算法）(p. 1337)
- SparkML 服务（算法）(p. 1337)
- Tensorflow (DLC) (p. 1338)
- Tensorflow 教练（DLC）(p. 1346)
- 张量流推断 (DLC) (p. 1347)
- Tensorflow Ray (DLC) (p. 1347)
- 大众（算法）(p. 1348)
- xgBoost（算法）(p. 1349)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='eu-
west-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
685385470294.dkr.ecr.eu-west-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='eu-
west-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='eu-
west-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
131013547314.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
245179582081.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
929884845733.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
224300973850.dkr.ecr.eu-west-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
438346466558.dkr.ecr.eu-west-1.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='eu-
west-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/	4.12.3	推断		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-inference：*<tag>*				
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
438346466558.dkr.ecr.eu-west-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
685385470294.dkr.ecr.eu-west-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='eu-
west-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='eu-
west-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
438346466558.dkr.ecr.eu-west-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
438346466558.dkr.ecr.eu-west-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
999678624901.dkr.ecr.eu-west-1.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
438346466558.dkr.ecr.eu-west-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='eu-
west-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-1.amazonaws.com/	1.8.0	推断	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference：*<tag>*				
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/mxnet-inference-eia：<tag>	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet-serving：<tag>	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet-serving-eia：<tag>	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet-eia：<tag>	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.1.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.6.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.6.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='eu-
west-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-<tag>	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
468650794304.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
438346466558.dkr.ecr.eu-west-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
802834080501.dkr.ecr.eu-west-1.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='eu-
west-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='eu-
west-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='eu-
west-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-	2.4.2	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
inference-tensorflow：*<tag>*				
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15-3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
802834080501.dkr.ecr.eu-west-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
685385470294.dkr.ecr.eu-west-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2Vc（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
438346466558.dkr.ecr.eu-west-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
438346466558.dkr.ecr.eu-west-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='eu-
west-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1-0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1-0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.10-2	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-training：*<tag>*	1.10-2	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.10-0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-training：*<tag>*	1.10-0	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.5.1	eia	CPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
438346466558.dkr.ecr.eu-west-1.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='eu-
west-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-ray-container训练:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='eu-
west-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
685385470294.dkr.ecr.eu-west-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Sq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='eu-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
685385470294.dkr.ecr.eu-west-1.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='eu-
west-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
571004829621.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
571004829621.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
571004829621.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='eu-west-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		

注册表路径	版本	Job 类型（图像范围）		
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='eu-
west-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.4.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.4.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.4.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.3.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.3.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	2.3.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	2.3.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.2	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.0.0	eia	CPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-1.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13-1	培训	CPU，GPU	py2
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.13-1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13-0	eia	CPU	-
763104351884.dkr.ecr.eu-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13-0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12-0	eia	CPU	-
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12-0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12-0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11-0	eia	CPU	-
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11-0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.11.0	eia	CPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.11.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.11.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.6.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='eu-
west-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-coach-container: 教练 -1.0.0-tf-<tag>	1.0.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-<tag>	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-	0.11.0	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.11.0-*<tag>*				
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='eu-
west-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.0	推断	inf	py3
802834080501.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='eu-
west-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-ray-containerFay-1.6.0-<tag>	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-<tag>	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-ray-containerPay-0.8.2-<tag>	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-<tag>	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-<tag>	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-<tag>	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-<tag>	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='eu-
west-1',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-*<tag>*	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='eu-west-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.5-1	1.5.2	训练、推理	
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.3-1	1.3.3	训练、推理	
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.2-2	1.2.0	训练、推理	
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.2-1	1.2.0	训练、推理	
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-xgboost：*<tag>*	1.0-1	1.0	训练、推理	
685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost：*<tag>*	1.5.2	1.5.2	训练、推理	
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-xgboost：*<tag>*	0.90-2	0.90	训练、推理	
141502667606.dkr.ecr.eu-west-1.amazonaws.com/sagemaker-xgboost：*<tag>*	0.90-1	0.90	训练、推理	

欧洲 Docker 注册表路径和示例代码（伦敦）(eu-west-2)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1351)
- BlazingText（算法）(p. 1351)
- Chainer（DLC）(p. 1352)
- 澄清（算法）(p. 1352)
- 数据 Wrangler（算法）(p. 1352)
- 调试器（算法）(p. 1353)
- DeepAR 预测（算法）(p. 1353)
- 分解机器（算法）(p. 1353)
- 拥抱脸（算法）(p. 1353)
- 知识产权见解（算法）(p. 1356)
- 图像分类（算法）(p. 1357)
- 推理 mXNet (DLC) (p. 1357)
- 推理 PyTorch (DLC) (p. 1357)
- K-Mean（算法）(p. 1358)
- KNN（算法）(p. 1358)
- LDA（算法）(p. 1358)
- 线性学习者（算法）(p. 1359)
- MxNet (DLC) (p. 1359)
- MxNet 教练（DLC）(p. 1361)
- 模型监视器（算法）(p. 1362)
- NTM（算法）(p. 1362)
- Neo 图像分类（算法）(p. 1362)
- Neo MxNet (DLC) (p. 1363)
- 新 PyTorch (DLC) (p. 1363)
- Neo TensorFlow (DLC) (p. 1364)
- Neo xgBoost（算法）(p. 1364)
- 对象检测（算法）(p. 1365)
- Object2VEC（算法）(p. 1365)
- PCA（算法）(p. 1365)
- pyTorch (DLC) (p. 1365)
- 随机森林砍伐 (算法) (p. 1368)
- 射线 PyTorch (DLC) (p. 1369)
- Scikit-learn (算法) (p. 1369)
- 语义分割（算法）(p. 1370)
- Seq2Seq (算法) (p. 1370)
- Spark（算法）(p. 1370)
- SparkML 服务（算法）(p. 1371)
- Tensorflow (DLC) (p. 1371)
- Tensorflow 教练（DLC）(p. 1379)
- 张量流推断 (DLC) (p. 1380)
- Tensorflow Ray (DLC) (p. 1381)
- 大众（算法）(p. 1381)

- xgBoost（算法）(p. 1382)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='eu-
west-2',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='eu-
west-2',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='eu-
west-2',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
440796970383.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
894491911112.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-	1.x	处理		

注册表路径	版本	Job 类型（图像范围）		
data-wrangler-container：*<tag>*				

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
250201462417.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='eu-
west-2',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/	4.11.0	推断		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-inference：*<tag>*				
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.eu-west-2.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='eu-
west-2',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='eu-
west-2',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Mean（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='eu-
west-2',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-2.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-2.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-2.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-2.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.eu-west-2.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.5.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-mxnet-eia：<tag>	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.6.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-mxnet：<tag>	1.6.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='coach-mxnet',region='eu-
west-2',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
749857270468.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='image-classification-neo',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
205493899709.dkr.ecr.eu-west-2.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='eu-
west-2',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='eu-
west-2',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
inference-pytorch：*<tag>*				
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='eu-
west-2',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
205493899709.dkr.ecr.eu-west-2.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='eu-
west-2',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.11.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.11.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='randomcutforest',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

射线 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-pytorch',region='eu-
west-2',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-*<tag>*	1.6.0	培训	CPU，GPU	py36
462105765813.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-火炬-*<tag>*	0.8.5	培训	CPU，GPU	py36

Scikit-learn (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='eu-
west-2',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Seq2Seq (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='eu-west-2')
```

注册表路径	版本	Job 类型（图像范围）		
644912444149.dkr.ecr.eu-west-2.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='eu-
west-2',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
836651553127.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
836651553127.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
836651553127.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='eu-west-2',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='eu-
west-2',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.eu-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.0.1	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.14.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.14.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.14.0	eia	CPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.13.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.14.0	eia	CPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13.0	eia	CPU	-
763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-west-2.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.11.0	eia	CPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='eu-
west-2',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-coach-containerPach-1.0.0-TF-<tag>	1.0.0	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='eu-
west-2',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.0	推断	inf	py3
205493899709.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='eu-
west-2',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
462105765813.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-ray-container:ray-1.6.0-TF-*<tag>*	1.6.0	培训	CPU，GPU	py37
462105765813.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.5-tf-*<tag>*	0.8.5	培训	CPU，GPU	py36
462105765813.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-ray-container:ray-0.8.2-TF-*<tag>*	0.8.2	培训	CPU，GPU	py36
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

大众（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='vw',region='eu-
west-2',version='8.7.0',image_scope='training')
```

注册表路径	版本	Job 类型（图像范围）		
462105765813.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-rl-vw-container:vw-8.7.0-<tag>	8.7.0	培训		

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='eu-west-2',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-xgboost：<tag>	eu5-1	1.5.2	训练、推理	
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-xgboost：<tag>	eu3-1	1.3.3	训练、推理	
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-xgboost：<tag>	eu2-2	1.2.0	训练、推理	
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-xgboost：<tag>	eu2-1	1.2.0	训练、推理	
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-xgboost：<tag>	eu0-1	1.0	训练、推理	
644912444149.dkr.ecr.eu-west-2.amazonaws.com/xgboost：<tag>	eu-	1.5.2	训练、推理	
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-xgboost：<tag>	eu90-2	0.90	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
764974769150.dkr.ecr.eu-west-2.amazonaws.com/sagemaker-xgboost：*<tag>*	0.90-1	0.90	训练、推理	

欧洲 Docker 注册表路径和示例代码（巴黎）(eu-west-3)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1384)
- BlazingText（算法）(p. 1384)
- Chainer（DLC）(p. 1385)
- 澄清（算法）(p. 1385)
- 数据 Wrangler（算法）(p. 1385)
- 调试器（算法）(p. 1386)
- DeepAR 预测（算法）(p. 1386)
- 分解机器（算法）(p. 1386)
- 拥抱脸（算法）(p. 1387)
- 知识产权见解（算法）(p. 1389)
- 图像分类（算法）(p. 1390)
- 推理 mXNet (DLC) (p. 1390)
- 推理 PyTorch (DLC) (p. 1390)
- K-Meensor（算法）(p. 1391)
- KNN（算法）(p. 1391)
- 线性学习者（算法）(p. 1391)
- MxNet (DLC) (p. 1392)
- MxNet 教练（DLC）(p. 1394)
- 模型监视器（算法）(p. 1395)
- NTM（算法）(p. 1395)
- Neo 图像分类 (算法) (p. 1395)
- Neo MxNet (DLC) (p. 1396)
- 新 PyTorch (DLC) (p. 1396)
- Neo TensorFlow (DLC) (p. 1397)
- Neo xgBoost（算法）(p. 1397)
- 对象检测（算法）(p. 1398)
- Object2VEC (算法) (p. 1398)
- PCA（算法）(p. 1398)
- pyTorch (DLC) (p. 1398)
- 随机森林砍伐 (算法) (p. 1401)
- Scikit-learn (算法) (p. 1402)
- 语义分割（算法）(p. 1402)
- Sq2Seq (算法) (p. 1402)

- Spark（算法）(p. 1403)
- SparkML 服务（算法）(p. 1403)
- Tensorflow (DLC) (p. 1403)
- Tensorflow 教练（DLC）(p. 1412)
- 张量流推断 (DLC) (p. 1413)
- Tensorflow Ray (DLC) (p. 1413)
- xgBoost（算法）(p. 1414)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='eu-
west-3',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/blazingtext：*<tag>*	1	训练推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='eu-
west-3',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='eu-
west-3',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
341593696636.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='data-wrangler',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
807237891255.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
447278800020.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/	1	训练、推理		

注册表路径	版本	Job 类型（图像范围）		
factorization-machines：<tag>				

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='eu-
west-3',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-training：<tag>	4.17.0	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-training：<tag>	4.17.0	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-inference：<tag>	4.17.0	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.17.0	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/	4.11.0	培训		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-training：<tag>				
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-training：<tag>	4.11.0	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.eu-west-3.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/ipinsights：*<tag>*	1	训练推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/image-classification：*<tag>*	1	训练推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='eu-
west-3',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='eu-
west-3',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meensor（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/kmeans：*<tag>*	1	训练推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/knn：*<tag>*	1	训练推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/linear-learner：*<tag>*	1	训练推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='eu-
west-3',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-3.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/	1.5.1	eia	CPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：*<tag>*				
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.eu-west-3.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.eu-west-3.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet-eia：<tag>	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：<tag>	1.6.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：<tag>	1.6.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='eu-
west-3',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
680080141114.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/ntm：*<tag>*	1	训练推理		

Neo 图像分类 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
254080097072.dkr.ecr.eu-west-3.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='eu-
west-3',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='eu-
west-3',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.5	推断	CPU，GPU	py3
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='eu-
west-3',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-inference-tensorflow：<tag>	2.4.2	推断	CPU，GPU	py3
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-inference-tensorflow：<tag>	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
254080097072.dkr.ecr.eu-west-3.amazonaws.com/xgboost-neo：<tag>	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/object-detection：*<tag>*	1	训练推理		

Object2VEC (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/object2vec：*<tag>*	1	训练推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/pca：*<tag>*	1	训练推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='eu-
west-3',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：<tag>	1.11.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：<tag>	1.11.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：<tag>	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：<tag>	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：<tag>	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：<tag>	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：<tag>	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：<tag>	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：<tag>	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：<tag>	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：<tag>	1.8.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='randomcutforest',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

Scikit-learn (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='eu-
west-3',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Sq2Seq (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='eu-west-3')
```

注册表路径	版本	Job 类型（图像范围）		
749696950732.dkr.ecr.eu-west-3.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='eu-
west-3',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
136845547031.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
136845547031.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
136845547031.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='eu-west-3',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='eu-
west-3',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.1.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.1.2	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.4.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.4.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.3.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：*<tag>*	2.2.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference-eia：<tag>	2.0.0	eia	CPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13.0	eia	CPU	-
763104351884.dkr.ecr.eu-west-3.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='eu-
west-3',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-<tag>	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-<tag>	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-<tag>	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-rl-	0.10	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.10-<tag>				

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='eu-
west-3',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-neo-tensorflow：<tag>	2.5.2	推断	inf	py3
254080097072.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-neo-tensorflow：<tag>	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='eu-
west-3',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-<tag>	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-<tag>	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-	0.5.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
rl-tensorflow:ray0.5.3-<tag>				
520713654638.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-<tag>	0.5	培训	CPU，GPU	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='eu-west-3',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-xgboost：<tag>	1.5-1	1.5.2	训练、推理	
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-xgboost：<tag>	1.3-1	1.3.3	训练、推理	
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-xgboost：<tag>	1.2-2	1.2.0	训练、推理	
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-xgboost：<tag>	1.2-1	1.2.0	训练、推理	
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-xgboost：<tag>	1.0-1	1.0	训练、推理	
749696950732.dkr.ecr.eu-west-3.amazonaws.com/xgboost：<tag>	1.5-1	1.5.2	训练、推理	
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
659782779980.dkr.ecr.eu-west-3.amazonaws.com/sagemaker-xgboost：*<tag>*	0.90-1	0.90	训练、推理	

欧洲 Docker 注册表路径和示例代码（斯德哥尔摩）(eu-north-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1416)
- BlazingText（算法）(p. 1416)
- Chainer（DLC）(p. 1417)
- 澄清（算法）(p. 1417)
- 数据 Wrangler（算法）(p. 1417)
- 调试器（算法）(p. 1418)
- DeepAR 预测（算法）(p. 1418)
- 分解机器（算法）(p. 1418)
- 拥抱脸（算法）(p. 1419)
- 知识产权见解（算法）(p. 1421)
- 图像分类（算法）(p. 1422)
- 推理 mXNet (DLC) (p. 1422)
- 推理 PyTorch (DLC) (p. 1422)
- K-Mees（算法）(p. 1423)
- KNN（算法）(p. 1423)
- 线性学习者（算法）(p. 1423)
- MxNet (DLC) (p. 1424)
- MxNet 教练（DLC）(p. 1426)
- 模型监视器（算法）(p. 1427)
- NTM（算法）(p. 1427)
- Neo 图像分类（算法）(p. 1427)
- Neo MxNet (DLC) (p. 1428)
- 新 PyTorch (DLC) (p. 1428)
- Neo TensorFlow (DLC) (p. 1429)
- Neo xgBoost（算法）(p. 1429)
- 对象检测（算法）(p. 1430)
- Object2Vc（算法）(p. 1430)
- PCA（算法）(p. 1430)
- pyTorch (DLC) (p. 1430)
- 随机砍伐森林（算法）(p. 1434)
- Scikit-learn（算法）(p. 1434)
- 语义分割（算法）(p. 1434)
- Sq2Sq（算法）(p. 1434)

- Spark（算法）(p. 1435)
- SparkML 服务（算法）(p. 1435)
- Tensorflow (DLC) (p. 1436)
- Tensorflow 教练（DLC）(p. 1444)
- 张量流推断 (DLC) (p. 1445)
- Tensorflow Ray (DLC) (p. 1445)
- xgBoost（算法）(p. 1446)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='eu-
north-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/blazingtext：<tag>	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='eu-
north-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-chainer：<tag>	5.0.0	训练推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-chainer：<tag>	4.1.0	训练推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-chainer：<tag>	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='eu-
north-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
763603941244.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-clarify-processing：<tag>	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='data-wrangler',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
054986407534.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
314864569078.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/	1	训练推理		

注册表路径	版本	Job 类型（图像范围）		
factorization-machines：<tag>				

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='eu-
north-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.17.0	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.17.0	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.17.0	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.17.0	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/	4.11.0	培训		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-training：<tag>				
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.11.0	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.eu-north-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='eu-
north-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='eu-
north-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Mees（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='eu-
north-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-north-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/	1.5.1	eia	CPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：*<tag>*				
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.eu-north-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.eu-north-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet-eia：<tag>	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：<tag>	1.6.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：<tag>	1.6.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='eu-
north-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-<tag>	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
895015795356.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-model-monitor-analyzer：<tag>		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/ntm：<tag>	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
601324751636.dkr.ecr.eu-north-1.amazonaws.com/image-classification-neo：<tag>	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='eu-
north-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-inference-mxnet：<tag>	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='eu-
north-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.8	推断	CPU，GPU	py3
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.7	推断	CPU，GPU	py3
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='eu-
north-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
601324751636.dkr.ecr.eu-north-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2Vc（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='eu-
north-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：*<tag>*	1.11.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：*<tag>*	1.11.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：<tag>	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：<tag>	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：<tag>	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：<tag>	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：<tag>	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：<tag>	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：<tag>	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：<tag>	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：<tag>	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：<tag>	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：<tag>	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference-eia：*<tag>*	1.3.1	eia	CPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机砍伐森林（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/randomcutforest：*<tag>*	1	训练推理		

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='eu-
north-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
662702820516.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练推理	
662702820516.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练推理		

Sq2Sq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='eu-north-1')
```

注册表路径	版本	Job 类型（图像范围）		
669576153137.dkr.ecr.eu-north-1.amazonaws.com/seq2seq：<tag>	1	训练推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='eu-
north-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
330188676905.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-spark-processing：<tag>	3.1	处理		
330188676905.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-spark-processing：<tag>	3.0	处理		
330188676905.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-spark-processing：<tag>	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='eu-north-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
662702820516.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.4	推断		
662702820516.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-sparkml-serving：<tag>	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='eu-
north-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.eu-north-1.amazonaws.com/	2.5.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow-inference：*<tag>*				
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.4.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：<tag>	2.4.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.4.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：<tag>	2.4.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.3.4	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：<tag>	2.3.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.3.3	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：<tag>	2.3.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.2	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.0.0	eia	CPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.15.0	eia	CPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	1.14.0	eia	CPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13.1	培训	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13.0	eia	CPU	-
763104351884.dkr.ecr.eu-north-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.11.0	eia	CPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.11.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.11.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.6.0	培训	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.5.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='eu-
north-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-rl-	0.10.1	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.10.1-*<tag>*				
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-*<tag>*	0.10	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='eu-
north-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
601324751636.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='eu-
north-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-rl-tensorflow: ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-	0.6	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
rl-tensorflow: ray0.6-*<tag>*				
520713654638.dkr.ecr.eu-north-1.amazonaws.com/ sagemaker- rl-tensorflow: ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-north-1.amazonaws.com/ sagemaker- rl-tensorflow: ray0.5-*<tag>*	0.5	培训	CPU，GPU	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='eu-north-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
662702820516.dkr.ecr.eu-north-1.amazonaws.com/ sagemaker- xgboost：*<tag>*	1.5-1	1.5.2	训练推理	
662702820516.dkr.ecr.eu-north-1.amazonaws.com/ sagemaker- xgboost：*<tag>*	1.3-1	1.3.3	训练推理	
662702820516.dkr.ecr.eu-north-1.amazonaws.com/ sagemaker- xgboost：*<tag>*	1.2-2	1.2.0	训练推理	
662702820516.dkr.ecr.eu-north-1.amazonaws.com/ sagemaker- xgboost：*<tag>*	1.2-1	1.2.0	训练推理	
662702820516.dkr.ecr.eu-north-1.amazonaws.com/ sagemaker- xgboost：*<tag>*	1.0-1	1.0	训练推理	
669576153137.dkr.ecr.eu-north-1.amazonaws.com/ xgboost：*<tag>*	1.5.2	1.5.2	训练推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
662702820516.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	训练推理	
662702820516.dkr.ecr.eu-north-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-1	0.90	训练推理	

欧洲 Docker 注册表路径和示例代码（米兰）(eu-south-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1448)
- BlazingText（算法）(p. 1448)
- Chainer（DLC）(p. 1449)
- 澄清（算法）(p. 1449)
- 数据 Wrangler（算法）(p. 1449)
- 调试器（算法）(p. 1450)
- DeepAR 预测（算法）(p. 1450)
- 分解机器（算法）(p. 1450)
- 拥抱脸（算法）(p. 1451)
- 知识产权见解（算法）(p. 1453)
- 图像分类（算法）(p. 1454)
- 推理 mXNet (DLC) (p. 1454)
- 推理 PyTorch (DLC) (p. 1454)
- K-Mees（算法）(p. 1455)
- KNN（算法）(p. 1455)
- 线性学习者（算法）(p. 1455)
- MxNet (DLC) (p. 1456)
- MxNet 教练（DLC）(p. 1458)
- 模型监视器（算法）(p. 1459)
- NTM（算法）(p. 1459)
- Neo 图像分类（算法）(p. 1459)
- Neo MxNet (DLC) (p. 1460)
- 新 PyTorch (DLC) (p. 1460)
- Neo TensorFlow (DLC) (p. 1461)
- Neo xgBoost（算法）(p. 1461)
- 对象检测（算法）(p. 1462)
- Object2VEC（算法）(p. 1462)
- PCA（算法）(p. 1462)
- pyTorch (DLC) (p. 1462)
- 随机森林砍伐（算法）(p. 1465)

- Scikit-learn（算法）(p. 1466)
- 语义分割（算法）(p. 1466)
- Sq2Seq（算法）(p. 1466)
- Spark（算法）(p. 1467)
- SparkML 服务（算法）(p. 1467)
- Tensorflow (DLC) (p. 1467)
- Tensorflow 教练（DLC）(p. 1476)
- 张量流推断 (DLC) (p. 1477)
- Tensorflow Ray (DLC) (p. 1477)
- xgBoost（算法）(p. 1478)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='eu-
south-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='eu-
south-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='eu-
south-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
638885417683.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='data-wrangler',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
488287956546.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
563282790590.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/	1	训练推理		

注册表路径	版本	Job 类型（图像范围）		
factorization-machines：<tag>				

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='eu-
south-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.17.0	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.17.0	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.17.0	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.17.0	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.12.3	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.12.3	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.12.3	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.12.3	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/	4.11.0	培训		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-training：<tag>				
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.11.0	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.11.0	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.11.0	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
692866216735.dkr.ecr.eu-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='eu-
south-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='eu-
south-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Mees（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='eu-
south-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
692866216735.dkr.ecr.eu-south-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
692866216735.dkr.ecr.eu-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
692866216735.dkr.ecr.eu-south-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
692866216735.dkr.ecr.eu-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
692866216735.dkr.ecr.eu-south-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
692866216735.dkr.ecr.eu-south-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
692866216735.dkr.ecr.eu-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
692866216735.dkr.ecr.eu-south-1.amazonaws.com/	1.5.1	eia	CPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：*<tag>*				
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
692866216735.dkr.ecr.eu-south-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
692866216735.dkr.ecr.eu-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
692866216735.dkr.ecr.eu-south-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet-eia：<tag>	1.3.0	eia	CPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：<tag>	1.2.1	推断	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：<tag>	1.4.0	推断	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：<tag>	1.6.0	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：<tag>	1.6.0	推断	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12-1	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-mxnet：<tag>	0.12-1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='eu-
south-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-<tag>	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
933208885752.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-model-monitor-analyzer：<tag>		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/ntm：<tag>	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
966458181534.dkr.ecr.eu-south-1.amazonaws.com/image-classification-neo：<tag>	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='eu-
south-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-inference-mxnet：<tag>	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='eu-
south-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.8	推断	CPU，GPU	py3
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.7	推断	CPU，GPU	py3
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-inference-pytorch：<tag>	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='eu-
south-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
966458181534.dkr.ecr.eu-south-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='eu-
south-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.4.1-0	推断	CPU，GPU	py38
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.4.1-0	培训	CPU，GPU	py38
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.4.0-2	推断	CPU，GPU	py38
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.4.0-2	培训	CPU，GPU	py38
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.4.0-0	推断	CPU，GPU	py38
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.4.0-0	培训	CPU，GPU	py38
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.3.1	推断	CPU，GPU	py38
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.3.1	培训	CPU，GPU	py38
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.3.0	推断	CPU，GPU	py38
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.3.0	培训	CPU，GPU	py38
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.2.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.8.1	培训	CPU，GPU	py3、py36
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.8.0	推断	CPU，GPU	py3、py36
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.8.0	培训	CPU，GPU	py3、py36
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.7.1	推断	CPU，GPU	py3、py36
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.7.1	培训	CPU，GPU	py3、py36
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.6.0	推断	CPU，GPU	py3、py36
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.6.0	培训	CPU，GPU	py3、py36
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.5.0	推断	CPU，GPU	py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.5.0	培训	CPU，GPU	py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：<tag>	1.4.0	推断	CPU，GPU	py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：<tag>	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='randomcutforest',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/randomcutforest：*<tag>*	1	训练推理		

Scikit-learn（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='eu-
south-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练推理	
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练推理		

Sq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='eu-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
257386234256.dkr.ecr.eu-south-1.amazonaws.com/seq2seq：*<tag>*	1	训练推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='eu-
south-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
753923664805.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
753923664805.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
753923664805.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='eu-south-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='eu-
south-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py38
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py38
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.5.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.4.2	培训	CPU，GPU	py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py2、py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.4.0	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.4.0	培训	CPU，GPU	py2、py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.0.0	eia	CPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py2、py3
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.5	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.5	培训	CPU，GPU	py3、py36、py37
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.4	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.4	培训	CPU，GPU	py3、py36、py37
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.13.1	培训	CPU，GPU	py2
692866216735.dkr.ecr.ap-south-1.amazonaws.com/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.13.0	eia	CPU	-
692866216735.dkr.ecr.eu-south-1.amazonaws.com/tensorflow-inference：<tag>	1.13.0	推断	CPU，GPU	-
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.12.0	eia	CPU	-
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.12.0	推断	CPU，GPU	-
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.12.0	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.11.0	eia	CPU	-
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.11.0	推断	CPU，GPU	-
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.11.0	培训	CPU，GPU	py2、py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow-eia：<tag>	1.10.0	eia	CPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	推断	CPU，GPU	py2
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	培训	CPU，GPU	py2
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	推断	CPU，GPU	py2
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	培训	CPU，GPU	py2
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	推断	CPU，GPU	py2
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
048378556238.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='eu-
south-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-<tag>	0.11.1	培训	CPU，GPU	py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-<tag>	0.11	培训	CPU，GPU	py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-<tag>	0.10.1	培训	CPU，GPU	py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-rl-	0.10	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.10-<tag>				

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='eu-
south-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-neo-tensorflow：<tag>	2.5.2	推断	inf	py3
966458181534.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-neo-tensorflow：<tag>	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='eu-
south-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-<tag>	0.6.5	培训	CPU，GPU	py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-<tag>	0.6	培训	CPU，GPU	py3
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-	0.5.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
rl-tensorflow:ray0.5.3-<tag>				
048378556238.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-<tag>	0.5	培训	CPU，GPU	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='eu-south-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-xgboost：<tag>	eu5-1	1.5.2	训练推理	
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-xgboost：<tag>	eu3-1	1.3.3	训练推理	
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-xgboost：<tag>	eu2-2	1.2.0	训练推理	
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-xgboost：<tag>	eu2-1	1.2.0	训练推理	
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-xgboost：<tag>	eu0-1	1.0	训练推理	
257386234256.dkr.ecr.eu-south-1.amazonaws.com/xgboost：<tag>	eu-	1.5.2	训练推理	
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-xgboost：<tag>	eu90-2	0.90	训练推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
978288397137.dkr.ecr.eu-south-1.amazonaws.com/sagemaker-xgboost：*<tag>*	0.90-1	0.90	训练推理	

中东（巴林）Docker 注册表路径和示例代码 (me-south-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1480)
- BlazingText（算法）(p. 1480)
- Chainer（DLC）(p. 1481)
- 澄清（算法）(p. 1481)
- 数据 Wrangler（算法）(p. 1481)
- 调试器（算法）(p. 1482)
- DeepAR 预测（算法）(p. 1482)
- 分解机器（算法）(p. 1482)
- 拥抱脸（算法）(p. 1483)
- 知识产权见解（算法）(p. 1485)
- 图像分类（算法）(p. 1486)
- 推理 mXNet (DLC) (p. 1486)
- 推理 PyTorch (DLC) (p. 1486)
- K-Meensor（算法）(p. 1487)
- KNN（算法）(p. 1487)
- 线性学习者（算法）(p. 1487)
- MxNet (DLC) (p. 1488)
- MxNet 教练（DLC）(p. 1490)
- 模型监视器（算法）(p. 1491)
- NTM（算法）(p. 1491)
- Neo 图像分类（算法）(p. 1491)
- Neo MxNet (DLC) (p. 1492)
- 新 PyTorch (DLC) (p. 1492)
- Neo TensorFlow (DLC) (p. 1493)
- Neo xgBoost（算法）(p. 1493)
- 对象检测（算法）(p. 1494)
- Object2VEC（算法）(p. 1494)
- PCA（算法）(p. 1494)
- pyTorch (DLC) (p. 1494)
- 随机森林砍伐 (算法) (p. 1497)
- Scikit-learn (算法) (p. 1498)
- 语义分割（算法）(p. 1498)
- Sq2Seq（算法）(p. 1498)

- Spark（算法）(p. 1499)
- SparkML 服务（算法）(p. 1499)
- Tensorflow (DLC) (p. 1499)
- Tensorflow 教练（DLC）(p. 1508)
- 张量流推断 (DLC) (p. 1509)
- Tensorflow Ray (DLC) (p. 1509)
- xgBoost（算法）(p. 1510)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='me-
south-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
217643126080.dkr.ecr.me-south-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/blazingtext：<tag>		训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='me-south-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-chainer：<tag>	5.0.0	训练、推理	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-chainer：<tag>	4.1.0	训练、推理	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-chainer：<tag>	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='me-south-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
835444307964.dkr.ecr.me-south-1.amazonaws.com/sagemaker-clarify-processing：<tag>	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='data-wrangler',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
376037874950.dkr.ecr.me-south-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
986000313247.dkr.ecr.me-south-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/forecasting-deepar：*<tag>*	1	推理、训练		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/	1	推理、训练		

注册表路径	版本	Job 类型（图像范围）		
factorization-machines：<tag>				

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='me-
south-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.17.0	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.17.0	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.17.0	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.17.0	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.12.3	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.12.3	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.12.3	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.12.3	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/	4.11.0	培训		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-training：<tag>				
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.11.0	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.11.0	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.11.0	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
217643126080.dkr.ecr.me-south-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/ipinsights：*<tag>*		训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/image-classification：*<tag>*		训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='me-
south-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.6	推断	inf	py3
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='me-
south-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meensor（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='me-
south-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
217643126080.dkr.ecr.me-south-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/	1.5.1	eia	CPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：*<tag>*				
724002660598.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
217643126080.dkr.ecr.ap-south-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
724002660598.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
217643126080.dkr.ecr.ap-south-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
217643126080.dkr.ecr.ap-south-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
724002660598.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
724002660598.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
724002660598.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12.1	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='me-
south-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-<tag>	0.11.0	培训	CPU，GPU	py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-<tag>	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
607024016150.dkr.ecr.me-south-1.amazonaws.com/sagemaker-model-monitor-analyzer：<tag>		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/ntm：<tag>	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
836785723513.dkr.ecr.me-south-1.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='me-
south-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='me-
south-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='me-
south-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.1.0	推断	CPU，GPU	py3
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
836785723513.dkr.ecr.me-south-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/object-detection：*<tag>*		训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/object2vec：*<tag>*		训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/pca：*<tag>*		训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='me-
south-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：<tag>	1.10.0	推断	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：<tag>	1.10.0	培训	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：<tag>	1.10.2	推断	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：<tag>	1.10.2	培训	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：<tag>	1.10.0	推断	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：<tag>	1.10.0	培训	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：<tag>	1.9.1	推断	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：<tag>	1.9.1	培训	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：<tag>	1.9.0	推断	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：<tag>	1.9.0	培训	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：<tag>	1.8.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='randomcutforest',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/randomcutforest：*<tag>*	1	推理、训练		

Scikit-learn (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='me-
south-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	推理、训练	
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	推理、训练	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/semantic-segmentation：*<tag>*	1	推理、训练		

Sq2Seq（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='me-south-1')
```

注册表路径	版本	Job 类型（图像范围）		
249704162688.dkr.ecr.me-south-1.amazonaws.com/seq2seq：*<tag>*	1	推理、训练		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='me-
south-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
750251592176.dkr.ecr.me-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
750251592176.dkr.ecr.me-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
750251592176.dkr.ecr.me-south-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='me-south-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='me-
south-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py39
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.3	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.3	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.2	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.2	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.1	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.3.0	eia	CPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：<tag>	2.6.2	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：<tag>	2.7.1	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：<tag>	2.8.3	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：<tag>	2.8.3	培训	CPU，GPU	py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：<tag>	2.8.4	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.0	培训	CPU，GPU	py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.0	培训	CPU，GPU	py2、py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.0	培训	CPU，GPU	py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.14.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13.1	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.1	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.1	培训	CPU，GPU	py2、py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.0.0	eia	CPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.0	培训	CPU，GPU	py2、py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.5	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.5	培训	CPU，GPU	py3、py36、py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.4	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.4	培训	CPU，GPU	py3、py36、py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：*<tag>*	1.15.3	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：*<tag>*	1.15.3	培训	CPU，GPU	py2、py3、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.13.1	培训	CPU，GPU	py2
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.12.0	eia	CPU	-
217643126080.dkr.ecr.me-south-1.amazonaws.com/tensorflow-inference：<tag>	1.12.0	推断	CPU，GPU	-
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.12.0	eia	CPU	-
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.12.0	推断	CPU，GPU	-
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.12.0	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.11.0	eia	CPU	-
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.11.0	推断	CPU，GPU	-
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.11.0	培训	CPU，GPU	py2、py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow-eia：<tag>	1.10.0	eia	CPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	推断	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	培训	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	推断	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	培训	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	推断	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.5.0	培训	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	推断	CPU，GPU	py2
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='me-
south-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-rl-	0.10	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.10-<tag>				

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='me-
south-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-neo-tensorflow：<tag>	2.5.2	推断	inf	py3
836785723513.dkr.ecr.me-south-1.amazonaws.com/sagemaker-neo-tensorflow：<tag>	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='me-
south-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-<tag>	0.6.5	培训	CPU，GPU	py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-<tag>	0.6	培训	CPU，GPU	py3
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-	0.5.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
rl-tensorflow:ray0.5.3-<tag>				
724002660598.dkr.ecr.me-south-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-<tag>	0.5	培训	CPU，GPU	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='me-south-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-xgboost：<tag>	1.5-1	1.5.2	推理、训练	
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-xgboost：<tag>	1.3-1	1.3.3	推理、训练	
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-2	1.2.0	推理、训练	
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-1	1.2.0	推理、训练	
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-xgboost：<tag>	1.0-1	1.0	推理、训练	
249704162688.dkr.ecr.me-south-1.amazonaws.com/xgboost：<tag>	1	1.5.2	推理、训练	
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	推理、训练	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
801668240914.dkr.ecr.me-south-1.amazonaws.com/sagemaker-xgboost：*<tag>*	0.90-1	0.90	推理、训练	

南美洲（圣保罗）Docker 注册表路径和示例代码 (sa-east-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法） (p. 1512)
- BlazingText（算法） (p. 1512)
- Chainer（DLC） (p. 1513)
- 澄清（算法） (p. 1513)
- 数据 Wrangler（算法） (p. 1513)
- 调试器（算法） (p. 1514)
- DeepAR 预测（算法） (p. 1514)
- 分解机器（算法） (p. 1514)
- 拥抱脸（算法） (p. 1515)
- 知识产权见解（算法） (p. 1517)
- 图像分类（算法） (p. 1518)
- 推理 mXNet (DLC) (p. 1518)
- 推理 PyTorch (DLC) (p. 1518)
- K-Meensor（算法） (p. 1519)
- KNN（算法） (p. 1519)
- 线性学习者（算法） (p. 1519)
- MxNet (DLC) (p. 1520)
- MxNet 教练（DLC） (p. 1522)
- 模型监视器（算法） (p. 1523)
- NTM（算法） (p. 1523)
- Neo 图像分类 (算法) (p. 1523)
- Neo MxNet (DLC) (p. 1524)
- 新 PyTorch (DLC) (p. 1524)
- Neo TensorFlow (DLC) (p. 1525)
- Neo xgBoost（算法） (p. 1525)
- 对象检测（算法） (p. 1526)
- Object2VEC (算法) (p. 1526)
- PCA（算法） (p. 1526)
- pyTorch (DLC) (p. 1526)
- 随机森林砍伐 (算法) (p. 1529)
- Scikit-learn (算法) (p. 1530)
- 语义分割（算法） (p. 1530)
- Sq2Seq (算法) (p. 1530)

- Spark（算法）(p. 1531)
- SparkML 服务（算法）(p. 1531)
- Tensorflow (DLC) (p. 1531)
- Tensorflow 教练（DLC）(p. 1540)
- 张量流推断 (DLC) (p. 1541)
- Tensorflow Ray (DLC) (p. 1541)
- xgBoost（算法）(p. 1542)

AutoGluon（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='sa-
east-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='sa-east-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='sa-east-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
520018980103.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-clarify-processing：*<tag>*	1.0	处理		

数据 Wrangler（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='data-wrangler',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
424196993095.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-data-wrangler-container：*<tag>*	1.x	处理		

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
818342061345.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/	1	训练、推理		

注册表路径	版本	Job 类型（图像范围）		
factorization-machines：<tag>				

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='sa-
east-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.17.0	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.17.0	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.17.0	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.17.0	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.12.3	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.12.3	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/	4.11.0	培训		

注册表路径	版本	Job 类型（图像范围）		
huggingface-pytorch-training：<tag>				
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.11.0	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.11.0	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-training：<tag>	4.10.2	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-inference：<tag>	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
763104351884.dkr.ecr.sa-east-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='sa-
east-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='sa-
east-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Meensor（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='sa-
east-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.sa-east-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.sa-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
763104351884.dkr.ecr.sa-east-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.sa-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.sa-east-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
763104351884.dkr.ecr.sa-east-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.sa-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.sa-east-1.amazonaws.com/	1.5.1	eia	CPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：*<tag>*				
520713654638.dkr.ecr.eu-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.eu-east-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.eu-east-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
763104351884.dkr.ecr.eu-east-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
763104351884.dkr.ecr.eu-east-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-east-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-east-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3
520713654638.dkr.ecr.eu-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.eu-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12.1	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-mxnet：*<tag>*	0.12.1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='sa-
east-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3

模型监视器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='model-monitor',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
539772159869.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-model-monitor-analyzer：*<tag>*		监控		

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
756306329178.dkr.ecr.sa-east-1.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='sa-
east-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='sa-
east-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.8	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='sa-
east-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.2	推断	CPU，GPU	py3
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
756306329178.dkr.ecr.sa-east-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='sa-
east-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.11.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.11.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.7.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.7.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
```

```
image_uris.retrieve(framework='randomcutforest',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

Scikit-learn (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='sa-
east-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Sq2Seq (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='sa-east-1')
```

注册表路径	版本	Job 类型（图像范围）		
855470959533.dkr.ecr.sa-east-1.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='sa-
east-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
737130764395.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
737130764395.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
737130764395.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='sa-east-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.2	推断		

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='sa-
east-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.sa-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.8.0	推断	CPU，GPU	-
763104351884.dkr.ecr.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.8.0	培训	CPU，GPU	py39
763104351884.dkr.ecr.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.7.1	培训	CPU，GPU	py38
763104351884.dkr.ecr.sa-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.7.0	推断	CPU，GPU	-
763104351884.dkr.ecr.sa-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.3	推断	CPU，GPU	-
763104351884.dkr.ecr.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.3	培训	CPU，GPU	py38
763104351884.dkr.ecr.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.2	培训	CPU，GPU	py38
763104351884.dkr.ecr.sa-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.6.0	推断	CPU，GPU	-
763104351884.dkr.ecr.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.6.0	培训	CPU，GPU	py38
763104351884.dkr.ecr.sa-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.5.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.1	培训	CPU，GPU	py37
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.5.0	培训	CPU，GPU	py37
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.3	推断	CPU，GPU	-
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.3	培训	CPU，GPU	py37
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.4.1	推断	CPU，GPU	-
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.4.1	培训	CPU，GPU	py37
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.2	推断	CPU，GPU	-
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.2	培训	CPU，GPU	py37
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-inference：*<tag>*	2.3.1	推断	CPU，GPU	-
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-training：*<tag>*	2.3.1	培训	CPU，GPU	py37
763104351884.dkr.ec2.sa-east-1.amazonaws.com/tensorflow-inference-eia：*<tag>*	2.3.0	eia	CPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-inference：<tag>	2.2.1	推断	CPU，GPU	-
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-training：<tag>	2.2.1	培训	CPU，GPU	py37
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-inference：<tag>	2.2.0	推断	CPU，GPU	-
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-training：<tag>	2.2.0	培训	CPU，GPU	py37
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.1.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.1.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.0.4	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.0.3	培训	CPU，GPU	py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.0.2	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.0.0	eia	CPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	1.15.5	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	1.15.5	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	1.15.4	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	1.15.4	培训	CPU，GPU	py3、py36、py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	1.14.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.13.1	培训	CPU，GPU	py2
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training：<tag>	1.13.1	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.13.0	eia	CPU	-
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference：<tag>	1.13.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.12.0	eia	CPU	-
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.12.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.12.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving-eia：<tag>	1.11.0	eia	CPU	-
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-serving：<tag>	1.11.0	推断	CPU，GPU	-
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-scriptmode：<tag>	1.11.0	培训	CPU，GPU	py2、py3
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow-eia：<tag>	1.10.0	eia	CPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.10.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.9.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.us-east-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.5.0	培训	CPU，GPU	py2
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	推断	CPU，GPU	py2
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.4.1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='sa-
east-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-*<tag>*	0.11.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-*<tag>*	0.11.0	培训	CPU，GPU	py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-*<tag>*	0.11	培训	CPU，GPU	py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-*<tag>*	0.10.1	培训	CPU，GPU	py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-rl-	0.10	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
tensorflow: 教练 0.10-<tag>				

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='sa-
east-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-neo-tensorflow：<tag>	2.1.2	推断	inf	py3
756306329178.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-neo-tensorflow：<tag>	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='sa-
east-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-<tag>	0.6.5	培训	CPU，GPU	py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-<tag>	0.6	培训	CPU，GPU	py3
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-	0.5.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
rl-tensorflow:ray0.5.3-<tag>				
520713654638.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-<tag>	0.5	培训	CPU，GPU	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='sa-east-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-xgboost：<tag>	1.5-1	1.5.2	训练、推理	
737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-xgboost：<tag>	1.3-1	1.3.3	训练、推理	
737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-2	1.2.0	训练、推理	
737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-xgboost：<tag>	1.2-1	1.2.0	训练、推理	
737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-xgboost：<tag>	1.0-1	1.0	训练、推理	
855470959533.dkr.ecr.sa-east-1.amazonaws.com/xgboost：<tag>	1.5-1	1.5.2	训练、推理	
737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-2	0.90	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
737474898029.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboost：*<tag>*	0.90-1	0.90	训练、推理	

Docker 注册表路径和示例代码Amazon GovCloud (美国西部) (us-gov-west-1)

以下主题列出了 Amazon SageMaker 提供的此区域中每种算法和深度学习容器的参数。

主题

- AutoGluon（算法）(p. 1544)
- BlazingText（算法）(p. 1544)
- Chainer（DLC）(p. 1545)
- 澄清（算法）(p. 1545)
- 调试器（算法）(p. 1546)
- DeepAR 预测（算法）(p. 1546)
- 分解机器（算法）(p. 1546)
- 拥抱脸（算法）(p. 1547)
- 知识产权见解（算法）(p. 1550)
- 图像分类（算法）(p. 1550)
- 推理 mXNet (DLC) (p. 1551)
- 推理 PyTorch (DLC) (p. 1551)
- K-Mees（算法）(p. 1551)
- KNN（算法）(p. 1552)
- LDA（算法）(p. 1552)
- 线性学习者（算法）(p. 1552)
- MxNet (DLC) (p. 1553)
- MxNet 教练（DLC）(p. 1556)
- NTM（算法）(p. 1556)
- Neo 图像分类（算法）(p. 1557)
- Neo MxNet (DLC) (p. 1557)
- 新 PyTorch (DLC) (p. 1557)
- Neo TensorFlow (DLC) (p. 1558)
- Neo xgBoost（算法）(p. 1559)
- 对象检测（算法）(p. 1559)
- Object2VEC（算法）(p. 1559)
- PCA（算法）(p. 1559)
- pyTorch (DLC) (p. 1560)
- 随机森林砍伐 (算法) (p. 1563)
- Scikit-learn (算法) (p. 1564)
- 语义分割（算法）(p. 1564)
- Sq2Seq (算法) (p. 1564)
- Spark（算法）(p. 1565)
- SparkML 服务（算法）(p. 1565)

- Tensorflow (DLC) (p. 1566)
- Tensorflow 教练（ DLC ）(p. 1576)
- 张量流推断 (DLC) (p. 1577)
- Tensorflow Ray (DLC) (p. 1577)
- xgBoost（ 算法 ）(p. 1578)

AutoGluon（ 算法 ）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='autogluon',region='us-gov-
west-1',image_scope='inference',version='0.4')
```

注册表路径	版本	Job 类型（ 图像范围 ）		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/autogluon-training：*<tag>*	0.4.0	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/autogluon-inference：*<tag>*	0.4.0	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/autogluon-training：*<tag>*	0.3.2	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.2	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/autogluon-training：*<tag>*	0.3.1	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/autogluon-inference：*<tag>*	0.3.1	推断		

BlazingText（ 算法 ）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='blazingtext',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/blazingtext：*<tag>*	1	训练、推理		

Chainer（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='chainer',region='us-gov-
west-1',version='5.0.0',py_version='py3',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-chainer：*<tag>*	5.0.0	训练、推理	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.1.0	训练、推理	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-chainer：*<tag>*	4.0.0	训练、推理	CPU，GPU	py2、py3

澄清（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='clarify',region='us-gov-
west-1',version='1.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
598674086554.dkr.ecr.us-gov-west-1.amazonaws.com/	1.0	处理		

注册表路径	版本	Job 类型（图像范围）		
sagemaker-clarify-processing：*<tag>*				

调试器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='debugger',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
515509971035.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-debugger-rules：*<tag>*	latest	调试器		

DeepAR 预测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='forecasting-deepar',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/forecasting-deepar：*<tag>*	1	训练、推理		

分解机器（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='factorization-machines',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/factorization-machines：*<tag>*	1	训练、推理		

拥抱脸（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='huggingface',region='us-gov-
west-1',version='4.4.2',image_scope='training',base_framework_version='tensorflow2.4.1')
```

注册表路径	版本	Job 类型（图像范围）		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.17.0	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.17.0	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.17.0	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.17.0	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.12.3	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.12.3	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.12.3	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.12.3	推断		

注册表路径	版本	Job 类型（图像范围）		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.11.0	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.11.0	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.11.0	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.11.0	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.10.2	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.10.2	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		

注册表路径	版本	Job 类型（图像范围）		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.10.2	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.10.2	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.6.1	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.6.1	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-inference：*<tag>*	4.6.1	推断		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-inference：*<tag>*	4.6.1	推断		

注册表路径	版本	Job 类型（图像范围）		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.5.0	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.5.0	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-pytorch-training：*<tag>*	4.4.2	培训		
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/huggingface-tensorflow-training：*<tag>*	4.4.2	培训		

知识产权见解（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ipinsights',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/ipinsights：*<tag>*	1	训练、推理		

图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/image-classification：*<tag>*	1	训练、推理		

推理 mXNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-mxnet',region='us-gov-
west-1',version='1.5.1',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.8	推断	inf	py3
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-neo-mxnet：*<tag>*	1.5.1	推断	inf	py3

推理 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-pytorch',region='us-gov-
west-1',version='1.9',py_version='py3')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.9	推断	inf	py3
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.8	推断	inf	py3
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-neo-pytorch：*<tag>*	1.7	推断	inf	py3

K-Mees（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='kmeans',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/kmeans：*<tag>*	1	训练、推理		

KNN（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='knn',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/knn：*<tag>*	1	训练、推理		

LDA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='lda',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/lda：*<tag>*	1	训练、推理		

线性学习者（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='linear-learner',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/linear-learner：*<tag>*	1	训练、推理		

MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='mxnet',region='us-gov-
west-1',version='1.4.1',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/mxnet-training：*<tag>*	1.8.0	培训	CPU，GPU	py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.8.0	推断	CPU，GPU	py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/mxnet-training：*<tag>*	1.7.0	培训	CPU，GPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.7.0	推断	CPU，GPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.7.0	eia	CPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/mxnet-training：*<tag>*	1.6.0	培训	CPU，GPU	py2、py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.6.0	推断	CPU，GPU	py2、py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/	1.5.1	eia	CPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
mxnet-inference-eia：*<tag>*				
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.1	培训	CPU，GPU	py2
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/mxnet-training：*<tag>*	1.4.1	培训	CPU，GPU	py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.1	推断	CPU，GPU	py2
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/mxnet-inference：*<tag>*	1.4.1	推断	CPU，GPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/mxnet-inference-eia：*<tag>*	1.4.1	eia	CPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet-serving：*<tag>*	1.4.0	推断	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet-serving-eia：*<tag>*	1.4.0	eia	CPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.3.0	推断	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet-eia：*<tag>*	1.3.0	eia	CPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2.1	推断	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2-1	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-mxnet：*<tag>*	1.2-1	推断	CPU，GPU	py2、py3

MxNet 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-mxnet',region='us-gov-
west-1',version='0.11',py_version='py3',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11.0-*<tag>*	0.11-0	培训	CPU，GPU	py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-mxnet: 教练 0.11-*<tag>*	0.11-	培训	CPU，GPU	py3

NTM（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ntm',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/ntm：*<tag>*	1	训练、推理		

Neo 图像分类（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='image-classification-neo',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/image-classification-neo：*<tag>*	latest	推断		

Neo MxNet (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-mxnet',region='us-gov-
west-1',version='1.8',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-inference-mxnet：*<tag>*	1.8	推断	CPU，GPU	py3

新 PyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-pytorch',region='us-gov-
west-1',version='1.6',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.7	推断	CPU，GPU	py3
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.6	推断	CPU，GPU	py3
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.5	推断	CPU，GPU	py3
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-inference-pytorch：*<tag>*	1.4	推断	CPU，GPU	py3

Neo TensorFlow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='neo-tensorflow',region='us-gov-
west-1',version='1.15.3',py_version='py3',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	2.4.2	推断	CPU，GPU	py3
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-inference-tensorflow：*<tag>*	1.15.3	推断	CPU，GPU	py3

Neo xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost-neo',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/xgboost-neo：*<tag>*	latest	推断		

对象检测（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object-detection',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/object-detection：*<tag>*	1	训练、推理		

Object2VEC（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='object2vec',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/object2vec：*<tag>*	1	训练、推理		

PCA（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pca',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/pca：*<tag>*	1	训练、推理		

pyTorch (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='pytorch',region='us-gov-
west-1',version='1.8.0',py_version='py3',image_scope='inference',
 instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.11.0	推断	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.11.0	培训	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.2	推断	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.10.2	培训	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.10.0	推断	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.10.0	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.1	推断	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.9.1	培训	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.9.0	推断	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.9.0	培训	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.1	推断	CPU，GPU	py3、py36
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.8.1	培训	CPU，GPU	py3、py36
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.8.0	推断	CPU，GPU	py3、py36
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.8.0	培训	CPU，GPU	py3、py36
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.7.1	推断	CPU，GPU	py3、py36

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.7.1	培训	CPU，GPU	py3、py36
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.6.0	推断	CPU，GPU	py3、py36
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.6.0	培训	CPU，GPU	py3、py36
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.5.0	推断	CPU，GPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.5.0	培训	CPU，GPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.4.0	推断	CPU，GPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.4.0	培训	CPU，GPU	py2、py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.3.1	推断	CPU，GPU	py2、py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.3.1	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-inference：*<tag>*	1.2.0	推断	CPU，GPU	py2、py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/pytorch-training：*<tag>*	1.2.0	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	推断	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.1.0	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	推断	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	1.0.0	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	推断	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-pytorch：*<tag>*	0.4.0	培训	CPU，GPU	py2、py3

随机森林砍伐 (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='randomcutforest',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/randomcutforest：*<tag>*	1	训练、推理		

Scikit-learn (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sklearn',region='us-gov-
west-1',version='0.23-1',image_scope='inference')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
414596584902.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.23-1	0.23.2	训练、推理	
414596584902.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-scikit-learn：*<tag>*	0.20.0	0.20.0	训练、推理	

语义分割（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='semantic-segmentation',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/semantic-segmentation：*<tag>*	1	训练、推理		

Sq2Seq (算法)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='seq2seq',region='us-gov-west-1')
```

注册表路径	版本	Job 类型（图像范围）		
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/seq2seq：*<tag>*	1	训练、推理		

Spark（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='spark',region='us-gov-
west-1',version='3.0',image_scope='processing')
```

注册表路径	版本	Job 类型（图像范围）		
271483468897.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.1	处理		
271483468897.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	3.0	处理		
271483468897.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-spark-processing：*<tag>*	2.4	处理		

SparkML 服务（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='sparkml-serving',region='us-gov-west-1',version='2.4')
```

注册表路径	版本	Job 类型（图像范围）		
414596584902.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-sparkml-serving：*<tag>*	2.4	推断		
414596584902.dkr.ecr.us-gov-west-1.amazonaws.com/	2.2	推断		

注册表路径	版本	Job 类型（图像范围）		
sagemaker-sparkml-serving：<tag>				

Tensorflow (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='tensorflow',region='us-gov-
west-1',version='1.12.0',image_scope='inference',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.8.0	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.8.0	培训	CPU，GPU	py39
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.7.1	培训	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.7.0	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.6.3	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.6.3	培训	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.6.2	培训	CPU，GPU	py38

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.6.0	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.6.0	培训	CPU，GPU	py38
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.5.1	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.5.1	培训	CPU，GPU	py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.5.0	培训	CPU，GPU	py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.4.3	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.4.3	培训	CPU，GPU	py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.4.1	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.4.1	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.3.2	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.3.2	培训	CPU，GPU	py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.3.1	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.3.1	培训	CPU，GPU	py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.3.0	eia	CPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.3.0	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.3.0	培训	CPU，GPU	py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.2.2	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.2.2	培训	CPU，GPU	py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.1.3	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.1.3	培训	CPU，GPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.1.2	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.1.2	培训	CPU，GPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.1.1	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.1	培训	CPU，GPU	py2、py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.1.0	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.1.0	培训	CPU，GPU	py2、py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.4	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.4	培训	CPU，GPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.3	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.3	培训	CPU，GPU	py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：*<tag>*	2.0.2	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：*<tag>*	2.0.2	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.0.1	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.0.1	培训	CPU，GPU	py2、py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference-eia：<tag>	2.0.0	eia	CPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	2.0.0	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	2.0.0	培训	CPU，GPU	py2、py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	1.15.5	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	1.15.5	培训	CPU，GPU	py3、py36、py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	1.15.4	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	1.15.4	培训	CPU，GPU	py3、py36、py37

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	1.15.3	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	1.15.3	培训	CPU，GPU	py2、py3、py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	1.15.2	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	1.15.2	培训	CPU，GPU	py2、py3、py37
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.15.0	eia	CPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	1.15.0	推断	CPU，GPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：<tag>	1.15.0	培训	CPU，GPU	py2、py3
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference-eia：<tag>	1.14.0	eia	CPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：<tag>	1.14.0	推断	CPU，GPU	-

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.14.0	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.13.1	培训	CPU，GPU	py2
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-training：*<tag>*	1.13.1	培训	CPU，GPU	py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.13.0	eia	CPU	-
442386744353.dkr.ecr.us-gov-west-1.amazonaws.com/tensorflow-inference：*<tag>*	1.13.0	推断	CPU，GPU	-
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.12.0	eia	CPU	-
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.12.0	推断	CPU，GPU	-
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.12.0	培训	CPU，GPU	py2、py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow-serving-eia：*<tag>*	1.11.0	eia	CPU	-
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow-serving：*<tag>*	1.11.0	推断	CPU，GPU	-
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow-scriptmode：*<tag>*	1.11.0	培训	CPU，GPU	py2、py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow-eia：*<tag>*	1.10.0	eia	CPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	推断	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.10.0	培训	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	推断	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：*<tag>*	1.9.0	培训	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	推断	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.8.0	培训	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	推断	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.7.0	培训	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	推断	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.6.0	培训	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	推断	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.5.0	培训	CPU，GPU	py2
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4.1	推断	CPU，GPU	py2

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-tensorflow：<tag>	1.4-1	培训	CPU，GPU	py2

Tensorflow 教练（DLC）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='coach-tensorflow',region='us-gov-
west-1',version='1.0.0',image_scope='training',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.1-<tag>	0.11-1	培训	CPU，GPU	py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11.0-<tag>	0.11-0	培训	CPU，GPU	py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.11-<tag>	0.11-	培训	CPU，GPU	py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10.1-<tag>	0.10-1	培训	CPU，GPU	py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-tensorflow: 教练 0.10-<tag>	0.10-	培训	CPU，GPU	py3

张量流推断 (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='inferentia-tensorflow',region='us-gov-
west-1',version='1.15.0',instance_type='ml.inf1.6xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	2.5.2	推断	inf	py3
263933020539.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-neo-tensorflow：*<tag>*	1.15.0	推断	inf	py3

Tensorflow Ray (DLC)

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='ray-tensorflow',region='us-gov-
west-1',version='0.8.5',instance_type='ml.c5.4xlarge')
```

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6.5-*<tag>*	0.6.5	培训	CPU，GPU	py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.6-*<tag>*	0.6	培训	CPU，GPU	py3
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5.3-*<tag>*	0.5.3	培训	CPU，GPU	py3

注册表路径	版本	Job 类型（图像范围）	处理器类型	Python 版本
246785580436.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-rl-tensorflow:ray0.5-<tag>	0.5	培训	CPU，GPU	py3

xgBoost（算法）

用于检索注册表路径的 SageMaker Python SDK 示例。

```
from sagemaker import image_uris
image_uris.retrieve(framework='xgboost',region='us-gov-west-1',version='1.2-1')
```

注册表路径	版本	程序包版本	Job 类型（图像范围）	
414596584902.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-xgboost：<tag>	us-5-1	1.5.2	训练、推理	
414596584902.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-xgboost：<tag>	us-3-1	1.3.3	训练、推理	
414596584902.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-xgboost：<tag>	us-2-2	1.2.0	训练、推理	
414596584902.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-xgboost：<tag>	us-2-1	1.2.0	训练、推理	
414596584902.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-xgboost：<tag>	us-0-1	1.0	训练、推理	
226302683700.dkr.ecr.us-gov-west-1.amazonaws.com/xgboost：<tag>	us-	1.5.2	训练、推理	
414596584902.dkr.ecr.us-gov-	us-90-2	0.90	训练、推理	

注册表路径	版本	程序包版本	Job 类型（图像范围）	
west-1.amazonaws.com/sagemaker-xgboost：<tag>				
414596584902.dkr.ecr.us-gov-west-1.amazonaws.com/sagemaker-xgboost：<tag>	0.90-1	0.90	训练、推理	

内置算法的常见数据格式

以下主题说明了 Amazon SageMaker 提供的算法的数据格式。

主题

- 用于训练的常见数据格式 (p. 1579)
- 用于推理的常见数据格式 (p. 1583)

用于训练的常见数据格式

要准备训练，您可以使用各种Amazon服务，包括Amazon Glue、Amazon EMR、Amazon Redshift、Amazon Relational Database Service 和 Amazon Athena。在进行预处理后，将数据发布到 Amazon S3 存储桶。对于训练，数据需要经过一系列的转换和变换，包括：

- 训练数据序列化 (由您处理)
- 训练数据反序列化 (由算法处理)
- 训练模型序列化 (由算法处理)
- 训练后的模型反序列化 (可选，由您处理)

使用亚马逊时 SageMaker 在算法的训练部分，请确保一次性上传所有数据。如果向该位置添加更多数据，则需发起新的训练调用以构建全新模型。

主题

- 内置算法支持的内容类型 (p. 1579)
- 使用管道模式 (p. 1580)
- 使用 CSV 格式 (p. 1580)
- 使用 RERecordIO 格式 (p. 1580)
- 训练后的模型反序列化 (p. 1582)

内置算法支持的内容类型

下表列出了一些常见支持的`ContentType`值和使用它们的算法：

内置算法的内容类型

ContentType	算法
应用程序/x-映像	对象检测算法、语义分段
applics/x-RECORDIO	对象检测算法

ContentType	算法
application/x-recordio-protobuf	分解机、K-均值、K-NN、潜在 Dirichlet 分配、线性学习者、NTM、PCA、RCF、序列对序
application /jsonlines	DeepingText
映像 /jpeg	对象检测算法、语义分段
图片/PNG	对象检测算法、语义分段
text/csv	知识产权见解、K-均值、K-NN、潜在 Dirichlet 分配、线性学习者、NTM、PCA、RCF、xgBoost
text/libsvm	XGBoost

有关每种算法使用的参数的摘要，请参阅各个算法的文档或此。桌子.

使用管道模式

In管道模式，您的训练作业直接从 Amazon Simple Storage Service (Amazon S3) 流式传输数据。流式传输可以为训练作业提供更快的启动时间和更好的吞吐量。这与文件模式，其中来自 Amazon S3 的数据存储在培训实例卷上。文件模式使用磁盘空间来存储最终的模型构件和完整的训练数据集。通过在管道模式下直接从 Amazon S3 传输数据，您可以减少训练实例的 Amazon Elastic Block Store 卷的大小。管道模式只需要足够的磁盘空间来存储最终模型构件。有关训练输入模式的更多详细信息，请参阅`AlgorithmSpecification`。

使用 CSV 格式

多个亚马逊 SageMaker 算法支持使用 CSV 格式的数据进行训练。要使用 CSV 格式的数据进行训练，请在输入数据通道规范中指定**`text/csv`**作为`ContentType`. 亚马逊 SageMaker 要求 CSV 文件没有标头记录，并且目标变量在第一列中。要运行没有目标的自主学习算法，请指定内容类型中的标签列数。例如，在此示例中，**`'content_type=text/csv;label_size=0'`**。有关使用 CSV 格式的笔记本示例，请参阅乳腺癌预测. 有关更多信息，请参阅 。现在，将 Pipe 模式与 CSV 数据集结合使用，以便在 Amazon SageMaker 内置算法上.

使用 RERecordIO 格式

在 protobuf recordIO 格式中， SageMaker 将数据集中的每个观察转换为二进制表示形式（一组 4 字节浮点数），然后将其加载到 protobuf 值字段。如果您使用 Python 进行数据准备，我们强烈建议您使用这些现有的转换。但是，如果您使用的是其他语言，则下面的 protobuf 定义文件提供了用于将数据转换为的架构。SageMaker protobuf 格式。

> **Note**
>
> 有关说明如何将常用 numPy 数组转换为 protobuf recordIO 格式的示例，请参阅使用 MNIST 的分解机器简介.

```
syntax = "proto2";

 package aialgs.data;

 option java_package = "com.amazonaws.aialgorithms.proto";
 option java_outer_classname = "RecordProtos";

 // A sparse or dense rank-R tensor that stores data as doubles (float64).
 message Float32Tensor   {
     // Each value in the vector. If keys is empty, this is treated as a
     // dense vector.
     repeated float values = 1 [packed = true];
```

```
    // If key is not empty, the vector is treated as sparse, with
    // each key specifying the location of the value in the sparse vector.
    repeated uint64 keys = 2 [packed = true];

    // An optional shape that allows the vector to represent a matrix.
    // For example, if shape = [ 10, 20 ], floor(keys[i] / 20) gives the row,
    // and keys[i] % 20 gives the column.
    // This also supports n-dimensonal tensors.
    // Note: If the tensor is sparse, you must specify this value.
    repeated uint64 shape = 3 [packed = true];
}

// A sparse or dense rank-R tensor that stores data as doubles (float64).
message Float64Tensor {
    // Each value in the vector. If keys is empty, this is treated as a
    // dense vector.
    repeated double values = 1 [packed = true];

    // If this is not empty, the vector is treated as sparse, with
    // each key specifying the location of the value in the sparse vector.
    repeated uint64 keys = 2 [packed = true];

    // An optional shape that allows the vector to represent a matrix.
    // For example, if shape = [ 10, 20 ], floor(keys[i] / 10) gives the row,
    // and keys[i] % 20 gives the column.
    // This also supports n-dimensonal tensors.
    // Note: If the tensor is sparse, you must specify this value.
    repeated uint64 shape = 3 [packed = true];
}

// A sparse or dense rank-R tensor that stores data as 32-bit ints (int32).
message Int32Tensor {
    // Each value in the vector. If keys is empty, this is treated as a
    // dense vector.
    repeated int32 values = 1 [packed = true];

    // If this is not empty, the vector is treated as sparse with
    // each key specifying the location of the value in the sparse vector.
    repeated uint64 keys = 2 [packed = true];

    // An optional shape that allows the vector to represent a matrix.
    // For Exmple, if shape = [ 10, 20 ], floor(keys[i] / 10) gives the row,
    // and keys[i] % 20 gives the column.
    // This also supports n-dimensonal tensors.
    // Note: If the tensor is sparse, you must specify this value.
    repeated uint64 shape = 3 [packed = true];
}

// Support for storing binary data for parsing in other ways (such as JPEG/etc).
// This is an example of another type of value and may not immediately be supported.
message Bytes {
    repeated bytes value = 1;

    // If the content type of the data is known, stores it.
    // This allows for the possibility of using decoders for common formats
    // in the future.
    optional string content_type = 2;
}

message Value {
    oneof value {
        // The numbering assumes the possible use of:
        // - float16, float128
        // - int8, int16, int32
        Float32Tensor float32_tensor = 2;
```

```
        Float64Tensor float64_tensor = 3;
        Int32Tensor int32_tensor = 7;
        Bytes bytes = 9;
    }
}

message Record {
    // Map from the name of the feature to the value.
    //
    // For vectors and libsvm-like datasets,
    // a single feature with the name `values`
    // should be specified.
    map<string, Value> features = 1;

    // An optional set of labels for this record.
    // Similar to the features field above, the key used for
    // generic scalar / vector labels should be 'values'.
    map<string, Value> label = 2;

    // A unique identifier for this record in the dataset.
    //
    // Whilst not necessary, this allows better
    // debugging where there are data issues.
    //
    // This is not used by the algorithm directly.
    optional string uid = 3;

    // Textual metadata describing the record.
    //
    // This may include JSON-serialized information
    // about the source of the record.
    //
    // This is not used by the algorithm directly.
    optional string metadata = 4;

    // An optional serialized JSON object that allows per-record
    // hyper-parameters/configuration/other information to be set.
    //
    // The meaning/interpretation of this field is defined by
    // the algorithm author and may not be supported.
    //
    // This is used to pass additional inference configuration
    // when batch inference is used (e.g. types of scores to return).
    optional string configuration = 5;
}
```

在创建协议缓冲区后，将它存储在 Amazon S3 位置。 SageMaker 可以访问并且可以作为`InputDataConfig`在`create_training_job`.

Note

对于所有亚马逊 SageMaker 算法，`ChannelName`在`InputDataConfig`必须将其设为`train`. 一些算法还支持验证或测试 `input channels`。它们通常用于通过使用保留数据集来评估模型的性能。保留数据集不用于初始训练，但可用于进一步调整模型。

训练后的模型反序列化

亚马逊 SageMaker 模型在中指定的 S3 存储桶中存储为 model.tar.gz。`OutputDataConfig` `S3OutputPath`的参数`create_training_job`调用。您可以在创建托管模型时指定这些模型构件中的大多数模型构件。您也可以在笔记本实例中打开并查看它们。在展开 `model.tar.gz` 时，它将包含 `model_algo-1`，后者是一个序列化的 Apache MXNet 对象。例如，您可以使用以下命令将 k-means 模型加载到内存中并进行查看：

```
import mxnet as mx
```

```
print(mx.ndarray.load('model_algo-1'))
```

用于推理的常见数据格式

亚马逊 SageMaker 对于用于检索在线和小批量预测的 HTTP 负载，算法可接受和生成多种不同的 MIME 类型。你可以使用各种Amazon在运行推理之前转换或预处理记录的服务。您至少需要将数据转换为以下内容：

- 推理请求序列化 (由您处理)
- 推理请求反序列化 (由算法处理)
- 推理响应序列化 (由算法处理)
- 推理响应反序列化 (由您处理)

主题

- 转换用于推理请求序列化的数据 (p. 1583)
- 转换用于推理响应反序列化的数据 (p. 1584)
- 所有算法的常见请求格式 (p. 1585)
- 将 Batch 转换结合使用内置算法 (p. 1587)

转换用于推理请求序列化的数据

亚马逊的内容类型选项 SageMaker 算法推理请求包括：`text/csv`、`application/json`, 和`application/x-recordio-protobuf`. 并不支持所有这些类型的算法可以支持其他类型。例如，XGBoost 仅支持此列表中的 `text/csv`，但也支持 `text/libsvm`。

适用于`text/csv`，Body 参数的值`invoke_endpoint`应为一个字符串，其中用逗号分隔每个特性的值。例如，具有四个特性的模型的记录可能类似于`1.5,16.0,14,23.0`. 在获得推理结果之前，还应对数据执行已对训练数据执行的任何转换。特性的顺序很重要，并且必须保持不变。

`application/json` 更为灵活，并为开发人员提供多种可能的格式以便在应用程序中使用。从较高的层面上说，在 JavaScript 中，负载可能类似于以下内容：

```
let request = {
  // Instances might contain multiple rows that predictions are sought for.
  "instances": [
    {
      // Request and algorithm specific inference parameters.
      "configuration": {},
      // Data in the specific format required by the algorithm.
      "data": {
         "<field name>": dataElement
       }
    }
  ]
}
```

对于指定 `dataElement`，您可使用以下选项：

协议缓冲区等效函数

```
// Has the same format as the protocol buffers implementation described for training.
let dataElement = {
  "keys": [],
  "values": [],
  "shape": []
}
```

简单数字向量

```
// An array containing numeric values is treated as an instance containing a
// single dense vector.
let dataElement = [1.5, 16.0, 14.0, 23.0]

// It will be converted to the following representation by the SDK.
let converted = {
  "features": {
    "values": dataElement
  }
}
```

对于多条记录

```
let request = {
  "instances": [
    // First instance.
    {
      "features": [ 1.5, 16.0, 14.0, 23.0 ]
    },
    // Second instance.
    {
      "features": [ -2.0, 100.2, 15.2, 9.2 ]
    }
  ]
}
```

转换用于推理响应反序列化的数据

亚马逊 SageMaker 算法以多种布局返回 JSON。从较高的层面上说，结构为：

```
let response = {
  "predictions": [{
    // Fields in the response object are defined on a per algorithm-basis.
  }]
}
```

对于各种算法而言，预测中包含的字段是不同的。以下是 k-means 算法的输出示例。

单记录推理

```
let response = {
  "predictions": [{
    "closest_cluster": 5,
    "distance_to_cluster": 36.5
  }]
}
```

多记录推理

```
let response = {
  "predictions": [
    // First instance prediction.
    {
      "closest_cluster": 5,
      "distance_to_cluster": 36.5
    },
    // Second instance prediction.
    {
```

```
      "closest_cluster": 2,
      "distance_to_cluster": 90.3
    }
  ]
}
```

具有 protobuf 输入的 protobuf 输入多记录推理

```
{
  "features": [],
  "label": {
    "closest_cluster": {
      "values": [ 5.0 ] // e.g. the closest centroid/cluster was 1.0
    },
    "distance_to_cluster": {
      "values": [ 36.5 ]
    }
  },
  "uid": "abc123",
  "metadata": "{ "created_at": '2017-06-03' }"
}
```

SageMaker 算法还支持 JSONLINES 格式，其中每条记录的响应内容与 JSON 格式的内容相同。多记录结构是用换行符分隔的每个记录响应对象的串联。内置 KMeans 算法对 2 个输入数据点的响应内容为：

```
{"distance_to_cluster": 23.40593910217285, "closest_cluster": 0.0}
{"distance_to_cluster": 27.250282287597656, "closest_cluster": 0.0}
```

在运行批量转换时，我们建议使用`jsonlines`通过设置响应类型`Accept`字段中的`CreateTransformJobRequest`到`application/jsonlines`.

所有算法的常见请求格式

大多数算法使用以下推理请求格式中的几种格式。

JSON 请求格式

内容类型：应用程序/JSON

密集格式

```
let request =   {
    "instances":    [
        {
            "features": [1.5, 16.0, 14.0, 23.0]
        }
    ]
}


let request =   {
    "instances":    [
        {
            "data": {
                "features": {
                    "values": [ 1.5, 16.0, 14.0, 23.0]
                }
            }
        }
    ]
}
```

稀疏格式

```
{
 "instances": [
  {"data": {"features": {
     "keys": [26, 182, 232, 243, 431],
     "shape": [2000],
     "values": [1, 1, 1, 4, 1]
    }
   }
  },
  {"data": {"features": {
     "keys": [0, 182, 232, 243, 431],
     "shape": [2000],
     "values": [13, 1, 1, 4, 1]
    }
   }
  },
 ]
}
```

JSONLINES 请求格式

内容类型：application /jsonlines

密集格式

密集格式的单个记录可以表示为：

```
{ "features": [1.5, 16.0, 14.0, 23.0] }
```

或：

```
{ "data": { "features": { "values": [ 1.5, 16.0, 14.0, 23.0] } }
```

稀疏格式

稀疏格式的单个记录表示为：

```
{"data": {"features": { "keys": [26, 182, 232, 243, 431], "shape": [2000], "values": [1, 1,
 1, 4, 1] } } }
```

多个记录表示为上述单记录表示的串联（用换行符分隔）：

```
{"data": {"features": { "keys": [0, 1, 3], "shape": [4], "values": [1, 4, 1] } } }
{ "data": { "features": { "values": [ 1.5, 16.0, 14.0, 23.0] } }
{ "features": [1.5, 16.0, 14.0, 23.0] }
```

CSV 请求格式

内容类型：text/CSV; label_size=0

> **Note**
>
> 因子分解机不提供 CSV 支持。

RECORDIO 请求格式

内容类型：applice/x-recordio-protobuf

将 Batch 转换结合使用内置算法

在运行批量转换时，如果算法支持，我们建议使用 JSONLINES 响应类型而不是 JSON。这是通过将 `CreateTransformJobRequest` 中的 `Accept` 字段设置为 `application/jsonlines` 来实现的。

在创建转换作业时，必须根据输入数据的 `ContentType` 来设置 `SplitType`。同样，必须根据 `CreateTransformJobRequest` 中的 `Accept` 字段来相应地设置 `AssembleWith`。请使用下表来帮助您正确设置这些字段：

ContentType	建议的 SplitType
`application/x-recordio-protobuf`	`RecordIO`
`text/csv`	`Line`
`application/jsonlines`	`Line`
`application/json`	`None`
`application/x-image`	`None`
`image/*`	`None`
Accept	建议的 AssembleWith
`application/x-recordio-protobuf`	`None`
`application/json`	`None`
`application/jsonlines`	`Line`

有关特定响应格式的更多信息算法，请参阅：

- DeepAR 推理格式 (p. 1608)
- 因子分解机响应格式 (p. 1619)
- IP 见解推理数据格式 (p. 1636)
- K-Means 响应格式 (p. 1643)
- k-NN 请求和响应格式 (p. 1649)
- 线性学员响应格式 (p. 1670)
- NTM 响应格式 (p. 1676)
- 用于 Object2Vec 推理的数据格式 (p. 1689)
- Object2Vec 的编码器嵌入 (p. 1690)
- PCA 响应格式 (p. 1703)
- RCF 响应格式 (p. 1709)

内置算法的实例类型

用于培训和托管亚马逊 SageMaker 算法，建议您使用以下 Amazon EC2 实例类型：

- ml.m5.xlarge、ml.m5.4xlarge 和 ml.m5.10xlarge
- ml.c5.xlarge、ml.c5.2xlarge 和 ml.c5.8xlarge
- ml.p3.xlarge、ml.p3.8xlarge 和 ml.p3.16xlarge

多数亚马逊 SageMaker 算法已设计为将 GPU 计算用于训练。尽管每实例成本较高，但 GPU 训练的速度更快，因此更经济高效。本指南中注明了例外情况。

数据的大小和类型会对最高效硬件配置产生很大的影响。在定期训练相同的模型时，在一系列实例类型中进行初始测试可找到在长时间运行中最经济高效的配置。此外，在 GPU 上进行最有效训练的算法可能不需要 GPU 来进行有效推理。进行试验，以确定最具成本效益的解决方案。

有关 SageMaker 请参阅硬件规格亚马逊 SageMaker ML 实例类型.

内置算法的日志

亚马逊 SageMaker 算法生成亚马逊 CloudWatch 日志，这些日志提供了有关训练过程的详细信息。要查看日志，请在Amazon管理控制台，选择CloudWatch，选择日志然后选择 /AWS/sagemaker/训练职位/AWS/sagemaker/训练作业日志组. 每个训练作业都有一个日志流，这些日志流在其上训练它的节点。日志流的名称以创建作业时在 `TrainingJobName` 参数中指定的值开头。

> **Note**
>
> 如果作业失败，并且 CloudWatch 中未显示日志，则在开始训练前可能已出错。原因包括指定了错误的训练镜像或 S3 位置。

日志的内容因算法而异。不过，您通常可以找到以下信息：

- 日志开头部分提供的参数的确认
- 训练期间出现的错误
- 测量算法的准确性或数字性能
- 算法的计时和算法中的任何主要阶段

常见错误

如果训练作业失败，则训练作业描述中的 `FailureReason` 返回值会提供有关失败情况的一些详细信息，如下所示：

```
sage = boto3.client('sagemaker')
sage.describe_training_job(TrainingJobName=job_name)['FailureReason']
```

其他信息仅在 CloudWatch 日志。常见错误包括：

1. 错误指定了超参数，或指定的超参数对于算法无效。

   从 CloudWatch 日志

   ```
   [10/16/2017 23:45:17 ERROR 139623806805824 train.py:48]
   Additional properties are not allowed (u'mini_batch_siz' was
   unexpected)
   ```

2. 为超参数指定的值无效。

   FailureReason

   ```
   AlgorithmError: u'abc' is not valid under any of the given
   schemas\n\nFailed validating u'oneOf' in
   schema[u'properties'][u'feature_dim']:\n    {u'oneOf':
   [{u'pattern': u'^([1-9][0-9]*)$', u'type': u'string'},\n
   {u'minimum': 1, u'type': u'integer'}]}\
   ```

   FailureReason

   ```
   [10/16/2017 23:57:17 ERROR 140373086025536 train.py:48] u'abc'
   ```

```
is not valid under any of the given schemas
```

3. protobuf 文件格式不正确。

   从 CloudWatch 日志

```
[10/17/2017 18:01:04 ERROR 140234860816192 train.py:48] cannot
                    copy sequence with size 785 to array axis with dimension 784
```

BlazingText 算法

Amazon SageMaker BlazingText 算法提供了 Word2vec 和文本分类算法的高度优化的实现。Word2vec 算法对很多下游自然语言处理 (NLP) 任务很有用，例如情感分析、命名实体识别、机器翻译等。对于执行 Web 搜索、信息检索、排名和文档分类的应用程序来说，文本分类是一项重要任务。

Word2vec 算法将单词映射到高质量的分布式向量。单词的结果向量表示被称为词嵌入。语义相似的单词对应于相近的向量。这样一来，词嵌入便能捕获单词之间的语义关系。

很多自然语言处理 (NLP) 应用程序通过对大量文档集进行训练来学习词嵌入。这些预训练的向量表示提供了有关语义和单词分布的信息，这些信息通常会改善稍后在更有限数量的数据上训练的其他模型的普遍性。Word2vec 算法的大多数实施未针对多核 CPU 架构进行优化。这使得难以扩展到大型数据集。

利用 BlazingText 算法，您可以轻松扩展到大型数据集。与 Word2vec 类似，它提供了 Skip-gram 和持续单词袋 (CBOW) 训练架构。BlazingText 实现的指导式多类、多标签文本分类算法扩展了 fastText 文本分类器，以将 GPU 加速与自定义 CUDA 内核配合使用。您可以使用多核 CPU 或 GPU 在几分钟内对超过 10 亿个单词的模型进行训练。而且，您获得的性能与最先进的深度学习文本分类算法相当。

BlazingText 算法不可并行化。有关训练相关参数的更多信息，请参阅。SageMaker 内置算法的 Docker 注册表路径.

SageMaker BlazingText 算法提供以下功能：

- 使用高度优化的 CUDA 内核在多核 CPU 或 GPU 以及 GPU 上的 Word2Vec 上加速训练 fastText 文本分类器。有关更多信息，请参阅 。BlazingText：使用多个 GPU 扩展和加快 Word2VEC.
- 通过学习字符 n-grams 的向量表示，丰富了包含子词信息的词向量。这种方法使 BlazingText 能够通过将其向量表示为字符 n-gram（子词）向量的总和来为词汇表以外 (OOV) 的单词生成有意义的向量。
- Word2Vec 算法的 `batch_skipgram mode`，它允许跨多个 CPU 节点进行更快的训练和分布式计算。`batch_skipgram mode` 使用负采样共享策略进行小型批处理，将级别 1 BLAS 运算转换为级别 3 BLAS 运算。这有效地利用了现代架构的乘-加指令。有关更多信息，请参阅在共享和分布式内存中并行处理 Word2Vec。

总而言之，BlazingText 在不同类型的实例上支持以下模式：

Modes	Word2Vec （自主学习）	文本分类 （指导式学习）

单 CPU 实例	`cbow`
	`Skip-gram`
	`Batch Skip-gram`
		`supervised`

单 GPU 实例（具有一个或多个 GPU）	`cbow` `Skip-gram`	`supervised`（具有一个 GPU）

Modes	Word2Vec （自主学习）	文本分类 （指导式学习）
多个 CPU 实例	`Batch Skip-gram`	无

有关 BlazingText 背后数学的更多信息，请参阅。BlazingText：使用多个 GPU 扩展和加快 Word2VEC.

主题

- BlazingText 算法的输入/输出接口 (p. 1590)
- BlazingText 算法的 EC2 实例建议 (p. 1592)
- BlazingText 示例笔记本 (p. 1593)
- BlazingText 超参数 (p. 1593)
- 调整 BlazingText 模型 (p. 1596)

BlazingText 算法的输入/输出接口

BlazingText 算法需要一个带有空格分隔的标记的预处理文本文件。文件中的每一行都应包含一个句子。如果您需要训练多个文本文件，请将它们连接成一个文件并在相应的通道中上传该文件。

训练和验证数据格式

Word2Vec 算法的训练和验证数据格式

对于 Word2Vec 训练，在训练 通道下上传文件。不支持任何其他通道。文件中的每一行都应包含一个训练句子。

文本分类算法的训练和验证数据格式

对于指导式模式，您可以使用文件模式或增强清单文本格式进行训练。

使用文件模式进行训练

对于 `supervised` 模式，训练/验证文件应包含训练句子（一行一个）以及标签。标签是以字符串 __label__ 作为前缀的单词。以下是训练/验证文件的示例：

```
__label__4  linux ready for prime time , intel says , despite all the linux hype , the
 open-source movement has yet to make a huge splash in the desktop market . that may be
 about to change , thanks to chipmaking giant intel corp .

__label__2  bowled by the slower one again , kolkata , november 14 the past caught up with
 sourav ganguly as the indian skippers return to international cricket was short lived .
```

> **Note**
>
> 句子中标签的顺序无关紧要。

在训练通道下上传训练文件，并可以选择在验证通道下上传验证文件。

使用增强清单文本格式进行训练

指导式模式还支持增强清单格式，这使您能够在管道模式下进行训练，而无需创建 RecordIO 文件。在使用此格式时，需要生成一个包含句子及其相应标签的列表的 S3 清单文件。清单文件格式应为 JSON 行格式，其中每行代表一个样本。使用 `source` 标签指定句子，并且可使用 `label` 标签指定标签。应在 `AttributeNames` 参数值下面提供 `source` 和 `label` 标签，如请求中指定。

```
{"source":"linux ready for prime time , intel says , despite all the linux hype",
 "label":1}
{"source":"bowled by the slower one again , kolkata , november 14 the past caught up with
 sourav ganguly", "label":2}
```

也可以通过指定标签的 JSON 数组来支持多标签训练。

```
{"source":"linux ready for prime time , intel says , despite all the linux hype", "label":
 [1, 3]}
{"source":"bowled by the slower one again , kolkata , november 14 the past caught up with
 sourav ganguly", "label": [2, 4, 5]}
```

有关增强清单文件的更多信息，请参阅通过增强清单文件，将数据集元数据提供给训练作业 (p. 2090)。

模型构件和推理

Word2Vec 算法的模型构件

对于 Word2Vec 训练，模型构件包含 vectors.txt（它包含单词到向量的映射）和 vectors.bin（BlazingText 用来托管和/或推理的二进制文件）。vectors.txt 以与 Gensim 和 Spacy 等其他工具兼容的格式存储向量。例如，Gensim 用户可以运行以下命令来加载 vectors.txt 文件：

```
from gensim.models import KeyedVectors
word_vectors = KeyedVectors.load_word2vec_format('vectors.txt', binary=False)
word_vectors.most_similar(positive=['woman', 'king'], negative=['man'])
word_vectors.doesnt_match("breakfast cereal dinner lunch".split())
```

如果将评估参数设置为 `True`，则会创建另一个文件 eval.json。此文件包含 WS-353 数据集的相似性评估结果（使用 Spearman 的秩相关系数）。报告训练语料库中不存在的 WS-353 数据集中的单词的数量。

对于推理请求，模型接受一个包含字符串列表的 JSON 文件，并返回一个向量列表。如果在词汇表中找不到该单词，则推理返回零向量。如果在训练期间将子词设置为 `True`，则该模型能够为词汇表以外 (OOV) 的单词生成向量。

示例 JSON 请求

Mime 类型： `application/json`

```
{
"instances": ["word1", "word2", "word3"]
}
```

文本分类算法的模型构件

使用指导式输出进行训练会创建一个 model.bin 文件，该文件可由 BlazingText 托管使用。对于推理，BlazingText 模型接受包含句子列表的 JSON 文件，并返回相应的预测标签和概率分数的列表。每个句子都应是一个字符串，其中包含空格分隔的标记和/或单词。

示例 JSON 请求

Mime 类型： `application/json`

```
{
 "instances": ["the movie was excellent", "i did not like the plot ."]
}
```

默认情况下，服务器仅返回一个预测，即概率最高的预测。要检索前 k 个预测，您可以在配置中设置 k，如下所示：

```
{
 "instances": ["the movie was excellent", "i did not like the plot ."],
 "configuration": {"k": 2}
}
```

对于 BlazingText， `content-type` 和 `accept` 参数必须相等。对于批量转换，它们都需要是 `application/jsonlines`。如果它们不同，则将忽略 `Accept` 字段。输入的格式如下：

```
content-type: application/jsonlines

{"source": "source_0"}
{"source": "source_1"}

if you need to pass the value of k for top-k, then you can do it in the following way:

{"source": "source_0", "k": 2}
{"source": "source_1", "k": 3}
```

输出的格式如下：

```
accept: application/jsonlines


{"prob": [prob_1], "label": ["__label__1"]}
{"prob": [prob_1], "label": ["__label__1"]}

If you have passed the value of k to be more than 1, then response will be in this format:

{"prob": [prob_1, prob_2], "label": ["__label__1", "__label__2"]}
{"prob": [prob_1, prob_2], "label": ["__label__1", "__label__2"]}
```

对于指导式（文本分类）模式和自主 (Word2Vec) 模式，BlazingText 生成的二进制文件 (*.bin) 可以由 fastText 交叉使用，反之亦然。您可以通过 fastText 使用由 BlazingText 生成的二进制文件。同样，您可以使用 BlazingText 托管使用 fastText 创建的模型二进制文件。

以下示例说明如何将 BlazingText 生成的模型与 fastText 结合使用：

```
#Download the model artifact from S3
aws s3 cp s3://<YOUR_S3_BUCKET>/<PREFIX>/model.tar.gz model.tar.gz

#Unzip the model archive
tar -xzf model.tar.gz

#Use the model archive with fastText
fasttext predict ./model.bin test.txt
```

但是，仅在 CPU 和单 GPU 上训练时支持二进制文件；在多 GPU 上训练时不会生成二进制文件。

有关数据集格式和模型托管的更多详细信息，请参阅示例笔记本使用 BlazingText 算法的文本分类、FastText 模型和使用 Word2Vec 算法生成子词嵌入。

BlazingText 算法的 EC2 实例建议

对于 `cbow` 和 `skipgram` 模式，BlazingText 支持单个 CPU 实例和单个 GPU 实例。这两种模式都支持学习 `subwords` 嵌入。为了在不影响准确性的情况下实现最高速度，我们建议您使用 ml.p3.2xlarge 实例。

对于 `batch_skipgram` 模式，BlazingText 支持单个或多个 CPU 实例。在多个实例上训练时，将传递给 `CreateTrainingJob` 的 `S3DataSource` 对象的 `S3DataDistributionType` 字段的值设置为 `FullyReplicated`。BlazingText 负责跨计算机分布数据。

对于指导式文本分类模式，如果训练数据集小于 2 GB，则建议使用 C5 实例。对于较大的数据集，请使用具有单个 GPU 的实例（ml.p2.xlarge 或 ml.p3.2xlarge）。

BlazingText 示例笔记本

有关使用 SageMaker BlazingText 算法训练和部署指导式二进制文件和多类分类模型的示例笔记本，请参阅。dbpedia 数据集上的炽热文本分类. 有关创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 在创建并打开笔记本实例后，请选择SageMaker 示例选项卡可查看所有 SageMaker 示例的列表。使用 Blazing Text 的主题建模示例笔记本位于 Amazon 算法简介部分中。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

BlazingText 超参数

在使用 `CreateTrainingJob` 请求开始训练作业时，可指定训练算法。您还可以指定算法特定的超参数作为字符串到字符串映射。BlazingText 算法的超参数取决于您使用的模式：Word2VEC（自主）和文本分类（指导式）。

Word2Vec 超参数

下表列出了 Amazon SageMaker 提供的 BlazingText Word2VEC 训练算法的超参数。

参数名称	描述
`mode`	用于训练的 Word2vec 架构。 必填 有效值：`batch_skipgram`、`skipgram` 或 `cbow`
`batch_size`	在 `mode` 设置为 `batch_skipgram` 时的每个批处理的大小。设置为一个介于 10 和 20 之间的数字。 可选 有效值：正整数 默认值：11
`buckets`	要用于子词的哈希存储桶的数量。 可选 有效值：正整数 默认值：2000000
`epochs`	传递训练数据的次数。 可选 有效值：正整数 默认值：5
`evaluation`	是否使用 WordSimilarity-353 测试来评估训练后的模型。 可选 有效值：（布尔值）`True`要么`False`

参数名称	描述
	默认值：`True`
`learning_rate`	用于参数更新的步长大小。 可选 有效值：float 默认值：0.05
`min_char`	要用于子词/支付 n-grams 的字符的最小数目。 可选 有效值：正整数 默认值：3
`min_count`	出现少于 `min_count` 次的单词将被丢弃。 可选 有效值：非负整数 默认值：5
`max_char`	要用于子词/支付 n-grams 的字符的最大数目 可选 有效值：正整数 默认值：6
`negative_samples`	负采样共享策略的负采样的数目。 可选 有效值：正整数 默认值：5
`sampling_threshold`	单词出现次数的阈值。对训练数据中出现频率较高的单词进行随机下采样。 可选 有效值：积极分数。建议的范围为 (0, 1e-3] 默认值：0.0001
`subwords`	是否学习子词嵌入。 可选 有效值：（布尔值）`True`要么`False` 默认值：`False`

参数名称	描述
vector_dim	该算法学习的单词向量的维度。 可选 有效值：正整数 默认值：100
window_size	上下文窗口的大小。上下文窗口是用于训练的目标单词周围的单词数。 可选 有效值：正整数 默认值：5

文本分类超参数

下表列出了 Amazon SageMaker 提供的文本分类训练算法的超参数。

> **Note**
>
> 虽然文本分类模式和 Word2Vec 模式之间的某些参数很常见，但根据上下文的不同，它们可能有不同的含义。

参数名称	描述
mode	训练模式。 必填 有效值：supervised
buckets	要用于单词 n-grams 的哈希存储桶的数量。 可选 有效值：正整数 默认值：2000000
early_stopping	是否在验证准确率在 patience 个纪元后未提高的情况下时停止训练。 可选 有效值：（布尔值）True要么False 默认值：False
epochs	完成传递训练数据的最大次数。 可选 有效值：正整数 默认值：5

参数名称	描述
learning_rate	用于参数更新的步长大小。 可选 有效值：float 默认值：0.05
min_count	出现少于 min_count 次的单词将被丢弃。 可选 有效值：非负整数 默认值：5
min_epochs	调用提前停止逻辑之前训练的最小纪元数。 可选 有效值：正整数 默认值：5
patience	当验证集没有任何进展时，在应用提前停止之前等待的纪元数。仅当 early_stopping 为 True 时使用。 可选 有效值：正整数 默认值：4
vector_dim	嵌入层的维度。 可选 有效值：正整数 默认值：100
word_ngrams	要使用的单词 n-gram 特征的数量。 可选 有效值：正整数 默认值：2

调整 BlazingText 模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

BlazingText 算法计算的指标

BlazingText Word2VEC 算法 (`skipgram`、`cbow`, 和`batch_skipgramMode`）在训练期间报告单个指标：`train:mean_rho`. 此指标的计算时间是在WS-353 单词相似度数据集. 在优化 Word2Vec 算法的超参数值时，请使用此指标作为目标。

BlazingText 文本分类算法（`supervised` 模式）也在训练期间报告单个指标：`validation:accuracy`。在优化文本分类算法的超参数值时，请使用这些指标作为目标。

指标名称	描述	优化方向
`train:mean_rho`	WS-353 单词相似性数据集的均值 rho（Spearman 的秩相关系数）	最大化
`validation:accuracy`	用户指定的验证数据集的分类准确率	最大化

可优化的 BlazingText 超参数

Word2Vec 算法的可优化超参数

使用以下超参数优化 Amazon SageMaker BlazingText Word2VEC 模型。对 Word2Vec 目标指标影响最大的超参数为：`mode`、 `learning_rate`、`window_size`、`vector_dim` 和 `negative_samples`。

参数名称	参数类型	建议的范围或值
`batch_size`	`IntegerParameterRange`	[8-32]
`epochs`	`IntegerParameterRange`	[5-15]
`learning_rate`	`ContinuousParameterRange`	MinValue：0.005，MaxValue：0.01
`min_count`	`IntegerParameterRange`	[0-100]
`mode`	`CategoricalParameterRange`	`['batch_skipgram', 'skipgram', 'cbow']`
`negative_samples`	`IntegerParameterRange`	[5-25]
`sampling_threshold`	`ContinuousParameterRange`	MinValue：0.0001，MaxValue：0.001
`vector_dim`	`IntegerParameterRange`	[32-300]
`window_size`	`IntegerParameterRange`	[1-10]

文本分类算法的可优化超参数

使用以下优化 Amazon SageMaker BlazingText 文本分类模型：超参数.

参数名称	参数类型	建议的范围或值
`buckets`	`IntegerParameterRange`	[1000000-10000000]
`epochs`	`IntegerParameterRange`	[5-15]

参数名称	参数类型	建议的范围或值
learning_rate	ContinuousParameterRange	MinValue：0.005，MaxValue：0.01
min_count	IntegerParameterRange	[0-100]
mode	CategoricalParameterRange	['supervised']
vector_dim	IntegerParameterRange	[32-300]
word_ngrams	IntegerParameterRange	[1-3]

DeepAR 预测算法

Amazon SageMaker DeepAR 预测算法是一种自主学习算法，使用递归神经网络 (RNN) 来预测标量（一维）时间序列。经典预测方法，如自回归积分滑动平均模型 (ARIMA) 或指数平滑法 (ETS)，将一个模型拟合到各个单独的时间序列。然后，他们使用该模型将时间序列推断到未来。

但是，在很多应用中，您有跨一组具有代表性单元的多个相似时间序列。例如，您可能会针对不同产品的需求、服务器负载和网页请求进行时间序列分组。对于此类应用程序，您可以从所有时间序列中联合训练单个模型中受益。DeepAR 采用这种方法。当数据集包含数百个相关时间序列时，DeepAR 将优于标准 ARIMA 和 ETS 方法。您还可以使用训练过的模型来生成与已训练过的时间序列相似的新时间序列的预测。

DeepAR 算法的训练输入为一个或多个（最好是多个）`target`由同一进程或类似进程生成的时间序列。该算法基于此输入数据集训练一个模型，该模型可以学习此过程/流程的近似值，并使用它来预测目标时间序列的演变情况。每个目标时间序列都可以选择性地与`cat`字段和动态（时间相关）时间序列矢量由`dynamic_feat`字段中返回的子位置类型。SageMaker 通过从训练数据集中的每个目标时间序列中随机抽取训练示例来训练 DeepAR 模型。每个训练示例包括一对具有固定的预定义长度的相邻上下文和预测窗口。要控制网络在过去可以往前看多长时间，请使用`context_length`超级参数。要控制未来可以做出预测的距离，请使用`prediction_length`超级参数。有关更多信息，请参阅 DeepAR 算法的工作原理 (p. 1601)。

主题

- DeepAR 算法的输入/输出接口 (p. 1598)
- 使用 DeepAR 算法的最佳实践 (p. 1600)
- DeepAR 算法的 EC2 实例建议 (p. 1601)
- DeepAR 示例笔记本 (p. 1601)
- DeepAR 算法的工作原理 (p. 1601)
- DeepAR 超参数 (p. 1603)
- 优化 DeepAR 模型 (p. 1607)
- DeepAR 推理格式 (p. 1608)

DeepAR 算法的输入/输出接口

DeepAR 支持两种数据通道。必需的`train`频道描述训练数据集。可选的`test`channel 描述了算法在训练后用于评估模型准确度的数据集。您可以在中提供培训和测试数据集JSON 行格式的日期和时间。文件可以在 gzip 中或Parquet文件格式。

指定训练和测试数据的路径时，可以指定一个文件或包含多个文件的目录，这些文件可以存储在子目录中。如果指定目录，DeepAR 将使用目录中的所有文件作为相应频道的输入，但以句点 (.) 开头的文件和名为的文件除外_成功. 这可以确保您可以直接使用 Spark 作业生成的输出文件夹作为 DeepAR 培训作业的输入渠道。

默认情况下，DeepAR 模型从文件扩展名 () 确定输入格式。`.json`、`.json.gz`，或者`.parquet`) 在指定的输入路径中。如果路径未以其中一个扩展名结束，则必须在适用于 Python 的 SDK 中明确指定格式。使用`content_type`的参数s3_ 输入类。

输入文件中的记录应包含以下字段：

- start一个格式为的字符串YYYY-MM-DD HH:MM:SS. 开始时间戳不能包含时区信息。
- target— 表示时间序列的浮点值或整数的数组。你可以将缺失的值编码为null文字，或者作为"NaN"JSON 中的字符串，或者nan实木复合地板中的浮点值。
- dynamic_feat（可选）-表示自定义特征时间序列（动态特征）向量的浮点值或整数的数组。如果设置此字段，则所有记录必须具有相同数量的内部数组（相同数量的要素时间序列）。此外，每个内部数组的长度必须与关联数组的长度相同target值加上下文的长度。特征中不支持缺少的值。例如，如果目标时间序列代表不同产品的需求，那么关联的dynamic_feat可能是布尔时间序列，表示促销是否应用于特定商品 (1) (0)：

```
{"start": ..., "target": [1, 5, 10, 2], "dynamic_feat": [[0, 1, 1, 0]]}
```

- cat（可选）-可用于对记录所属的组进行编码的分类特征的数组。分类要素必须编码为从 0 开始的正整数序列。例如，分类域名 {R, G, B} 可以编码为 {0, 1, 2}。训练数据集中必须表示每个类别域的所有值。这是因为 DeepAR 算法只能对训练期间观察到的类别进行预测。而且，每个分类要素都嵌入在低维空间中，其维度由embedding_dimension超级参数。有关更多信息，请参阅 DeepAR 超参数 (p. 1603)。

如果您使用 JSON 文件，它必须位于JSON 行格式的日期和时间。例如：

```
{"start": "2009-11-01 00:00:00", "target": [4.3, "NaN", 5.1, ...], "cat": [0, 1],
 "dynamic_feat": [[1.1, 1.2, 0.5, ...]]}
{"start": "2012-01-30 00:00:00", "target": [1.0, -5.0, ...], "cat": [2, 3], "dynamic_feat":
 [[1.1, 2.05, ...]]}
{"start": "1999-01-30 00:00:00", "target": [2.0, 1.0], "cat": [1, 4], "dynamic_feat":
 [[1.3, 0.4]]}
```

在此示例中，每个时间序列都有两个关联的分类特征和一个时间序列特征。

对于 Parquet，您使用相同的三个字段作为列。此外，"start" 可以是 datetime 类型。您可以使用 gzip (gzip) 或 Snappy 压缩库 (snappy) 来压缩 Parquet 文件。

如果在无 cat 和 dynamic_feat 字段情况下训练该算法，则它学习一个“全局”模型，即跟推理时的目标时间序列的具体身份无关而仅以其形状为条件的模型。

如果模型基于cat和dynamic_feat为每个时间序列提供的要素数据，预测可能会受到时间序列的性质的影响，并且相应cat功能。例如，如果target时间序列代表服装物品的需求，你可以关联二维cat用于编码第一个组件中物品类型（例如 0 = 鞋子，1 = 连衣裙）以及第二个组件中物品的颜色（例如 0 = 红色，1 = 蓝色）的矢量。示例输入将如下所示：

```
{ "start": ..., "target": ..., "cat": [0, 0], ... } # red shoes
{ "start": ..., "target": ..., "cat": [1, 1], ... } # blue dress
```

在推理时，您可以通过以下方式请求对目标进行预测cat是组合的值cat训练数据中观察到的值，例如：

```
{ "start": ..., "target": ..., "cat": [0, 1], ... } # blue shoes
{ "start": ..., "target": ..., "cat": [1, 0], ... } # red dress
```

以下准则适用于训练数据：

- 时间序列的开始时间和长度可能会有所不同。例如，在营销中，产品通常在不同的日期进入零售目录，因此它们的开始日期自然不同。但是，所有序列必须具有相同的频率、分类特征数和动态特征的数量。
- 根据文件中时间序列的位置将训练文件随机排序。换而言之，时间序列在文件中以随机顺序出现。
- 确保将start字段正确。该算法使用start用于派生内部要素的时间戳。

- 如果您使用分类特征 (`cat`），所有时间序列必须具有相同数目的分类特征。如果数据集包含`cat`字段中，算法使用它并从数据集中提取组的基数。默认情况下，`cardinality`是`"auto"`. 如果数据集包含`cat`字段，但您不想使用它，可以通过设置禁用它`cardinality`到`""`. 如果模型是使用`cat`功能，你必须包含它才能推断。
- 如果你的数据集包含`dynamic_feat`字段中，算法会自动使用它。所有时间序列必须具有相同数量的要素时间序列。每个要素时间序列中的时间点与目标中的时间点一对应。此外，中的条目`dynamic_feat`字段的长度应与`target`. 如果数据集包含`dynamic_feat`字段，但您不想使用它，请通过设置 (`num_dynamic_feat`到`""`)。如果模型是用`dynamic_feat`字段中，您必须提供此字段以便推理。此外，每个功能都必须有所提供目标的长度加上`prediction_length`. 换句话说，您必须在将来提供特征值。

如果您指定可选的测试通道数据，DeepAR 算法使用不同的准确性指标评估训练后的模型。该算法通过以下方式计算测试数据上的均方根误差 (RMSE)：

$$\text{RMSE} = \sqrt{\frac{1}{nT}\sum_{i,t}\left(\hat{y}_{i,t} - y_{i,t}\right)^2}$$

$y_{i,t}$是时间序列的真正价值i当时t.$\hat{y}_{i,t}$是平均预测。总和结束了n测试集中的时间序列和每个时间序列的最后时间点，其中对应于预测展望期。您可以通过将`prediction_length`超级参数。有关更多信息，请参阅DeepAR 超参数 (p. 1603)。

此外，该算法使用加权分位数损失评估预测分布的准确性。对于范围为 [0, 1] 的分位数，加权分位数损失定义如下：

$$\text{wQuantileLoss}[\tau] = 2\frac{\sum_{i,t} Q_{i,t}^{(\tau)}}{\sum_{i,t}|y_{i,t}|}, \quad \text{with} \quad Q_{i,t}^{(\tau)} = \begin{cases} (1-\tau)|q_{i,t}^{(\tau)} - y_{i,t}| & \text{if } q_{i,t}^{(\tau)} > y_{i,t} \\ \tau|q_{i,t}^{(\tau)} - y_{i,t}| & \text{otherwise} \end{cases}$$

$Q_{i,t}^{(\tau)}$模型预测分布的 τ 分位数。要指定要计算损失的分位数，请将`test_quantiles`超级参数。除此之外，还报告了规定的分位数损失的平均值，作为训练记录的一部分。有关信息，请参阅DeepAR 超参数 (p. 1603)。

为了推断，DeepAR 接受 JSON 格式和以下字段：

- `"instances"`，其中包括 JSON Line 格式的一个或多个时间序列
- 的名字`"configuration"`，其中包括生成预测的参数

有关更多信息，请参阅 DeepAR 推理格式 (p. 1608)。

使用 DeepAR 算法的最佳实践

准备时间序列数据时，请遵循以下最佳实践以获得最佳结果：

- 除了拆分数据集以进行训练和测试之外，始终提供用于训练、测试和调用模型进行推理时的整个时间序列。无论你怎么设置`context_length`不要分解时间序列或仅提供时间序列的一部分。该模型使用的数据点比中设置的值更早`context_length`对于滞后值功能。
- 当优化 DeepAR 模型时，可以拆分数据集以创建训练数据集和测试数据集。在典型的评估中，您将采用训练所用的相同时间序列，但在未来测试模型。`prediction_length`在训练期间可见的最后一个时间点之后紧跟的时间点。您可以使用整个数据集（可用的所有时间序列的完整长度）作为测试集并删除最后个数据集，从而创建和测试满足此条件的训练和测试数据集。`prediction_length`每个时间序列的积分用于训练。在训练期间，模型将看不到测试期间评估它的时间点的目标值。在测试过程中，算法会扣留最后一个`prediction_length`测试集中每个时间序列的点数并生成预测。然后它将预测与预扣值进行比较。您可以在测试集中多次重复时间序列，但在不同的终端节点处切割它们，从而创建更复杂的评估。通过此方法，准确率指标将与来自不同时间点的多个预测平均。有关更多信息，请参阅 优化 DeepAR 模型 (p. 1607)。

- 避免对使用非常大的值 (>400)`prediction_length`因为它使模型变慢而且不那么准确。如果您想进一步预测将来的情况，请考虑以较低的频率聚合数据。例如，使用 `5min` 而不是 `1min`。
- 由于使用滞后，所以模型在时间序列中可以比为指定的值进一步回顾`context_length`. 因此，您不需要将此参数设置为较大的值。我们建议从您使用的值开始。`prediction_length`.
- 我们建议您在尽可能多的时间序列上训练 DeepAR 模型。尽管在单个时间序列上训练的 DeepAR 模型可能正常工作，但标准预测算法（如 ARIMA 或 ETS）可能提供更准确的结果。当数据集包含数百个相关时间序列时，DeepAR 算法便开始优于标准方法。目前，DeepAR 要求所有训练时间序列中可用的总观察数至少为 300。

DeepAR 算法的 EC2 实例建议

您可以在 GPU 和 CPU 实例上，在单机器和多机器设置中训练 DeepAR。我们建议您从单 CPU 实例开始（例如，ml.c4.2xlarge 或 ml.c4.4xlarge），并仅在必要时切换到 GPU 实例和多机器。使用 GPU 和多台机器只能提高大型模型（每层有多个单元和具有多个层）以及大量小批量大小（例如，大于 512）的吞吐量。

为了推断，DeepAR 仅支持 CPU 实例。

为指定较大的值`context_length`、`prediction_length`、`num_cells`、`num_layers`，或者`mini_batch_size`可以创建对于小型实例来说太大的模型。在这种情况下，请使用较大的实例类型或减少这些参数的值。运行超参数调整作业时也经常出现此问题。在这种情况下，请使用足够大的实例类型来进行模型调整任务，并考虑限制关键参数的上限值以避免作业失败。

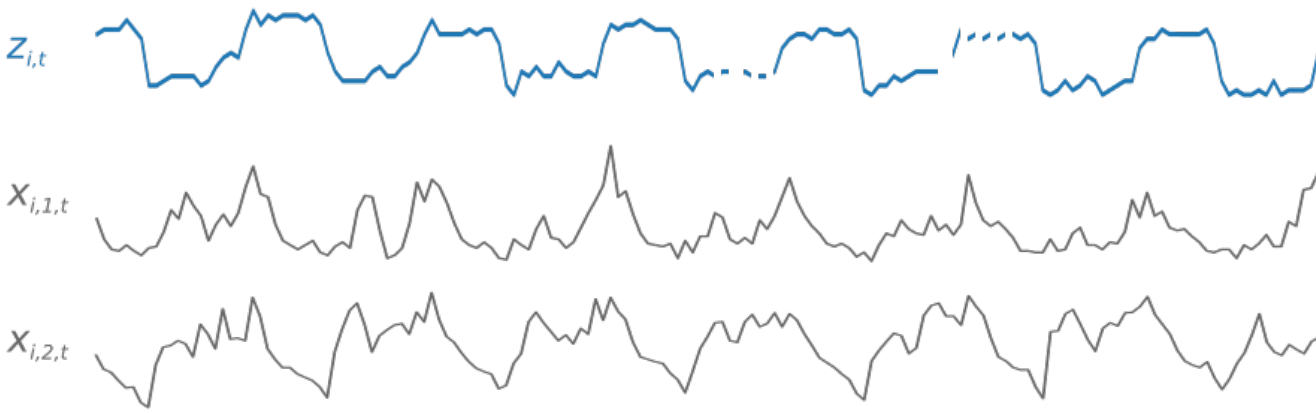
DeepAR 示例笔记本

有关示例笔记本，该笔记本显示如何准备用于训练 SageMaker DeepAR 算法的时间序列数据集以及如何部署训练后的模型以执行推理，请参阅使用 DeepAR 进行时序预测-综合数据以及关于电力数据集的 DeepAR 演示，其中说明了 DeepAR 在真实世界数据集上的高级功能。有关创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 在创建并打开笔记本实例后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

DeepAR 算法的工作原理

在训练期间，DeepAR 接受训练数据集和可选的测试数据集。它使用测试数据集评估训练后的模型。通常，数据集不必包含相同的时间序列集。您可以使用在给定训练集上训练的模型来生成训练集中时间序列的未来以及其他时间序列的预测。训练数据集和测试数据集都由一个或多个（最好是多个）目标时间序列组成。可以选择将每个目标时间序列与特征时间序列向量和分类特征向量关联。有关更多信息，请参阅 DeepAR 算法的输入/输出接口 (p. 1598)。

例如，以下是索引所用的训练集的一个元素。i其中包括一个目标时间序列，$z_{i,t}$，以及两个相关的要素时间序列，$x_{i,1,t}$和$x_{i,2,t}$：

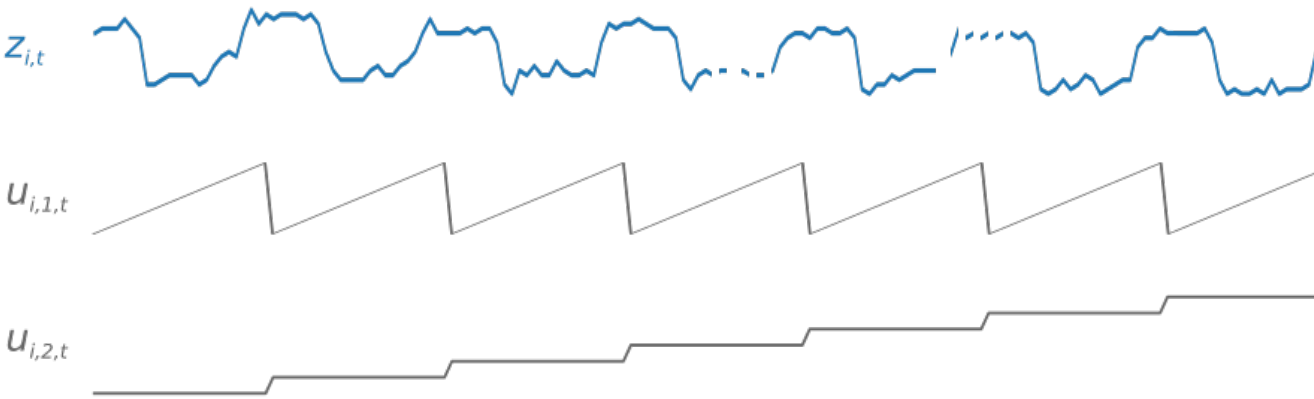


目标时间序列可能包含缺失值，这些值由时间系列中的换行符表示。DeepAR 仅支持将来已知的特征时间序列。这允许您运行 “假设” 场景。例如，如果我以某种方式改变产品价格，会发生什么？

每个目标时间序列也可以与大量分类特征关联。您可以使用这些功能对时间序列所属的分组进行编码。使用分类特征允许模型学习组的典型行为，可用于提高模型准确性。DeepAR 通过学习每个组的嵌入向量来实现这一点，该嵌入向量捕获组中所有时间序列的公共属性。

DeepAR 算法中要素时间序列的工作原理

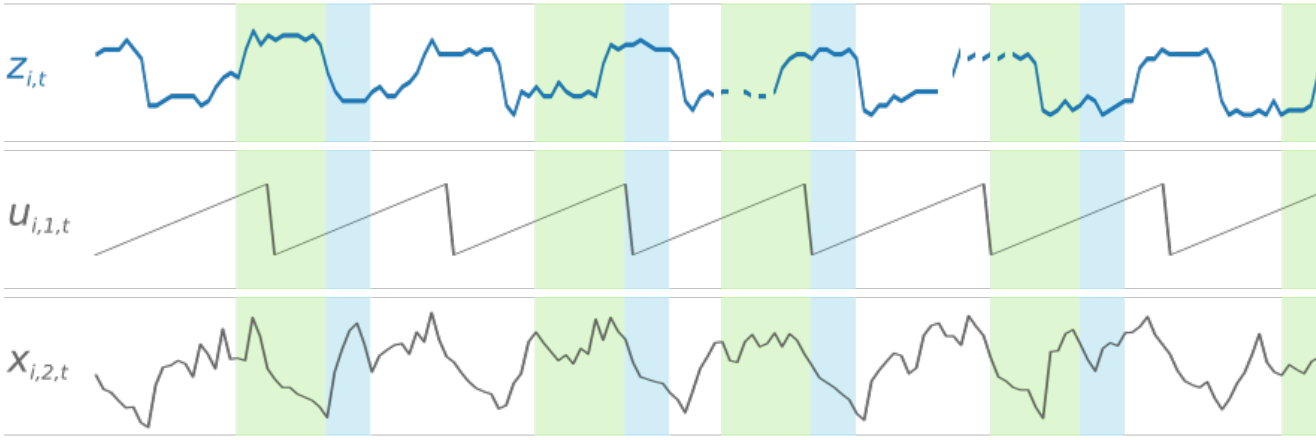
为了便于学习与时间相关的模式（如周末的峰值），DeepAR 将根据目标时间序列的频率自动创建特征时间序列。例如，DeepAR 为每周时间序列频率创建两个特征时间序列（月中的某天和一年中的某天）。它将这些派生的特征时间序列与您在训练和推理期间提供的自定义特征时间序列一起使用。下图显示了这些派生的时间序列功能中的两个：$U_{i,1,t}$代表一天中的小时$U_{i,2,t}$一星期中的日子。



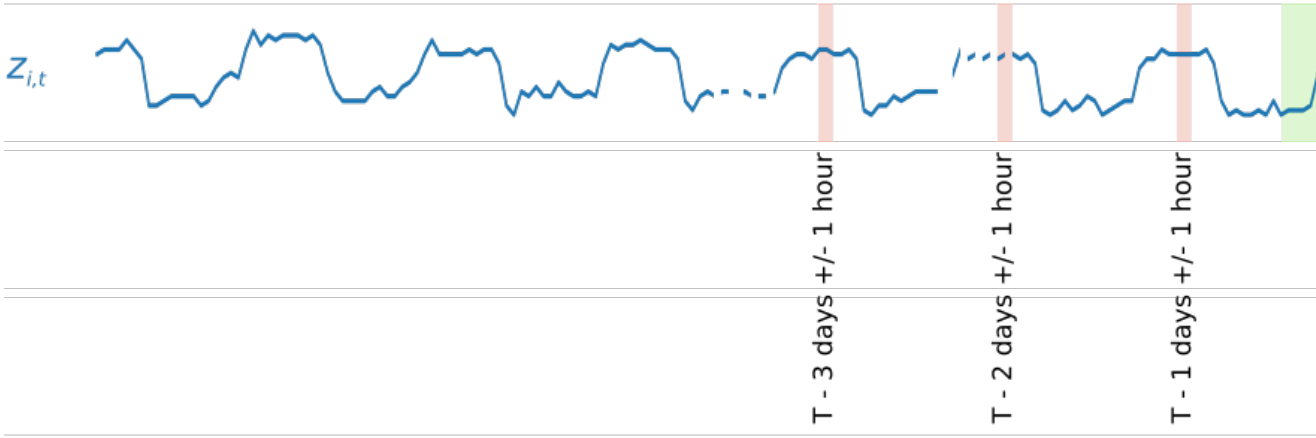
DeepAR 算法会自动生成这些要素时间序列。下表列出了支持的基本时间频率的派生特征。

时间序列的频率	派生的特征
`Minute`	`minute-of-hour`, `hour-of-day`, `day-of-week`, `day-of-month`, `day-of-year`
`Hour`	`hour-of-day`, `day-of-week`, `day-of-month`, `day-of-year`
`Day`	`day-of-week`, `day-of-month`, `day-of-year`
`Week`	`day-of-month`, `week-of-year`
`Month`	一年中的某天

DeepAR 通过从训练数据集中的每个时间序列中随机抽取多个训练示例来训练模型。每个训练示例包括一对具有固定的预定义长度的相邻上下文和预测窗口。这些区域有：`context_length`超参数控制网络可以往前看多长时间，`prediction_length`超参数控制可以往后预测多长时间。在训练过程中，该算法会忽略包含小于指定预测长度的时间序列的训练集元素。下图代表了从元素中抽取的上下文长度为 12 小时，预测长度为 6 小时的五个样本。i. 为简洁起见，我们省略了特征时间序列。$x_{i,1,t}$和$U_{i,2,t}$.



为了捕获季节性模式，DeepAR 还自动提供目标时间序列中的滞后值。在每小时频率的示例中，对于每个时间指数，t = T，该模型暴露了$z_{i,t}$值，过去大约 1 天、2 天和 3 天执行此操作一次。



为了推理，训练后的模型获取输入目标时间序列 (这些时间序列可能用于也可能未用于训练)，并预测下一个模型的概率分布。`prediction_length`值。由于 DeepAR 是针对整个数据集进行训练的，因此，预测会考虑从类似时间序列中学习的模式。

有关 DeepAR 背后数学的信息，请参阅DeepAR：利用自回归循环网络进行概率预测.

DeepAR 超参数

参数名称	描述
`context_length`	在进行预测之前，模型获取以查看的时间点数量。此参数的值应该与 `prediction_length` 大致相同。此模型还接收来自目标的滞后输入，因此 `context_length` 可以比典型的季节性小得多。例如，每日时间序列可以具有每年的季节性。模型自动包括一年的滞后，因此上下文长度可以短于一年。模型选取的滞后值取决于时间序列的频率。例如，对于每日频率，滞后值是前 1 周、前 2 周、前 3 周、前 4 周和前 1 年。 必填 有效值：正整数

参数名称	描述
epochs	扫描训练数据的最大次数。最佳值取决于您的数据大小和学习率。另请参阅 early_stopping_patience。典型值范围为 10 到 1000。 必填 有效值：正整数
prediction_length	训练模型来预测的时间步长数，也称为预测期。训练后的模型始终生成此长度的预测。它无法生成更长的预测。这些区域有：prediction_length在训练模型时是固定的，以后无法更改模型。 必填 有效值：正整数
time_freq	数据集中时间序列的粒度。使用time_freq选择适当的日期特征和滞后。该模型支持以下基本频率。它还支持这些基本频率的倍数。例如，5min指定 5 分钟的频率。 • M：每月 • W：每周 • D：每天 • H：每小时 • min：每分钟 必填 有效值：后面是一个整数M、W、D、H，或者分钟. 例如，5min.
cardinality	使用分类功能时（cat),cardinality是一个数组，指定每个分类特征的类别（组）数。将此设置为auto从数据中推断基数。这些区域有：auto当数据集中未使用分类要素时，模式也可用。这是参数的推荐设置。 将基数设置为ignore强制 DeepAR 不使用分类功能，即使它们存在于数据中。 要执行额外的数据验证，可以将此参数显式设置为实际值。例如，如果提供了两个类别要素，其中第一个类别要素有 2，另一个有 3 个可能的值，则将其设置为 [2, 3]。 有关如何使用分类功能的更多信息，请参阅 DeepAR 主文档页面上的数据部分。 可选 有效值：auto、ignore、正整数组、空字符串或 默认值：auto

参数名称	描述
dropout_rate	训练期间使用的丢弃比率。该模型使用 zoneout 正则化。对于每次迭代，不更新隐藏神经元的随机子集。典型值小于 0.2。 可选 有效值：浮点值 默认值：0.1
early_stopping_patience	如果设置此参数，训练在指定的数量内没有取得进展时停止。epochs. 返回具有最低损失的模型作为最后的模型。 可选 有效值：整数
embedding_dimension	每个分类特征学习的嵌入向量的大小（所有分类特征使用相同的值）。 在提供类别分组功能时，DeepAR 模型可以学习组级别的时间序列模式。为此，模型学习大小为的嵌入向量。embedding_dimension捕获组中所有时间序列的公共属性。较大的 embedding_dimension 允许模式捕获更复杂的模式。但是，因为增加embedding_dimension增加模型中的参数数量，准确学习这些参数需要更多训练数据。此参数的典型值在 10-100 之间。 可选 有效值：正整数 默认值：10
learning_rate	训练中使用的学习率。典型值范围从 1e-4 到 1e-1。 可选 有效值：浮点值 默认值：1e-3

参数名称	描述
likelihood	模型生成一个概率预测，并可以提供分布的分位数和返回样本。根据您的数据，选择用于不确定性估算的适当可能性 (噪声模型)。可以选择以下可能性： • 高斯：用于实值数据。 • 测试版：用于 0 和 1 之间（含 0 和 1）的实值目标。 • 负值-二项式：用于计数数据（非负整数）。 • T 学生：对突发式数据来说非常适合突发式数据的实值数据的替代。 • 确定性-L1：一个损失函数，它不估算不确定性，只学习点预测。 可选 有效值：其中之一高斯、测试版、负值-二项式、T 学生，或者确定性-L1. 默认值：student-T
mini_batch_size	训练期间使用的小型批处理的大小。典型值范围为 32 到 512。 可选 有效值：正整数 默认值：128
num_cells	RNN 的各个隐藏层中使用的单元数。典型值范围为 30 到 100。 可选 有效值：正整数 默认值：40
num_dynamic_feat	的数量dynamic_feat数据中提供。将此设置为auto从数据推断动态特征的数量。这些区域有：auto当数据集中未使用动态要素时，模式也可用。这是参数的推荐设置。 要强制 DeepAR 不使用动态功能（即使它们存在于数据中），请设置num_dynamic_feat到ignore. 要执行额外的数据验证，可以将此参数显式设置为实际整数值。例如，如果提供了两个动态特征，则将它设置为 2。 可选 有效值：auto、ignore、正整数或空字符串 默认值：auto

参数名称	描述
num_eval_samples	计算测试准确度指标时，每个时间序列使用的样本数量。此参数对训练或最终模型没有任何影响。特别是，可以使用不同数量的示例查询模型。此参数仅影响训练后测试频道上报告的准确率分数。值越小，评估越快，但评估分数通常更差且更不确定。当使用更高的分位数（例如 0.95）进行评估时，增加评估样本的数量可能非常重要。 可选 有效值：整数 默认值：100
num_layers	RNN 中的隐藏层数。典型值范围为 1 到 4。 可选 有效值：正整数 默认值：2
test_quantiles	用于计算测试通道上的分位数损失的分位数。 可选 有效值：浮点数数组 默认值：[0.1、0.2、0.3、0.4、0.5、0.6、0.7、0.8、0.9]

优化 DeepAR 模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

DeepAR 算法计算的指标

DeepAR 算法报告在训练期间计算的三个指标。在优化模型时，请选择其中之一作为目标。对于目标，请使用提供的测试频道上的预测准确度（推荐）或训练损失。有关 DeepAR 算法的训练/测试拆分的建议，请参阅使用 DeepAR 算法的最佳实践 (p. 1600).

指标名称	描述	优化方向
test:RMSE	测试集上计算的预测与实际目标之间的均方根误差。	最小化
test:mean_wQuantileLoss	测试集中计算出的平均总体分位数损失。要控制使用哪些分位数，请将test_quantiles超级参数。	最小化
train:final_loss	训练负对数似然损失在该模型的上一个训练时期的平均值。	最小化

DeepAR 算法的可优化超参数

使用以下超参数优化 DeepAR 模型。对 DeepAR 目标指标影响最大的超参数为：epochs、context_length、mini_batch_size、learning_rate, 和num_cells.

参数名称	参数类型	建议的范围
epochs	IntegerParameterRanges	MinValue：1、MaxValue：1000
context_length	IntegerParameterRanges	MinValue：1、MaxValue：200
mini_batch_size	IntegerParameterRanges	MinValue：32，MaxValue：1028
learning_rate	ContinuousParameterRange	最 MinValue：1e-5，MaxValue：1e-1
num_cells	IntegerParameterRanges	MinValue：30、MaxValue：200
num_layers	IntegerParameterRanges	MinValue：1、MaxValue：8
dropout_rate	ContinuousParameterRange	MinValue：0.00，MaxValue：0.2
embedding_dimension	IntegerParameterRanges	MinValue：1、MaxValue：50

DeepAR 推理格式

DeepAR JSON 请求格式

使用模型的终端节点查询经过训练的模型。终端节点采用以下 JSON 请求格式。

在请求中，instances字段对应于应由模型预测的时间序列。

如果模型已使用类别训练，您必须提供cat针对每个实例。如果在未使用cat字段中，应省略它。

如果模型是使用自定义要素时间序列训练的 (dynamic_feat)，你必须提供相同数量的dynamic_feat每个实例的值。他们每个人都应该有给出的长度length(target) + prediction_length，最后一次prediction_length值与将要预测的未来时间点相对应。如果模型在没有自定义要素时间序列的情况下进行了训练，则该字段不应包括在请求中。

```
{
    "instances": [
        {
            "start": "2009-11-01 00:00:00",
            "target": [4.0, 10.0, "NaN", 100.0, 113.0],
            "cat": [0, 1],
            "dynamic_feat": [[1.0, 1.1, 2.1, 0.5, 3.1, 4.1, 1.2, 5.0, ...]]
        },
        {
            "start": "2012-01-30",
            "target": [1.0],
            "cat": [2, 1],
            "dynamic_feat": [[2.0, 3.1, 4.5, 1.5, 1.8, 3.2, 0.1, 3.0, ...]]
        },
```

```
        {
            "start": "1999-01-30",
            "target": [2.0, 1.0],
            "cat": [1, 3],
            "dynamic_feat": [[1.0, 0.1, -2.5, 0.3, 2.0, -1.2, -0.1, -3.0, ...]]
        }
    ],
    "configuration": {
         "num_samples": 50,
         "output_types": ["mean", "quantiles", "samples"],
         "quantiles": ["0.5", "0.9"]
    }
}
```

这些区域有：configuration字段为可选项。configuration.num_samples设置模型生成的示例路径数，用于估算平均值和分位数。configuration.output_types描述在请求中返回的信息。有效值为"mean" "quantiles"和"samples". 如果你指定"quantiles"，中的每个分位数值configuration.quantiles将作为时间序列返回。如果您指定 "samples"，则模型还将返回用于计算其他输出的原始样本。

DeepAR JSON 响应格式

以下是响应的格式，其中 [...] 是数字数组：

```
{
    "predictions": [
        {
            "quantiles": {
                "0.9": [...],
                "0.5": [...]
            },
            "samples": [...],
            "mean": [...]
        },
        {
            "quantiles": {
                "0.9": [...],
                "0.5": [...]
            },
            "samples": [...],
            "mean": [...]
        },
        {
            "quantiles": {
                "0.9": [...],
                "0.5": [...]
            },
            "samples": [...],
            "mean": [...]
        }
    ]
}
```

DeepAR 的响应超时时间为 60 秒。在单个请求中传递多个时间序列时，预测将按顺序生成。由于每个时间序列的预测通常需要大约 300 至 1000 毫秒或更长时间，具体取决于模型大小，因此在单个请求中传递太多的时间序列可能会导致超时。最好是减少每个请求发送的时间序列并发送更多的请求。由于 DeepAR 算法每个实例使用多个工作线程，因此您可以通过并行发送多个请求来实现更高的吞吐量。

默认情况下，如果每个 CPU 有足够的内存，DeepAR 每个 CPU 使用一个工作线程进行推理。如果模型很大，并且没有足够的内存来在每个 CPU 上运行模型，则会减少工作线程的数量。可以使用环境变量覆盖用于推理的工件数MODEL_SERVER_WORKERS例如，通过设置MODEL_SERVER_WORKERS=1) 调用 SageMaker 时CreateModelAPI。

批量转换与 DeepAR 算法

DeepAR 预测支持使用 JSON 行格式使用数据进行批量转换来获取推断。在此格式中，每条记录在一个行上表示为 JSON 对象，并且行由换行符分隔。格式与用于模型训练的 JSON Line 格式相同。有关信息，请参阅DeepAR 算法的输入/输出接口 (p. 1598)。例如：

```
{"start": "2009-11-01 00:00:00", "target": [4.3, "NaN", 5.1, ...], "cat": [0, 1],
 "dynamic_feat": [[1.1, 1.2, 0.5, ..]]}
{"start": "2012-01-30 00:00:00", "target": [1.0, -5.0, ...], "cat": [2, 3], "dynamic_feat":
 [[1.1, 2.05, ...]]}
{"start": "1999-01-30 00:00:00", "target": [2.0, 1.0], "cat": [1, 4], "dynamic_feat":
 [[1.3, 0.4]]}
```

> **Note**
>
> 在使用 CreateTransformJob 创建转换作业时，将 BatchStrategy 值设置为 SingleRecord 并将TransformInput 配置中的 SplitType 值设置为 Line，因为默认值当前会导致运行时故障。

与托管终端节点推理请求格式类似，如果满足以下两个条件，则每个实例的 cat 和 dynamic_feat 字段都是必填的：

- 在数据集上训练模型，该数据集同时包含cat和dynamic_feat字段之间没有不同。
- 相应的cardinality和num_dynamic_feat训练作业中使用的值未设置为"".

与托管终端节点推理不同，使用名为 DEEPAR_INFERENCE_CONFIG 的环境变量为整个批量推理作业设置一次配置字段。在通过调用 CreateTransformJob API 创建模型时，可以传递 DEEPAR_INFERENCE_CONFIG 的值。如果容器环境中缺少 DEEPAR_INFERENCE_CONFIG，则推理容器使用以下默认值：

```
{
    "num_samples": 100,
    "output_types": ["mean", "quantiles"],
    "quantiles": ["0.1", "0.2", "0.3", "0.4", "0.5", "0.6", "0.7", "0.8", "0.9"]
}
```

输出也采用 JSON 行格式，每个预测对应一行，其顺序与相应输入文件中的实例顺序相同。预测被编码为与在线推理模式下响应返回的对象相同的对象。例如：

```
{ "quantiles": { "0.1": [...], "0.2": [...] }, "samples": [...], "mean": [...] }
```

请注意，在TransformInputSageMaker 的配置CreateTransformJob请求客户端必须明确设置AssembleWith的值Line，作为默认值None在同一行中连接所有 JSON 对象。

例如，这是 SageMakerCreateTransformJob使用自定义请求 DeepAR 任务DEEPAR_INFERENCE_CONFIG：

```
{
   "BatchStrategy": "SingleRecord",
   "Environment": {
      "DEEPAR_INFERENCE_CONFIG" : "{ \"num_samples\": 200, \"output_types\": [\"mean\"] }",
      ...
   },
   "TransformInput": {
      "SplitType": "Line",
      ...
   },
   "TransformOutput": {
```

```
      "AssembleWith": "Line",
      ...
   },
   ...
}
```

因子分解机算法

因子分解机算法是通用的受监管机器学习算法，适用于分类和回归任务。它是线性模型的扩展，旨在经济地捕获高维稀疏数据集中的各特征之间的交互。例如，在一个点击预测系统中，因式分解机模型可以捕获当特定广告类别的广告放置在特定页面类别的页面上时所观察到的点击率模式。对于处理高维稀疏数据集的任务(如点击预测和项目建议)，因子分解机是不错的选择。

> **Note**
>
> Amazon SageMaker 实施因式分解机算法时仅考虑功能之间的两两两 (二阶) 交互。

主题

- 因子分解机算法的输入/输出接口 (p. 1611)
- 因子分解机算法的 EC2 实例建议 (p. 1611)
- 因子分解机示例笔记本 (p. 1611)
- 因子分解机的工作原理 (p. 1612)
- 因子分解机超参数 (p. 1612)
- 调整因子分解机模型 (p. 1617)
- 因子分解机响应格式 (p. 1619)

因子分解机算法的输入/输出接口

因子分解机算法可在二进制分类模式或回归模式下运行。在每种模式下，可以向测试通道提供数据集以及训练通道数据集。评分取决于使用的模式。在回归模式下，使用均方根误差 (RMSE) 对测试数据集计分。在二元分类模式下，使用二元交叉熵 (记录丢失)、准确度 (阈值=0.5) 和 F1 分数 (阈值=0.5) 对测试数据集计分。

适用于培训，分解机算法目前仅支持`recordIO-protobuf`使用的格式`Float32`张量。由于使用案例主要针对稀疏数据，`CSV` 并不是合适选项。recordIO 包装的 protobuf 支持文件和管道模式训练。

适用于推断，分解机算法支持`application/json`和`x-recordio-protobuf`格式。

- 对于二元分类问题，该算法预测分数和标签。标签是一个数字，可以是 `0` 或 `1`。分数是一个数字，它表示该算法认为标签应该为 `1` 的强烈程度。该算法先计算分数，然后从分数值中得出标签。如果分数大于或等于 0.5，则标签为 `1`。
- 对于回归问题，仅返回分数，并且它是预测的值。例如，如果使用因子分解机预测电影评级，则分数是预测的评级值。

有关训练和推理文件格式的更多详细信息，请参阅因子分解机示例笔记本 (p. 1611)。

因子分解机算法的 EC2 实例建议

Amazon SageMaker 因子分解机算法高度可扩展，并可跨分布式实例进行训练。建议对稀疏和密集数据集使用包含 CPU 实例的训练和推理。在某些情况下，对密集数据使用包含一个或多个 GPU 的训练可能具有一些优势。包含 GPU 的训练仅适用于密集数据。对稀疏数据使用 CPU 实例。

因子分解机示例笔记本

有关使用 SageMaker 因子分解机算法来分析 MNIST 数据集中从 0 到 9 的手写数字图像的示例笔记本，请参阅使用 MNIST 的分解机器简介. 有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147). 在您创建笔记本实例并将其打开后，

请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。使用分解机算法的示例笔记本电脑位于Amazon 算法简介部分。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

因子分解机的工作原理

分解机模型的预测任务是从功能集中估计函数 x_i转到目标域。该域对于回归是实际值，对于分类是二元值。因子分解机模型是指导式的，因此它具有一个训练数据集 $(x)_iy_j)$ 可用。该模型呈现的优势是它使用因子分解的参数化方法来捕获两两特征交互。它可以用数学表示，如下所示：

$$\hat{y} = w_0 + \sum_i w_i x_i + \sum_i \sum_{j>i} < v_i, v_j > x_i x_j$$

该等式中的三个项分别对应于模型的三个分量：

- w_0术语表示全局偏置。
- w_i线性项对 i 的强度进行建模th变量。
- $<v_iv_j>$ 因子分解项对 i 之间的两两交互进行建模th和 jth变量。

全局偏置项和线性项与线性模型中的相同。第三项将两两特征交互建模为每个特征所学习的相应因子的内积。所学因子也可视为每种特征的嵌入向量。例如，在分类任务中，如果一对特征往往更频繁地在积极标记样本中共同发生，则其因子的内积将较大。也就是说，其嵌入向量在余弦相似性方面彼此接近。有关因子分解机模型的更多信息，请参阅因子分解机.

对于回归任务，通过尽可能减小模型预测之间的平方误差来训练模型。$_n$和目标值 y_n. 这称为平方损失：

$$L = \frac{1}{N} \sum_n (y_n - \hat{y}_n)^2$$

对于分类任务，通过尽可能减少交叉熵损失（也称作日志丢失）来训练模型：

$$L = \frac{1}{N} \sum_n [y_n \log \hat{p}_n + (1 - y_n) \log (1 - \hat{p}_n)]$$

其中：

$$\hat{p}_n = \frac{1}{1 + e^{-\hat{y}_n}}$$

有关分类的损失函数的更多信息，请参阅分类的损失函数。

因子分解机超参数

下表包含因子分解机算法的超参数。这些是由用户设置的参数，以便于从数据中评估模型参数。首先，按字母顺序列出必须设置的所需超参数。接下来，也按字母顺序列出可以设置的可选超参数。

参数名称	描述
`feature_dim`	输入特征空间的维度。对于稀疏输入，这可能非常高。 必填 有效值：正整数。建议的值范围：[10000,10000000]
`num_factors`	因子分解的维度。 必填

参数名称	描述
	有效值：正整数。建议值范围：[2,1000]，64 通常会产生良好的结果，并是一个很好的起点。
predictor_type	预测器的类型。 • binary_classifier：对于二元分类任务。 • regressor：对于回归任务。 必填 有效值：字符串:binary_classifier要么regressor
bias_init_method	偏置项的初始化方法： • normal：使用从正态分布采样的随机值对权重进行初始化，平均值为零且标准差由 bias_init_sigma 指定。 • uniform：使用从 [-bias_init_scale, +bias_init_scale] 指定的范围均匀采样的随机值对权重进行初始化。 • constant：将权重初始化为 bias_init_value 指定的标量值。 可选 有效值：uniform、normal 或 constant 默认值：normal
bias_init_scale	偏置项的初始化范围。在 bias_init_method 设置为 uniform 时生效。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：无
bias_init_sigma	偏置项的初始化标准差。在 bias_init_method 设置为 normal 时生效。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：0.01
bias_init_value	偏置项的初始值。在 bias_init_method 设置为 constant 时生效。 可选 有效值：浮点。建议的值范围：[1e-8, 512]。 默认值：无

参数名称	描述
bias_lr	偏置项的学习率。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：0.1
bias_wd	偏置项的权重衰减。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：0.01
clip_gradient	梯度裁剪优化程序参数。通过投射到间隔 [-clip_gradient, +clip_gradient] 来剪辑梯度。 可选 有效值：Float 默认值：无
epochs	要运行的训练纪元数。 可选 有效值：正整数 默认值：1
eps	Epsilon 参数，以避免被 0 除。 可选 有效值：浮点。建议的值：小。 默认值：无
factors_init_method	因子分解项的初始化方法： • normal 使用从正态分布采样的随机值对权重进行初始化，平均值为零且标准差由 factors_init_sigma 指定。 • uniform：使用从 [-factors_init_scale, +factors_init_scale] 指定的范围均匀采样的随机值对权重进行初始化。 • constant：将权重初始化为 factors_init_value 指定的标量值。 可选 有效值：uniform、normal 或 constant。 默认值：normal

参数名称	描述
factors_init_scale	因子分解项的初始化范围。在 factors_init_method 设置为 uniform 时生效。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：无
factors_init_sigma	因子分解项的初始化标准差。在 factors_init_method 设置为 normal 时生效。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：0.001
factors_init_value	因子分解项的初始值。在 factors_init_method 设置为 constant 时生效。 可选 有效值：浮点。建议的值范围：[1e-8, 512]。 默认值：无
factors_lr	因子分解项的学习率。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：0.0001
factors_wd	因子分解项的权重衰减。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：0.00001
linear_lr	线性项的学习率。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：0.001

参数名称	描述
linear_init_method	线性项的初始化方法： • normal 使用从正态分布采样的随机值对权重进行初始化，平均值为零且标准差由 linear_init_sigma 指定。 • uniform 使用从 [-linear_init_scale, +linear_init_scale] 指定的范围均匀采样的随机值对权重进行初始化。 • constant 将权重初始化为 linear_init_value 指定的标量值。 可选 有效值：uniform、normal 或 constant。 默认值：normal
linear_init_scale	线性项的初始化范围。在 linear_init_method 设置为 uniform 时生效。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：无
linear_init_sigma	线性项的初始化标准差。在 linear_init_method 设置为 normal 时生效。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：0.01
linear_init_value	线性项的初始值。在 linear_init_method 设置为 constant 时生效。 可选 有效值：浮点。建议的值范围：[1e-8, 512]。 默认值：无
linear_wd	线性项的权重衰减。 可选 有效值：非负浮点值。建议的值范围：[1e-8, 512]。 默认值：0.001
mini_batch_size	用于训练的小批次大小。 可选 有效值：正整数 默认值：1000

参数名称	描述
rescale_grad	梯度重新扩展优化程序参数。如果设置，则在更新之前将梯度与 rescale_grad 相乘。通常选择为 1.0/batch_size。 可选 有效值：Float 默认值：无

调整因子分解机模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

由因子分解机算法计算的指标

因子分解机算法具有二元分类和回归预测器类型。预测器类型确定可用于自动模型优化的指标。该算法报告在训练期间计算的 test:rmse 回归量指标。在为回归任务优化模型时，请选择此指标作为目标。

指标名称	描述	优化方向
test:rmse	均方根误差	最小化

因子分解机算法报告在训练期间计算的 3 个二元分类指标。在为二元分类任务优化模型时，请选择其中之一作为目标。

指标名称	描述	优化方向
test:binary_classification_accuracy	Accuracy	最大化
test:binary_classification_cross_entropy	交叉熵	最小化
test:binary_f_beta	测试版	最大化

可优化因子分解机超参数

您可以优化因子分解机算法的以下超参数。包含偏置项、线性项和因子分解项的初始化参数取决于其初始化方法。有三种初始化方法：uniform、normal 和 constant。这些初始化方法本身是不可优化的。可优化参数依赖于初始化方法的选择。例如，如果初始化方法为 uniform，则仅 scale 参数是可优化的。具体而言，如果 bias_init_method==uniform，则 bias_init_scale、linear_init_scale 和 factors_init_scale 是可优化的。同样，如果初始化方法为 normal，则仅 sigma 参数是可优化的。如果初始化方法为 constant，则仅 value 参数是可优化的。下表列出了这些依赖项。

参数名称	参数类型	建议的范围	依赖关系
bias_init_scale	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	bias_init_method==uniform

参数名称	参数类型	建议的范围	依赖关系
bias_init_sigma	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	bias_init_method==normal
bias_init_value	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	bias_init_method==constant
bias_lr	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	无
bias_wd	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	无
epoch	IntegerParameterRange	MinValue：1，MaxValue：1000	无
factors_init_scale	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	bias_init_method==uniform
factors_init_sigma	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	bias_init_method==normal
factors_init_value	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	bias_init_method==constant
factors_lr	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	无
factors_wd	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512]	无
linear_init_scale	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	bias_init_method==uniform
linear_init_sigma	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	bias_init_method==normal
linear_init_value	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	bias_init_method==constant
linear_lr	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	无
linear_wd	ContinuousParameterRange	最 MinValue：1e-8，MaxValue：512	无

参数名称	参数类型	建议的范围	依赖关系
mini_batch_size	IntegerParameterRange	MinValue： MaxValue：100 10000	无

因子分解机响应格式

JSON 响应格式

二进制分类

```
let response =   {
    "predictions":    [
        {
            "score": 0.4,
            "predicted_label": 0
        }
    ]
}
```

回归

```
let response =   {
    "predictions":    [
        {
            "score": 0.4
        }
    ]
}
```

JSONLINES 响应格式

二进制分类

```
{"score": 0.4, "predicted_label": 0}
```

回归

```
{"score": 0.4}
```

RECORDIO 响应格式

二进制分类

```
[
    Record = {
        features = {},
        label = {
            'score': {
                keys: [],
                values: [0.4]  # float32
            },
            'predicted_label': {
                keys: [],
                values: [0.0]  # float32
            }
        }
    }
```

```
]
```

回归

```
[
    Record = {
        features = {},
        label = {
            'score': {
                keys: [],
                values: [0.4]  # float32
            }
        }
    }
]
```

图像分类算法

Amazon SageMaker 图像分类算法是一种有监督的学习算法，它支持多标签分类。该算法将一个图像作为输入，并输出分配给该图像的一个或多个标签。它使用可从头训练或在大量训练图像不可用时使用迁移学习训练的卷积神经网络。

对于 Amazon，建议的输入格式 SageMaker 映像分类算法是 Apache MXNetRecordIO. 但是，您也可以使用 .jpg 或 .png 格式的原始图像。有关机器学习系统的高效数据准备和加载的广泛概述，请参阅本讨论。

> Note
>
> 为了与现有深度学习框架保持更好的互操作性，这不同于其他 Amazon SageMaker 算法常用的 protobuf 数据格式。

有关卷积网络的更多信息，请参阅：

- 《图像识别中的深度残差学习》，何凯明等人编著，计算机视觉和模式识别 2016 年 IEEE 会议
- Imagenet 图像数据库
- 使用 Gluon-CV 和 MxNet 进行图像分类

主题

- 图像分类算法的输入/输出接口 (p. 1620)
- 图像分类算法的 EC2 实例建议 (p. 1622)
- 图像分类示例笔记本 (p. 1622)
- 图像分类的工作原理 (p. 1623)
- 图像分类超参数 (p. 1623)
- 优化图像分类模型 (p. 1629)

图像分类算法的输入/输出接口

这些区域有： SageMaker 图像分类算法同时支持 RecordIO (`application/x-recordio`) 和图片 (`image/png`、`image/jpeg`, 和`application/x-image`) 在文件模式下进行训练的内容类型，并支持 RecordIO (`application/x-recordio`) 在管道模式下训练的内容类型。但是，您还可以使用图像文件（`image/png`、`image/jpeg` 和 `application/x-image`）在管道模式下进行训练，无需使用扩增清单格式创建 RecordIO 文件。

对于文件模式和管道模式支持分布式训练。在管道模式下使用 RecordIO 内容类型时，必须将 `S3DataSource` 的 `S3DataDistributionType` 设置为 `FullyReplicated`。该算法支持完全复制的模型，将数据复制到每台计算机上。

该算法支持使用 `image/png`、`image/jpeg` 和 `application/x-image` 进行推理。

使用 RecordIO 格式进行训练

如果您使用 RecordIO 格式进行训练，则同时指定 `train` 和 `validation` 通道作为 `InputDataConfig` 请求的 `CreateTrainingJob` 参数值。在 `train` 通道中指定一个 RecordIO (`.rec`) 文件，在 `validation` 通道中指定一个 RecordIO 文件。将两个通道的内容类型设置为 `application/x-recordio`。

使用图像格式进行训练

如果您使用图像格式进行训练，则指定 `train`、`validation`、`train_lst` 和 `validation_lst` 通道作为 `InputDataConfig` 请求的 `CreateTrainingJob` 参数值。为 `train` 或 `validation` 通道指定相应的图像数据（`.jpg` 和 `.png` 文件）。在 `.lst` 和 `train_lst` 通道各指定一个 `validation_lst` 文件。将所有四个通道的内容类型均设置为 `application/x-image`。

> **Note**
>
> SageMaker 从不同的渠道分别读取训练和验证数据，因此您必须将训练和验证数据存储在不同的文件夹中。

`.lst` 文件是一个包含图像文件列表的三列制表符分隔文件。第一列指定图像索引，第二列指定图像的分类标签索引，第三列指定图像文件的相对路径。第一列中所有图像的图像索引必须唯一。分类标签索引集采用连续编号，并且编号应从 0 开始。例如，cat 分类为 0，dog 分类为 1，其他分类以此分类推。

以下是 `.lst` 文件的示例：

```
5       1   your_image_directory/train_img_dog1.jpg
1000    0   your_image_directory/train_img_cat1.jpg
22      1   your_image_directory/train_img_dog2.jpg
```

例如，如果您的训练图像存储在 `s3://<your_bucket>/train/class_dog`、`s3://<your_bucket>/train/class_cat` 等位置，请将 `train` 通道的路径指定为 `s3://<your_bucket>/train`，这是数据的顶级目录。在 `.lst` 文件中，将 `train_image_dog1.jpg` 分类目录中名为 `class_dog` 的单个文件的相对路径指定为 `class_dog/train_image_dog1.jpg`。您也可以将所有图像文件存储在 `train` 目录内的一个子目录下。在这种情况下，请将该子目录用于相对路径。例如，`s3://<your_bucket>/train/your_image_directory`。

使用扩增清单图像格式进行训练

扩增清单格式使您可以使用图像文件在管道模式下进行训练，而无需创建 RecordIO 文件。您需要将训练通道和验证通道指定为 `CreateTrainingJob` 请求的 `InputDataConfig` 参数的值。使用该格式时，需要生成包含图像列表及其相应注释的 S3 清单文件。清单文件格式应为 JSON 行格式，其中每行代表一个样本。使用指向图像 S3 位置的 `'source-ref'` 标签来指定图像。将在 `"AttributeNames"` 参数值下面提供注释，如 `CreateTrainingJob` 请求中指定。它还可以在 `metadata` 标签下包含其他元数据，但这些会被算法忽略。在以下示例中，`"AttributeNames"`包含在图像和注释引用 `["source-ref", "class"]` 列表中。第一个图像的相应标签值为 `"0"`，第二个图像的标签值为 `“1”`：

```
{"source-ref":"s3://image/filename1.jpg", "class":"0"}
{"source-ref":"s3://image/filename2.jpg", "class":"1", "class-metadata": {"class-name":
 "cat", "type" : "groundtruth/image-classification"}}
```

按顺序分别为`"AttributeNames"`在输入文件中训练时很重要 ImageClassification 算法。它按特定的顺序接受管道数据，先接受 `image`，然后接受 `label`。因此，本示例中的“AttributeNames”先使用 `"source-ref"` 提供，然后使用 `"class"` 提供。使用 ImageClassification 具有增强清单的算法，`RecordWrapperType`参数必须为`"RecordIO"`.

也可以通过指定值的 JSON 数组来支持多标签训练。必须将 `num_classes` 超参数设置为与类的总数匹配。有两种有效的标签格式：multi-hot 和 class-id。

在 multi-hot 格式中，每个标签都是所有类的 multi-hot 编码向量，其中每个类的值为 0 或 1。在以下示例中，有三种类别。第一个图像标记为类 0 和 2，而第二个图像仅标记为类 2：

```
{"image-ref": "s3://mybucket/sample01/image1.jpg", "class": "[1, 0, 1]"}
```

```
{"image-ref": "s3://mybucket/sample02/image2.jpg", "class": "[0, 0, 1]"}
```

在 class-id 格式中，每个标签都是适用于数据点的类 ID 的列表，从 [0, num_classes) 开始。前面的例子看起来像是这样：

```
{"image-ref": "s3://mybucket/sample01/image1.jpg", "class": "[0, 2]"}
{"image-ref": "s3://mybucket/sample02/image2.jpg", "class": "[2]"}
```

Multi-hot 格式是默认的，但可在内容类型中使用label-format参数："application/x-recordio; label-format=multi-hot".必须显式设置 class-id 格式，即 GroundTruth 输出的格式："application/x-recordio; label-format=class-id".

有关增强清单文件的更多信息，请参阅通过增强清单文件，将数据集元数据提供给训练作业 (p. 2090)。

增量训练

您还可以使用以前使用 SageMaker 训练的模型中的构件为新模型的训练做种。当您想要训练具有相同或相似数据的新模型时，增量训练可以节省训练时间。 SageMaker 只能使用在 SageMaker 训练的另一个内置图像分类模型来为图像分类模型做种。

要使用预训练模型，请在 CreateTrainingJob 请求中，在 InputDataConfig 参数中将 ChannelName 指定为“model”。将模型通道的 ContentType 设置为 application/x-sagemaker-model。您上传到模型通道的新模型和预训练模型的输入超参数必须与 num_layers、image_shape 和 num_classes 输入参数具有相同的设置。这些参数定义了网络架构。对于预训练模型文件，请使用 SageMaker 输出的压缩模型构件（采用 .tar.gz 格式）。您可以对输入数据使用 RecordIO 或图像格式。

有关演示如何将增量训练与 SageMaker 图像分类算法一起使用的示例笔记本，请参阅端到端增量训练图像分类示例. 有关增量训练及其使用方式说明的更多信息，请参阅Amazon SageMaker 中的增量训练 (p. 2081)。

使用图像分类算法进行推理

生成的模型可以托管用于推理，并且支持编码的 .jpg 和 .png 图像格式作为 image/png, image/jpeg 和 application/x-image 内容类型。输入图像将自动调整大小。输出是以 JSON 格式编码的所有分类的概率值，或者对批量转换采用 JSON 行文本格式。图像分类模型按请求处理单个图像，因此只输出采用 JSON 的一行或 JSON 行格式。以下是 JSON 行格式的响应示例：

```
accept: application/jsonlines

 {"prediction": [prob_0, prob_1, prob_2, prob_3, ...]}
```

有关训练和推理的更多详细信息，请参阅简介中引用的图像分类示例笔记本实例。

图像分类算法的 EC2 实例建议

对于图像分类，我们支持以下 GPU 实例用于训练：ml.p2.xlarge、ml.p2.8xlarge、ml.p2.16xlarge、ml.p3.2xlarge、ml.p3.8xlarge 和 ml.p3.16xlarge。对于大批量训练，建议使用具有更多内存的 GPU 实例。但是，CPU (如 C4) 和 GPU (如 P2 和 P3) 实例均可用于推理。您也可以在多 GPU 和多机器设置上运行该算法以进行分布式训练。

P2 和 P3 实例均受图像分类算法支持。

图像分类示例笔记本

对于使用 SageMaker 图像分类算法在 caltech-256 数据集上训练模型，然后部署模型以执行推理，请参阅端到端多类别图像分类示例. 有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 在您创建笔记本实例并打开之后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例。示例图像分类笔记本位于 Amazon 算法简介部分。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

图像分类的工作原理

图像分类算法将图像作为输入，并将其归入输出类别之一。深度学习彻底改变了图像分类域，并且表现非常出色。各种深度学习网络 (如 ResNet [1]、DenseNet、Inception 等) 的开发均旨在实现高度准确的图像分类。与此同时，我们一直在努力收集对训练这些网络非常必要的标签图像数据。ImageNet[2] 便是这样一种大型数据集，它拥有大约 11,000 种类别和超过 1,100 万张图像。一旦网络接受了培训 ImageNet 然后，该数据集可用于通过简单的重新调整或微调进行泛化。在这种迁移学习方法中，网络通过权重进行初始化（对于本例，在 ImageNet 上进行训练），稍后可针对不同数据集中的图像分类任务进行优化。

Amazon 中的图像分类 SageMaker 可在两种模式下运行：完整训练和迁移学习。在完整训练模式下，网络使用随机权重进行初始化，并从头开始根据用户数据进行训练。在迁移学习模式下，网络使用预先训练的权重进行初始化，并且只有顶部完全连接的层使用随机权重进行初始化。然后，使用新数据对整个网络进行微调。在此模式下，甚至可以使用较小数据集实现训练。这是因为网络已经过训练，因此可在训练数据不足的情况下使用。

图像分类超参数

超参数是在机器学习模型开始学习之前设置的参数。Amazon 支持以下超参数： SageMaker 内置图像分类算法。有关图像分类超参数优化的信息，请参阅优化图像分类模型 (p. 1629)。

参数名称	描述
num_classes	输出分类的数量。此参数定义网络输出的维度，通常设置为数据集中的类别数。 除了多类别分类外，还支持多标签分类。有关如何使用增强清单文件处理多标签分类的详情，请参阅图像分类算法的输入/输出接口 (p. 1620)。 必填 有效值：正整数
num_training_samples	输入数据集中的训练示例数。 如果此值与训练集中的示例数不匹配，则 lr_scheduler_step 参数的行为未定义，且分布式训练准确率可能会受影响。 必填 有效值：正整数
augmentation_type	数据扩增类型。输入图像可按照下面指定的多种方式扩增。 • crop：随机裁剪图像并水平翻转图像 • crop_color：除了“裁剪”之外，还向相应的色调、饱和度和亮度通道分别添加 [-36, 36]、[-50, 50] 和 [-50, 50] 范围内的三个随机值 • crop_color_transform：除了 crop_color 之外，还对图像应用随机转换，包括旋转、剪切和宽高比变化。最大旋转角度为 10 度，最大剪切比为 0.1，而最大宽高变化比为 0.25。 可选 有效值：crop、crop_color 或 crop_color_transform。 默认值：无默认值
beta_1	adam 的 beta1，也就是一阶矩估计的指数衰减速率。

参数名称	描述
	可选 有效值：浮点值。范围为 [0, 1]。 默认值：0.9
beta_2	adam 的 beta2，也就是二阶矩估计的指数衰减速率。 可选 有效值：浮点值。范围为 [0, 1]。 默认值：0.999
checkpoint_frequency	存储模型参数的周期 (纪元数)。 请注意，所有检查点文件都会作为最终模型文件“model.tar.gz”的一部分保存，并上传到 S3 中的指定模型位置。这将根据训练期间保存的检查点数按比例增加模型文件的大小。 可选 有效值：不大于 epochs 的正整数。 默认值：无默认值（在验证准确率最高的纪元保存检查点。）
early_stopping	True 表示在训练期间使用提前停止逻辑。False 表示不使用它。 可选 有效值：True 或 False 默认值：False
early_stopping_min_epochs	可以调用提前停止逻辑之前必须运行的最小纪元数。仅当 early_stopping = True 时使用它。 可选 有效值：正整数 默认值：10
early_stopping_patience	如果在相关指标中没有改进，则在结束训练前等待的纪元数。仅当 early_stopping = True 时使用它。 可选 有效值：正整数 默认值：5

参数名称	描述
early_stopping_tolerance	用于评估准确率验证指标改善的相对容差。如果准确率改善除以上一个最佳准确率的比率小于设置的 early_stopping_tolerance 值，则提前停止逻辑会认为没有改善。仅当 early_stopping = True 时使用它。 可选 有效值：0 ≤ 浮点型 ≤ 1 默认值：0.0
epochs	训练纪元数。 可选 有效值：正整数 默认值：30
eps	adam 和 rmsprop 的 epsilon 值。它通常设置为较小的值，以避免被 0 除。 可选 有效值：浮点值。范围为 [0, 1]。 默认值：1e-8
gamma	rmsprop 的 gamma 值，平方梯度的移动平均值的衰减系数。 可选 有效值：浮点值。范围为 [0, 1]。 默认值：0.9
image_shape	输入图像尺寸，该尺寸与网络输入层的大小相同。其格式定义为“num_channels, 高度, 宽度”。图像尺寸可具有任意值，因为网络可处理各种尺寸的输入。但是，如果使用较大图像尺寸，可能会有内存限制。预训练的模型只能使用固定的 224 x 224 图像尺寸。图像分类的典型图像尺寸为“3, 224, 224”。这类似于 ImageNet 数据集。 对于训练，如果任何维度中的任何输入图像小于此参数，则训练将失败。如果图像较大，则会裁剪图像的一部分，并由此参数指定裁剪区域。如果超参数augmentation_type被设置，随机作物；否则，将采取中央作物。 在推断时，输入图像的大小调整为image_shape这是在训练过程中使用的。不保留纵横比，图像也不会被裁剪。 可选 有效值：字符串 默认值：'3, 224, 224 '

参数名称	描述
kv_store	分布式训练期间的权重更新同步模式。可对各机器同步或异步更新权重更新。同步更新通常比异步更新提供更好的准确率，但速度较慢。请参阅 MXNet 中的分布式训练，以了解更多详细信息。 此参数不适用于单个机器训练。 • dist_sync：每次批处理后，梯度会与所有工作线程同步。对于 dist_sync，批次大小现在表示每台机器上使用的批次大小。因此，如果有 n 台机器，并且我们使用批次大小 b，则 dist_sync 行为类似于本地行为且批次大小为 n*b • dist_async：执行异步更新。每当从任何机器收到梯度时，权重就会更新，并且权重更新是原子更新。然而，该顺序并不能得到保证。 可选 有效值：dist_sync 或 dist_async 默认值：无默认值
learning_rate	初始学习率。 可选 有效值：浮点值。范围为 [0, 1]。 默认值：0.1
lr_scheduler_factor	与 lr_scheduler_step 参数结合使用以降低学习率的比率，定义为 lr_new = lr_old * lr_scheduler_factor。 可选 有效值：浮点值。范围为 [0, 1]。 默认值：0.1
lr_scheduler_step	降低学习率的纪元。正如 lr_scheduler_factor 参数中所述，学习率在这些纪元降低了 lr_scheduler_factor。例如，如果将值设置为“10, 20”，则学习率在第 10 纪元之后降低 lr_scheduler_factor，然后在第 20 纪元之后再次降低 lr_scheduler_factor。纪元之间用“,”分隔。 可选 有效值：字符串 默认值：无默认值

参数名称	描述
mini_batch_size	训练的批次大小。在单机器多 GPU 设置中，每个 GPU 处理 mini_batch_size/num_gpu 个训练样本。对于 dist_sync 模式下的多机器训练，实际批次大小为 mini_batch_size*机器数量。请参阅 MXNet 文档了解更多详细信息。 可选 有效值：正整数 默认值：32
momentum	用于 sgd 和 nag 的动量，其他优化程序忽略此值。 可选 有效值：浮点值。范围为 [0, 1]。 默认值：0.9
multi_label	用于多标签分类的标志，其中每个示例可以分配多个标签。记录所有分类的平均准确率。 可选 有效值：0 或 1 默认值：0
num_layers	网络的层数。对于包含大尺寸图像的数据 (例如 224x224 - 像 ImageNet)，建议从数据集 [18, 34, 50, 101, 152, 200] 中选择层数。对于包含小尺寸图像的数据 (例如 28x28 - 像 CIFAR)，建议从数据集 [20, 32, 44, 56, 110] 中选择层数。每个集合中的层数均基于 ResNet 文章)。对于迁移学习，层数定义了基础网络的架构，因此只能从数据集 [18, 34, 50, 101, 152, 200] 中选择。 可选 有效值：[18, 34, 50, 101, 152, 200] 或 [20, 32, 44, 56, 110] 中的正整数 默认值：152
optimizer	优化程序类型。有关优化程序参数的更多详细信息，请参阅 MXNet 的 API。 可选 有效值：以下值之一：sgd、adam、rmsprop 或 nag。 • sgd：随机梯度下降 • adam：适应性动量估计 • rmsprop： 均方根传播 • nag：Nesterov 加速梯度 默认值：sgd

参数名称	描述
precision_dtype	用于训练的权重精度。该算法可以对权重使用单精度 (float32) 或半精度 (float16)。对权重使用半精度会使内存消耗量减少。 可选 有效值：float32 或 float16 默认值：float32
resize	调整图像大小以进行训练后，图像最短边的像素数。如果未设置此参数，则使用训练数据时不调整大小。参数应该大于的宽度和高度分量image_shape以防止训练失败。 使用图像内容类型时为必需 使用 RecordIO 内容类型时为可选 有效值：正整数 默认值：无默认值
top_k	训练期间报告前 k 个最大数准确率。此参数必须大于 1，因为 top-1 训练准确率与已报告的常规训练准确率相同。 可选 有效值：大于 1 的正整数。 默认值：无默认值
use_pretrained_model	指示使用预先训练的模型进行训练的标志。如果设置为 1，则加载具有相应层数的预先训练的模型并将其用于训练。只有顶部 FC 层使用随机权重进行重新初始化。否则，从头训练网络。 可选 有效值：0 或 1 默认值：0
use_weighted_loss	用于多标签分类的加权交叉熵损失的标志（仅在 multi_label = 1 时使用），其中权重是基于分类的分布计算的。 可选 有效值：0 或 1 默认值：0
weight_decay	用于 sgd 和 nag 的权重衰减系数，其他优化程序忽略此值。 可选 有效值：浮点值。范围为 [0, 1]。 默认值：0.0001

优化图像分类模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

由图像分类算法计算的指标

图像分类算法是一种指导式算法。它报告在训练期间计算的准确率指标。优化模型时，请选择此指标作为目标指标。

指标名称	描述	优化方向
validation:accuracy	正确预测数量与预测总数之比。	最大化

可优化图像分类超参数

使用以下超参数优化图像分类模型。对图像分类目标指标影响最大的超参数包括：mini_batch_size、learning_rate 和 optimizer。根据选定 optimizer 优化与优化程序相关的超参数，例如 momentum、weight_decay、beta_1、beta_2、eps 和 gamma。例如，仅当 adam 是 optimizer 时，使用 beta_1 和 beta_2。

有关每个优化程序中使用哪些超参数的更多信息，请参阅图像分类超参数 (p. 1623)。

参数名称	参数类型	建议的范围
beta_1	ContinuousParameterRanges	最 MinValue：1e-6，MaxValue：0.999
beta_2	ContinuousParameterRanges	最 MinValue：1e-6，MaxValue：0.999
eps	ContinuousParameterRanges	最 MinValue：1e-8，MaxValue：1.0
gamma	ContinuousParameterRanges	最 MinValue：1e-8，MaxValue：0.999
learning_rate	ContinuousParameterRanges	最 MinValue：1e-6，MaxValue：0.5
mini_batch_size	IntegerParameterRanges	MinValue：8，MaxValue：512
momentum	ContinuousParameterRanges	MinValue：0.0，MaxValue：0.999
optimizer	CategoricalParameterRanges	['sgd'、‘adam’、‘rmsprop’、'nag']
weight_decay	ContinuousParameterRanges	MinValue：0.0，MaxValue：0.999

IP 见解

Amazon SageMaker IP 见解是一种自主学习算法，可以学习 IPv4 地址的使用模式。它旨在捕获 IPv4 地址与各种实体（例如用户 ID 或帐号）之间的关联。例如，您可以使用它来识别试图从异常 IP 地址登录 Web 服务的用户。或者，您可以使用它来识别尝试从异常 IP 地址创建计算资源的账户。经过训练的 IP 见解模型可以托管在终端节点上以进行实时预测或用于处理批量转换。

SageMaker IP 见解以（实体、IPv4 地址）对的形式提取历史数据，并了解每个实体的 IP 使用模式。当使用（实体、IPv4 地址）事件进行查询时，SageMaker IP 见解模型返回一个分数，用于推断事件模式的异常程度。例如，当用户尝试从 IP 地址登录时，如果 IP Insights 分数足够高，Web 登录服务器可能会决定触发多因素身份验证系统。在更高级的解决方案中，您可以将 IP Insights 分数提供到另一种机器学习模型中。例如，您可以将 IP Insight 分数与其他功能结合起来，以对另一个安全系统的调查结果进行排名，例如来自Amazon GuardDuty.

SageMaker IP 见解算法还可以学习 IP 地址的向量表示，称为嵌入. 您可以在下游机器学习任务中使用矢量编码嵌入作为使用 IP 地址中观察到的信息的功能。例如，您可以在诸如测量群集和可视化任务中 IP 地址之间的相似性等任务中使用它们。

主题

- IP 见解算法的输入/输出接口 (p. 1630)
- IP 见解算法的 EC2 实例建议 (p. 1630)
- IP 见解示例笔记本 (p. 1631)
- IP 见解如何运作 (p. 1631)
- IP 见解超参数 (p. 1632)
- 调整 IP 见解模型 (p. 1634)
- IP 见解数据格式 (p. 1635)

IP 见解算法的输入/输出接口

训练和验证

SageMaker IP 见解算法支持训练和验证数据通道。它使用可选的验证渠道来计算预定义的负面采样策略的曲线下面积 (AUC) 得分。AUC 指标验证模型在正样本和负面样本之间的区分程度。需要使用培训和验证数据内容类型`text/csv`格式的日期和时间。CSV 数据的第一列是一个不透明的字符串，它为实体提供了唯一标识符。第二列是十进制点表示法的 IPv4 地址。IP 见解目前仅支持文件模式。有关详细信息以及示例，请参阅IP 见解训练数据格式 (p. 1635).

推理

为了推断，IP 见解支持`text/csv`、`application/json`, 和`application/jsonlines`数据内容类型。有关 SageMaker 提供的推理的常见数据格式的详细信息，请参阅用于推理的常见数据格式 (p. 1583). IP Insights 推理返回格式为之一的输出`application/json`要么`application/jsonlines`. 输出数据中的每条记录都包含相应的`dot_product`每个输入数据点的兼容性分数（或兼容性分数）。有关详细信息以及示例，请参阅IP 见解推理数据格式 (p. 1636).

IP 见解算法的 EC2 实例建议

SageMaker IP 见解算法可以在 GPU 和 CPU 实例上运行。对于培训作业，我们建议使用 GPU 实例。但是，对于具有大型训练数据集的某些工作负载，分布式 CPU 实例可能会降低培训 为了推断，我们建议使用 CPU 实例。

IP 见解算法的 GPU 实例

IP 见解支持所有可用的 GPU。如果您需要加快训练速度，我们建议从单个 GPU 实例（例如 ml.p3.2xlarge）开始，然后迁移到多 GPU 环境，例如 ml.p3.8xlarge 和 ml.p3.16xlarge。多 GPU 会自动划分迷你批次的训练数据。如果从单个 GPU 切换到多个 GPU，`mini_batch_size`平均分为使用的 GPU 数量。您可能希望增加的值`mini_batch_size`来弥补这一点。

IP 见解算法的 CPU 实例

我们推荐的 CPU 实例类型在很大程度上取决于实例的可用内存和型号大小。模型大小由两个超参数决定：vector_dim和num_entity_vectors. 支持的最大型号大小为 8 GB。下表针对各种模型大小，列出了基于这些输入参数部署的典型 EC2 实例类型。在表 1 中，vector_dim在第一列范围从 32 到 2048 年，以及num_entity_vectors在第一排中，从 1 万到 50,000,000 个不等。

vector_d \ num_enti	10000	50000	100000	500,000	1000000	500 万	10,000,000	50,000,000
32	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.xlarge	ml.m5.2xlarge	ml.m5.4xlarge
64	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.2xlarge	ml.m5.2xlarge	
128	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.2xlarge	ml.m5.4xlarge	
256	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.xlarge	ml.m5.4xlarge		
512	ml.m5.large	ml.m5.large	ml.m5.large	ml.m5.2xlarge				
1024	ml.m5.large	ml.m5.large	ml.m5.xlarge	ml.m5.4xlarge				
2048	ml.m5.large	ml.m5.xlarge	ml.m5.xlarge					

字段mini_batch_size、num_ip_encoder_layers、random_negative_sampling_rate, 和shuffled_negative_sampling_rate超参数还会影响所需的内存量。如果这些值很大，您可能需要使用比正常更大的实例类型。

IP 见解示例笔记本

有关显示如何训练 SageMaker IP 见解算法并使用它执行推断的示例笔记本，请参阅SageMakerIP 见解算法简介. 有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147). 创建笔记本实例后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

IP 见解如何运作

Amazon SageMaker IP 见解是一种自主算法，它以（实体、IPv4 地址）对的形式使用观察到的数据，这些配对将实体与 IP 地址相关联。IP Insights 通过学习实体和 IP 地址的潜在矢量表示法来确定实体使用特定 IP 地址的可能性。然后，这两种表示之间的距离可以作为代理这种关联的可能性。

IP Insights 算法使用神经网络来学习实体和 IP 地址的潜在矢量表示形式。实体首先被哈希处理为大但固定的哈希空间，然后由简单的嵌入层进行编码。当用户名或帐户 ID 等字符串出现在日志文件中时，可以直接输入 IP Insights。您不需要预处理实体标识符的数据。在训练和推理期间，您可以将实体作为任意字符串值提供。哈希大小应配置一个足够高的值，以确保碰撞，当不同的实体映射到同一潜伏向量时，发生这种情况仍然微不足道。有关如何选择适当的哈希大小的详细信息，请参阅适用于大规模多任务学习的功能哈希. 另一方面，为了表示 IP 地址，IP Insights 使用专门设计的编码器网络，通过利用 IP 地址的前缀结构来唯一表示每个可能的 IPv4 地址。

在训练期间，IP Insights 通过随机配对实体和 IP 地址自动生成负面样本。这些负样本表示实际不太可能出现的数据。该模型经过训练以区分训练数据中观察到的阳性样本和这些生成的阴性样本。更具体地说，模型经过训练以尽量减少交叉熵，也称为记录丢失，定义如下：

$$L = \frac{1}{N}\sum_{n}\left[y_n \log p_n + (1 - y_n)\log\left(1 - p_n\right)\right]$$

y_n是指示样本是否来自管理观测数据的实际分布（y）的标签$_n$=1) 或生成负样本的分布 (y)$_n$=0)。p_n是模型所预测的样本来自实际分布的概率。

生成负采样是一个重要的过程，用于实现观察到的数据的准确模型。如果负样本极不可能出现，例如，如果负样本中的所有 IP 地址都是 10.0.0.0，那么模型就可以轻松地学会区分负面样本，并且无法准确地描述实际观察到的数据集。为了保持负面样本更加逼真，IP Insights 通过随机生成 IP 地址和从训练数据中随机选择 IP 地址来生成负面样本。您可以配置负采样的类型和生成负样本的速率`random_negative_sampling_rate`和`shuffled_negative_sampling_rate`超参数。

给定 n 个（实体、IP 地址对），IP Insights 模型会输出分数，S_n，表示实体与 IP 地址的兼容程度。该分数与来自实际分配的货币对的给定（实体、IP 地址）的对数赔率比相对应于负分配。它的定义如下：

$$S_n = \log\left(\frac{P_{real}(n)}{P_{neg}(n)}\right)$$

该分数基本上是衡量第 n 个实体的矢量表示与 IP 地址之间的相似性的一种衡量标准。它可以解释为在现实中观察这个事件的可能性比在随机生成的数据集中更有可能性。在训练过程中，算法使用此分数来计算来自实际分布的样本概率的估计值，p_n，在交叉熵最小化中使用，其中：

$$p_n = \frac{1}{1 + e^{-S_n}}$$

IP 见解超参数

在 `CreateTransformJob` 请求中，您可以指定训练算法。您还可以指定算法特定的超参数作为字符串到字符串映射。下表列出了 Amazon SageMaker IP 见解算法的超参数。

参数名称	描述
`num_entity_vectors`	要训练的实体矢量表示（实体嵌入向量）的数量。使用哈希函数将训练集中的每个实体随机分配给其中一个向量。由于哈希冲突，可能会将多个实体分配给同一个向量。这将导致同一向量表示多个实体。只要碰撞率不太严重，这对模型性能的影响通常微不足道。要保持较低的碰撞率，请尽可能设置此值。但是，模型大小以及因此，训练和推理的内存需求都会随此超参数线性扩展。我们建议您将此值设置为唯一实体标识符数的两倍。 必填 有效值：1 ≤ 正整数 ≤ 250,000,000
`vector_dim`	用于表示实体和 IP 地址的嵌入向量的大小。该值越大，使用这些表示法编码的信息越多。实际上，模型尺寸与此参数线性缩放，并限制维度的大小。此外，使用太大的矢量表示可能会导致模型过度拟合，尤其是对于小型训练数据集而言。如果模型没有学习数据中的任何模式，但有效地记住训练数据，因此，在推理过程中不能很好地概括并表现不佳，就会发生过度拟合。建议值为 128。 必填 有效值：4 ≤ 正整数 ≤ 4096
`batch_metrics_publish_interval`	Apache MXNet Speedometer 函数输出网络训练速度（采样数/秒）的间隔（每 X 个批次）。 可选 有效值：正整数 ≥ 1 默认值：1000

参数名称	描述
epochs	训练数据的传递次数。最佳值取决于您的数据大小和学习率。典型值范围为 5 到 100。 可选 有效值：正整数 ≥ 1 默认值：10
learning_rate	优化程序的学习率。IP 见解使用基于渐变的亚当优化器。学习率可以有效地控制每次迭代时更新模型参数的步长大小。过大的学习率可能会导致模型分歧，因为训练可能超过最低限度。另一方面，学习率太小会减慢趋同速度。典型值范围从 1e-4 到 1e-1。 可选 有效值：1e-6 ≤ 浮点值 ≤ 10.0 默认值：0.001
mini_batch_size	每个小批量中的示例数。培训过程分批处理数据。最佳值取决于数据集中唯一账户标识符的数量。一般来说，越大mini_batch_size, 训练速度越快, 可能的洗牌-负样本组合的数量也越多. 但是，有一个很大的mini_batch_size, 培训更有可能收敛到不佳的当地最低水平, 而且在推断方面表现相对较差. 可选 有效值：1 ≤ 正整数 ≤ 500000 默认值：10000
num_ip_encoder_layers	用于编码 IP 地址嵌入的完全连接的层数。层数越多，模型在 IP 地址之间捕获模式的能力就越大。但是，使用大量图层会增加过度拟合的可能性。 可选 有效值：0 ≤ 正整数 ≤ 100 默认值：1
random_negative_sampling_rate	针对每个输入示例生成的随机负采样的数量 R。训练过程依赖于负采样来防止模型的向量表示折叠到单个点。随机负采样为小批量中的每个输入账户生成 R 个随机 IP 地址。的总和random_negative_sampling_rate(R) 和shuffled_negative_sampling_rate(S) 必须在时间间隔内：1 ≤ R + S ≤ 500。 可选 有效值：0 ≤ 正整数 ≤ 500 默认值：1

参数名称	描述
shuffled_negative_sampling_rate	针对每个输入示例生成的随机负采样的数量 S。在某些情况下，它有助于使用从训练数据本身随机挑选的更真实的负采样。这种负抽样是通过在一个小批次中洗牌数据来实现的。随机排列负采样通过在小批量内随机排列 IP 地址和账户对来生成 S 负 IP 地址。的总和random_negative_sampling_rate(R)和shuffled_negative_sampling_rate(S) 必须在时间间隔内：1 ≤ R + S ≤ 500。 可选 有效值：0 ≤ 正整数 ≤ 500 默认值：1
weight_decay	权重衰减系数。此参数添加了一个 L2 正则化因子，这是防止模型过度拟合训练数据所必需的。 可选 有效值：0.0 ≤ 浮点值 ≤ 10.0 默认值：0.00001

调整 IP 见解模型

自动调整模型（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

IP 见解算法计算的指标

Amazon SageMaker IP 见解算法是一种自主学习算法，可以学习 IP 地址和实体之间的关联。该算法训练一个鉴别器模型，该模型可以学习分离观察到的数据点（正样本）来自随机生成的数据点 (阴性示例)。IP Insights 上的自动模型调整可帮助您找到能够最准确地区分未标签验证数据和自动生成的负面样本的模型。验证数据集上的模型准确率由受试者操作特征曲线下的面积来衡量。该validation:discriminator_auc指标可以采用介于 0.0 和 1.0 之间的值，1.0 表示完美的准确度。

IP 见解算法计算validation:discriminator_auc验证期间的指标，其值用作优化超参数调整的目标函数。

指标名称	描述	优化方向
validation:discriminator_auc	验证数据集上受试者操作特征曲线下的面积。验证数据集没有标记。曲线下方的区域 (AUC) 是一个指标，它描述了模型区分验证数据点与随机生成的数据点的能力。	最大化

可优化 IP 见解超参数

你可以优化以下内容超参数针对 SageMaker IP 见解算法。

参数名称	参数类型	建议的范围
epochs	IntegerParameterRange	MinValue：1，MaxValue：100
learning_rate	ContinuousParameterRange	最 MinValue：1e-4，MaxValue：0.1
mini_batch_size	IntegerParameterRanges	MinValue：100，MaxValue：50000
num_entity_vectors	IntegerParameterRanges	MinValue：10000，MaxValue：1000000
num_ip_encoder_layers	IntegerParameterRanges	MinValue：1，MaxValue：10
random_negative_sampling_rate	IntegerParameterRanges	MinValue：0，MaxValue：10
shuffled_negative_sampling_rate	IntegerParameterRanges	MinValue：0，MaxValue：10
vector_dim	IntegerParameterRanges	MinValue：8，MaxValue：256
weight_decay	ContinuousParameterRange	MinValue：0.0，MaxValue：1.0

IP 见解数据格式

本节提供了 IP Insights 算法在训练和推理期间使用的可用输入和输出数据格式的示例。

主题

- IP 见解训练数据格式 (p. 1635)
- IP 见解推理数据格式 (p. 1636)

IP 见解训练数据格式

下面是 IP 见解算法的可用数据输入格式。Amazon SageMaker 内置算法遵循中所述的常见输入训练格式。用于训练的常见数据格式 (p. 1579). 但是，SageMaker IP 见解算法当前只支持 CSV 数据输入格式。

IP 见解训练数据输入格式

输入：CSV

CSV 文件一定具有两个列。第一列是对应于实体的唯一标识符的不透明字符串。第二列是十进制点表示法中实体访问事件的 IPv4 地址。

content-type：text/csv

```
entity_id_1, 192.168.1.2
entity_id_2, 10.10.1.2
```

IP 见解推理数据格式

下面是 IP 见解算法的可用输入和输出格式。Amazon SageMaker 内置算法遵循中所述的常见输入推理格式。用于推理的常见数据格式 (p. 1583). 但是，SageMaker IP 见解算法当前不支持 RecordIO 格式。

IP 见解输入请求格式

输入：CSV 格式

CSV 文件一定具有两个列。第一列是对应于实体的唯一标识符的不透明字符串。第二列是十进制点表示法中实体访问事件的 IPv4 地址。

content-type：text/csv

```
entity_id_1, 192.168.1.2
entity_id_2, 10.10.1.2
```

输入：JSON 格式

可以使用不同格式提供 JSON 数据。IP 见解遵循常见的 SageMaker 格式。有关推理格式的详细信息，请参阅用于推理的常见数据格式 (p. 1583).

content-type: application/json

```
{
  "instances": [
    {"data": {"features": {"values": ["entity_id_1", "192.168.1.2"]}}},
    {"features": ["entity_id_2", "10.10.1.2"]}
  ]
}
```

输入：JSONLINES 格式

JSON Llines 内容类型对于运行批量转换作业非常有用。有关 SageMaker 推理格式的详细信息，请参阅用于推理的常见数据格式 (p. 1583). 有关运行批处理转换作业的详细信息，请参阅使用批量转换 (p. 2229).

content-type: application/jsonlines

```
{"data": {"features": {"values": ["entity_id_1", "192.168.1.2"]}}},
{"features": ["entity_id_2", "10.10.1.2"]}]
```

IP 见解输出响应格式

输出: JSON 响应格式

SageMaker IP 见解算法的默认输出是`dot_product`在输入实体和 IP 地址之间。dot_product 表示模型考虑实体和 IP 地址的兼容性。这些区域有：`dot_product`是无界的。要预测某个事件是否异常，您需要根据定义的分布设置阈值。有关如何使用`dot_product`有关异常检测，请参阅SageMakerIP 见解算法简介.

accept: application/json

```
{
  "predictions": [
    {"dot_product": 0.0},
    {"dot_product": 2.0}
  ]
}
```

高级用户可以通过提供额外的内容类型参数来访问模型的学习实体和 IP 嵌入`verbose=True`转到接受标题。您可以使用`entity_embedding`和`ip_embedding`用于调试、可视化和理解模型。此外，您可以在其他机器学习技术（例如分类或群集）中使用这些嵌入内容。

接受: applicw/json; verbose=true

```
{
  "predictions": [
    {
        "dot_product": 0.0,
        "entity_embedding": [1.0, 0.0, 0.0],
        "ip_embedding": [0.0, 1.0, 0.0]
    },
    {
        "dot_product": 2.0,
        "entity_embedding": [1.0, 0.0, 1.0],
        "ip_embedding": [1.0, 0.0, 1.0]
    }
  ]
}
```

输出: JSONLINES 响应格式

accept: application/jsonlines

```
{"dot_product": 0.0}
{"dot_product": 2.0}
```

接受: 申请/jsonlines; verbose=true

```
{"dot_product": 0.0, "entity_embedding": [1.0, 0.0, 0.0], "ip_embedding": [0.0, 1.0, 0.0]}
{"dot_product": 2.0, "entity_embedding": [1.0, 0.0, 1.0], "ip_embedding": [1.0, 0.0, 1.0]}
```

K-Means 算法

K-means 是一种自主学习算法。它尝试在数据中寻找离散组，其中一个组的成员尽可能彼此相似，而与其他组的成员尽可能互不相同。您定义想要该算法用来确定相似性的属性。

Amazon SageMaker 使用修改版网络规模 k-means 聚类算法。与该算法的原始版本相比，Amazon SageMaker 使用的版本更加准确。与原始算法类似，它可扩展到大规模数据集并在训练时间方面加以改进。为此，Amazon SageMaker 使用的版本对小批量 (小型随机子集) 训练数据进行流式处理。有关小批量 k-means 的更多信息，请参阅网络规模 k-means 聚类。

k-means 算法预计生成表格数据，其中的行代表您要聚类的观察结果，而列代表观察结果的属性。每行中的 n 属性表示 n 维空间的一个点。这些点之间的欧几里得距离表示相应观察结果的相似度。该算法将具有类似属性值的观察结果分为一组 (与这些观察结果对应的点彼此离得更近)。有关 k-means 在 Amazon SageMaker 中的工作原理的更多信息，请参阅K-Means 聚类的工作原理 (p. 1638).

主题

- K-Means 算法的输入/输出接口 (p. 1638)
- K-Means 算法的 EC2 实例建议 (p. 1638)
- K-Means 示例笔记本 (p. 1638)
- K-Means 聚类的工作原理 (p. 1638)
- K-Means 超参数 (p. 1640)
- 优化 K-Means 模型 (p. 1642)
- K-Means 响应格式 (p. 1643)

K-Means 算法的输入/输出接口

在训练中，k-means 算法预计将在训练 通道（建议使用 `S3DataDistributionType=ShardedByS3Key`）中提供数据，并使用可选的测试 通道（建议使用 `S3DataDistributionType=FullyReplicated`）对数据计分。`recordIO-wrapped-protobuf` 和 `CSV` 格式均支持用于训练。您可以使用文件模式或管道模式，针对格式为 `recordIO-wrapped-protobuf` 或 `CSV` 的数据训练模型。

对于推理，支持 `text/csv`、`application/json` 和 `application/x-recordio-protobuf`。k-means 会为每个观察返回 `closest_cluster` 标签以及 `distance_to_cluster`。

有关输入和输出文件格式的更多信息，请参阅K-Means 响应格式 (p. 1643)（对于推理）和K-Means 示例笔记本 (p. 1638)。k-means 算法不支持多实例学习，在这种学习中，训练集由标记的“包”组成，每个包就是一个未标记实例的集合。

K-Means 算法的 EC2 实例建议

建议在 CPU 实例上训练 k-means。您可以在 GPU 实例上进行训练，但 GPU 训练应仅限 `p*.xlarge` 实例，因为每个实例只使用一个 GPU。

K-Means 示例笔记本

有关使用 SageMaker K-means 算法通过使用主成分分析识别的属性来对美国各县的人口进行分段的示例笔记本，请参阅。使用 Amazon SageMaker 分析美国人口普查数据以进行人口细分. 有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 创建笔记本实例并将其打开后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

K-Means 聚类的工作原理

K-means 是一种训练模型的算法，可将相似对象组合在一起。k-means 算法通过将输入数据集中的每个观察结果映射到 n 维空间的某个点 (其中 n 是观察结果的属性数量) 来完成此操作。例如，您的数据集可能包含某个特定位置的温度和湿度的观察结果，这些观察结果映射到 2 维空间的某些点 (t, h)。

> **Note**
>
> 聚类算法为自主型。在自主学习中，不使用可能与训练数据集中的对象相关联的标签。

在 k-means 聚类中，每个聚类都有一个中心。在模型训练期间，k-means 算法使用对应于数据集中每个观察结果的点与聚类中心之间的距离作为聚类的基础。您选择要创建的聚类数量 (k)。

例如，假设您要创建一个可识别手写数字的模型，并且您选择 MNIST 数据集用于训练。该数据集提供了数千个手写数字 (0 到 9) 的图像。在本例中，您可以选择创建 10 个聚类，每个数字 (0、1、...、9) 一个聚类。作为模型训练的一部分，k-means 算法将输入图像划分为 10 个聚类。

MNIST 数据集中的每张图就是一个 28x28 像素的图像，总共 784 像素。每个图像对应于 784 维空间里的一个点，类似于 2 维空间的点 (x, y)。要查找某个点所属的聚类，k-means 算法会查找该点与所有聚类中心之间的距离。然后，它会选择最靠近中心的聚类作为图像所属的聚类。

> **Note**
>
> Amazon SageMaker 使用自定义版本的算法，在该算法中，而不是指定由该算法创建。K群集，您可以选择通过指定额外的集群中心来提高模型准确性(K = k*x). 但是，该算法最终会将其减少为 k 个聚类。

在 SageMaker 中，您指定创建训练任务时的集群数量。有关更多信息，请参阅 `CreateTrainingJob`。在请求正文中，您添加 `HyperParameters` 字符串映射以指定 `k` 和 `extra_center_factor` 字符串。

下文概括了 k-means 如何适用于 SageMaker 中的模型训练：

1. 它确定初始 K 个聚类中心。

   Note

   在以下主题中，K 个聚类指的是 k * x，您将在创建模型训练作业时指定其中的 k 和 x。
2. 它将迭代输入训练数据并重新计算聚类中心。
3. 这样可将生成的聚类数量减少至 k (如果数据科学家已在请求中指定创建 k*x 个聚类)。

下面几节也介绍了数据科学家可能指定的一些参数，这些参数用于配置模型训练作业以作为 `HyperParameters` 字符串映射的一部分。

主题

- 第 1 步：确定初始集群中心 (p. 1639)
- 第 2 步：迭代训练数据集并计算集群中心 (p. 1639)
- 第 3 步：减少群集K到K (p. 1640)

第 1 步：确定初始集群中心

在 SageMaker 中使用 k-means 时，将从观察结果中选择随机采样的小批量初始集群中心。选择下列策略之一来确定这些初始聚类中心的选择方式：

- 随机方法 — 随机选择K作为集群中心。例如，您可以选择一个指向 784 维空间的聚类中心 (对应于 MNIST 训练数据集中的任意 10 个图像)。
- k-means++ 方法，其工作原理如下所示：
  1. 从一个聚类开始，并确定其中心。您从训练数据集中随机选择一个观察结果，并将对应于该观察结果的点用作聚类中心。例如，在 MNIST 数据集中，随机选择一个手写数字图像。然后，在 784 维空间中选择与该图像对应的点作为聚类中心。这是聚类中心 1。
  2. 确定聚类 2 的中心。从训练数据集的剩余观察结果中随机选择一个观察结果。选择一个不同于之前所选的观察结果。此观察结果对应于远离聚类中心 1 的一个点。以 MNIST 数据集为例，您需要执行以下操作：
     - 对于每张剩余的图像，找出相应点与聚类中心 1 之间的距离。对距离求平方，并分配一个与距离平方成正比的概率。这样一来，不同于之前所选的图像被选作聚类中心 2 的概率更高。
     - 基于上一步中分配的概率，随机选择一个图像。对应于该图像的点便是聚类中心 2。
  3. 重复步骤 2 找到聚类中心 3。这次，找出剩余图像与聚类中心 2 之间的距离。
  4. 重复该过程，直到您拥有 K 个聚类中心。

要在 SageMaker 中训练模型，请创建一个训练作业。在请求中，您通过指定以下 `HyperParameters` 字符串映射来提供配置信息：

- 要指定待创建的聚类数量，请添加 `k` 字符串。
- 为提高准确性，请添加可选的 `extra_center_factor` 字符串。
- 要指定希望用于确定初始聚类中心的策略，请添加 `init_method` 字符串并将其值设置为 `random` 或 `k-means++`。

有关 SageMaker k-means 估算器的更多信息，请参阅。k-means中的Amazon SageMaker Python 开发工具包文档中 ）。

现在，您拥有一组初始聚类中心。

第 2 步：迭代训练数据集并计算集群中心

您在上一步创建的聚类中心大多数是随机的，并且有一些关于训练数据集的考量。在此步骤中，您使用训练数据集将这些中心移向真正的聚类中心。该算法将迭代训练数据集，并重新计算 K 个聚类中心。

1. 从训练数据集读取小批量观察结果 (从所有记录中随机选择的小型子集)，然后执行以下操作。

   Note

   创建模型训练作业时，请在 `mini_batch_size` 字符串映射的 `HyperParameters` 字符串中指定批次大小。

   a. 将小批量中的所有观察结果分配到其中一个具有最近聚类中心的聚类。
   b. 计算分配到每个聚类的观察结果的数量。然后，计算每个聚类分配的新点比例。

      例如，考虑以下聚类：

      聚类 c1 = 100 个之前分配的点。您在此步骤中从小批量添加了 25 个点。

      聚类 c2 = 150 个之前分配的点。您在此步骤中从小批量添加了 40 个点。

      聚类 c3 = 450 个之前分配的点。您在此步骤中从小批量添加了 5 个点。

      按如下方式计算分配给每个聚类的新点比例：

      ```
      p1 = proportion of points assigned to c1 = 25/(100+25)
      p2 = proportion of points assigned to c2 = 40/(150+40)
      p3 = proportion of points assigned to c3 = 5/(450+5)
      ```

   c. 计算添加到每个聚类的新点的中心：

      ```
      d1 = center of the new points added to cluster 1
      d2 = center of the new points added to cluster 2
      d3 = center of the new points added to cluster 3
      ```

   d. 按如下方式计算加权平均数，以查找更新后的聚类中心：

      ```
      Center of cluster 1 = ((1 - p1) * center of cluster 1) + (p1 * d1)
      Center of cluster 2 = ((1 - p2) * center of cluster 2) + (p2 * d2)
      Center of cluster 3 = ((1 - p3) * center of cluster 3) + (p3 * d3)
      ```

2. 读取下一个小批量，并重复步骤 1 以重新计算聚类中心。
3. 有关小批量 k-means 的更多信息，请参阅网络规模 k-means 聚类。

第 3 步：减少群集K到K

如果算法创建K集群 —(K = k*x)哪里x大于 1-那么它会减少K集群到K集群。(有关更多信息，请参阅之前讨论的 `extra_center_factor`。) 它通过使用 Lloyd 的方法来执行此操作kmeans++初始化到K集群中心。有关 Lloyd 的方法的更多信息，请参阅 k-means 聚类。

K-Means 超参数

在 `CreateTrainingJob` 请求中，您指定要使用的训练算法。您还可以指定算法特定的超参数作为字符串到字符串映射。下表列出了 Amazon SageMaker 提供的 k-means 训练算法的超参数。有关 k-means 聚类工作原理的更多信息，请参阅K-Means 聚类的工作原理 (p. 1638)。

参数名称	描述
`feature_dim`	输入数据中的特征数。 必填 有效值：正整数

参数名称	描述
k	所需聚类的数量。 必填 有效值：正整数
epochs	对训练数据完成的扫描次数。 可选 有效值：正整数 默认值：1
eval_metrics	一个用于报告模型分数的指标类型的 JSON 列表。允许的值包括：msd对于均值平方偏差和ssd对于平方距离和。如果提供了测试数据，则会针对所请求的每个指标报告分数。 可选 有效值：或者[\"msd\"]要么[\"ssd\"]要么[\"msd\",\"ssd\"]. 默认值：[\"msd\"]
extra_center_factor	该算法会在运行时创建 K 个中心 = num_clusters * extra_center_factor，并在最终生成模型时将中心数量从 K 减少至 k。 可选 有效值：或者为正整数或auto. 默认值：auto
half_life_time_size	用于确定计算聚类均值时赋予观察的权重。随着观察到的点越来越多，此权重呈指数倍衰减。当首次观察到一个点的情况下，在计算聚类均值时，它的权重为 1。选择指数衰减函数的衰减常数，以便在观察 half_life_time_size 个点后，其权重为 1/2。如果设置为 0，则没有衰减。 可选 有效值：非负整数 默认值：0
init_method	算法选择初始聚类中心的方法。标准 k-means 方法会随机选择这些方法。备用 k-means++ 方法会随机选择第一个聚类中心。然后，它通过加权中心选择来扩展其余初始聚类的位置，其概率分布与其余数据点到现有中心的距离的平方成比例。 可选 有效值：random 或 kmeans++。 默认值：random

参数名称	描述
local_lloyd_init_method	用于构建包含 k 个中心的最终模型的劳埃德最大期望算法 (EM) 过程的初始化方法。 可选 有效值：random 或 kmeans++。 默认值：kmeans++
local_lloyd_max_iter	用于构建包含 k 个中心的最终模型的劳埃德最大期望算法 (EM) 过程的最大迭代次数。 可选 有效值：正整数 默认值：300
local_lloyd_num_trials	在构建包含 k 个中心的最终模型时，运行具有最小损失的劳埃德最大期望算法 (EM) 过程的次数。 可选 有效值：或者为正整数或auto. 默认值：auto
local_lloyd_tol	用于构建包含 k 个中心的最终模型的劳埃德最大期望算法 (EM) 过程的提前停止损失变化的容忍度。 可选 有效值：浮点。范围为 [0, 1]。 默认值：0.0001
mini_batch_size	用于数据迭代器的每个小批量的观察次数。 可选 有效值：正整数 默认值：5000

优化 K-Means 模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

Amazon SageMaker k-means 算法是一种非监督算法，它将数据分组为成员尽可能相似的集群。因为它是非自主型的，所以不使用可优化超参数的验证数据集。但是，它确实接受测试数据集并发出指标，这些指标取决于每次训练结束时数据点与最终聚类中心之间的距离平方值。要找到报告有关测试数据集的最紧密聚类的模型，可以使用超参数优化作业。聚类可优化其成员的相似性。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

K-Means 算法计算的指标

k-means 算法在训练期间计算以下指标。在优化模型时，选择这些指标之一作为目标指标。

指标名称	描述	优化方向
test:msd	测试集中每个记录与模型最近中心之间的均方距离。	最小化
test:ssd	测试集中每个记录与模型最近中心之间的平方距离总和。	最小化

可优化的 K-Means 超参数

使用以下超参数优化 Amazon SageMaker k-means 模型。对 k-means 目标指标影响最大的超参数为：mini_batch_size、extra_center_factor 和 init_method。优化超参数 epochs 通常会得到较小的改进。

参数名称	参数类型	建议的范围
epochs	IntegerParameterRanges	MinValue：1，最大值：10
extra_center_factor	IntegerParameterRanges	MinValue：4、最大值：10
init_method	CategoricalParameterRanges	['kmeans++', 'random']
mini_batch_size	IntegerParameterRanges	MinValue：3000，最大值：15000

K-Means 响应格式

所有 SageMaker 内置算法遵循中所述的常见输入推理格式。常见数据格式-推理. 本主题包含 SageMaker k-means 算法的可用输出格式列表。

JSON 响应格式

```
{
    "predictions": [
        {
            "closest_cluster": 1.0,
            "distance_to_cluster": 3.0,
        },
        {
            "closest_cluster": 2.0,
            "distance_to_cluster": 5.0,
        },

        ....
    ]
}
```

JSONLINES 响应格式

```
{"closest_cluster": 1.0, "distance_to_cluster": 3.0}
```

```
{"closest_cluster": 2.0, "distance_to_cluster": 5.0}
```

RECORDIO 响应格式

```
[
    Record = {
        features = {},
        label = {
            'closest_cluster': {
                keys: [],
                values: [1.0, 2.0]  # float32
            },
            'distance_to_cluster': {
                keys: [],
                values: [3.0, 5.0]  # float32
            },
        }
    }
]
```

CSV 响应格式

每行中的第一个值对应于 `closest_cluster`。

每行中的第二个值对应于 `distance_to_cluster`。

```
1.0,3.0
2.0,5.0
```

K 最近邻 (k-NN) 算法

Amazon SageMaker k 最近邻 (k-NN) 算法是基于索引的算法. 它使用非参数化方法来进行分类或回归。对于分类问题，算法查询最接近采样点的 k 个点，并返回其分类最常用的标签作为预测标签。对于回归问题，算法查询最接近采样点的 k 个点，并返回其特征值的平均值作为预测值。

使用 k-NN 算法的训练过程有三个步骤：采样、维度缩减和索引构建。采样过程可减少初始数据集的大小，这样可放到内存中。对于维度缩减，算法将减少数据的特征维度，以便减少 k-NN 模型占用的内存和降低推理延迟。我们提供了两种维度缩减方法：随机投影和快速 Johnson-Lindenstrauss 变换。通常情况下，您可以对高维 (d > 1000) 数据集使用维度缩减，以避免“诅咒维度”，后者会损害随着维度增加而变得稀疏的数据的统计分析。k-NN 训练的主要目标是构建索引。该索引能够有效地查找尚未确定其值或分类标签的点与用于推理的 k 个最近点之间的距离。

主题

- 输入/输出接口k-NN 算法 (p. 1644)
- k-NN 示例笔记本 (p. 1645)
- k-NN 算法的工作原理 (p. 1645)
- EC2实例推荐k-NN 算法 (p. 1646)
- k-NN 超参数 (p. 1646)
- 优化 k-NN 模型 (p. 1648)
- k-NN 训练输入的数据格式 (p. 1649)
- k-NN 请求和响应格式 (p. 1649)

输入/输出接口k-NN 算法

SageMaker k-NN 支持训练和测试数据通道。

- 对要采样并构建到 k-NN 索引中的数据使用训练通道。
- 使用测试通道可以向日志文件发送分数。分数按照每个小批量一行的方式列出：分数的 `classifier` 代表准确率，`regressor` 代表均方误差。

对于训练输入，k-NN 支持 `text/csv` 和 `application/x-recordio-protobuf` 数据格式。对于输入类型 `text/csv`，第一个 `label_size` 列解释为相应行的标签向量。您可以使用文件模式或管道模式，针对格式为 `recordIO-wrapped-protobuf` 或 `CSV` 的数据训练模型。

对于推理输入，k-NN 支持 `application/json`、`application/x-recordio-protobuf` 和 `text/csv` 数据格式。`text/csv` 格式接受 `label_size` 和编码参数。它采用值为 0 的 `label_size` 和 UTF-8 编码。

对于推理输出，k-NN 支持 `application/json` 和 `application/x-recordio-protobuf` 数据格式。这两种数据格式还支持详细输出模式。在详细输出模式中，API 为搜索结果提供从最小到最大排序的距离向量，以及标签向量中的对应元素。

对于批量变换，k-NN 支持对输入和输出使用 `application/jsonlines` 数据格式。示例输入如下所示：

```
content-type: application/jsonlines

{"features": [1.5, 16.0, 14.0, 23.0]}
{"data": {"features": {"values": [1.5, 16.0, 14.0, 23.0]}}
```

示例输出如下所示：

```
accept: application/jsonlines

{"predicted_label": 0.0}
{"predicted_label": 2.0}
```

有关输入和输出文件格式的更多信息，请参阅k-NN 训练输入的数据格式 (p. 1649)（对于训练）、k-NN 请求和响应格式 (p. 1649)（对于推理）和k-NN 示例笔记本 (p. 1645)。

k-NN 示例笔记本

有关使用 SageMaker k 最近邻算法从地质和森林服务数据预测荒野覆盖类型的示例笔记本，请参阅K-最近邻封面.

使用 Jupyter 笔记本实例在 SageMaker 中运行示例。要了解如何在 SageMaker 中创建和打开 Jupyter 笔记本实例，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147). 创建笔记本实例并将其打开后，选择SageMaker 示例选项卡以查看所有 SageMaker 示例笔记本的列表。在 Amazon 算法简介 部分中查找 K 最近邻笔记本。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

k-NN 算法的工作原理

第 1 步：示例

要指定从训练数据集中采样的数据点总数，请使用 `sample_size` 参数。例如，如果初始数据集有 1000 个数据点，而 `sample_size` 设置为 100，使用的实例总数为 2，则每个工作实例将采样 50 个点。总共将收集 100 个数据点。在采样运行过程中，时间与数据点数呈线性方式。

第 2 步：执行维度减少

k-NN 算法的当前实施有两种维度缩减方法。您可以在 `dimension_reduction_type` 超参数中指定方法。`sign` 方法指定一个随机投影，使用采用随机符号矩阵的线性投影，而 `fjlt` 方法指定一个快速 Johnson-Lindenstrauss 变换，这是一种基于傅立叶变换的方法。这两种方法保留 L2 和内积距离。当目标维度很大并且 CPU 推理能够提供更好的性能时，应该使用 `fjlt` 方法。这些方法的计算复杂性不同。`sign` 方法需要 O(ndk) 时间，以便将一批 n 个维度 d 数据点的维度减少到目标维度 k。`fjlt` 方法需要 O(nd log(d)) 时间，但仅在涉及的常量更大时使用。使用维度缩减会将噪点引入到数据中，这种噪点会降低预测准确度。

第 3 步：构建索引

在推理期间，算法将查询索引以获取样本点的 K 最近邻点。根据对数据点的参考，该算法进行分类或回归预测。它根据提供的分类标签或值进行预测。k-NN 提供三种不同类型的索引：平面索引、反向索引以及带有乘积量化的反向索引。您可以使用 `index_type` 参数指定类型。

序列化模型

当 k-NN 算法完成训练后，它会序列化三个文件以准备推理。

- model_algo-1：包含用于计算最近邻点的序列化索引。
- model_algo-1. 标签：包含序列化标签 (np.float32二进制格式），用于根据索引的查询结果计算预测标签。
- model_algo-1.json：包含 JSON 格式的模型元数据，该元数据存储`k`和`predictor_type`来自推断训练的超参数以及其他相关状态。

借助 k-NN 的当前实施，您可以修改元数据文件以更改预测的计算方式。例如，您可以将 `k` 更改为 10，或将 `predictor_type` 更改为 regressor。

```
{
  "k": 5,
  "predictor_type": "classifier",
  "dimension_reduction": {"type": "sign", "seed": 3, "target_dim": 10, "input_dim": 20},
  "normalize": False,
  "version": "1.0"
}
```

EC2实例推荐k-NN 算法

使用 k-NN 算法进行训练的实例建议

开始时，尝试在 CPU 上运行训练，例如使用 ml.m5.2xlarge 实例；或者在 GPU 上运行，例如使用 ml.p2.xlarge 实例。

使用 k-NN 算法进行推理的实例建议

来自 CPU 的推理请求通常具有比来自 GPU 的请求更低的平均延迟，因为当您使用 GPU 硬件时存在 CPU 到 GPU 的通信负担。但是，对于更大的批处理，GPU 通常具有更高的吞吐量。

k-NN 超参数

参数名称	描述
`feature_dim`	输入数据中的特征数。 必填 有效值：正整数。
`k`	最近邻点的数量。 必填 有效值：正整数
`predictor_type`	要在数据标签上使用的推理类型。 必填

参数名称	描述
	有效值：对于分类为 classifier；对于回归为 regressor。
sample_size	要从训练数据集中采样的数据点数。 必填 有效值：正整数
dimension_reduction_target	要缩减到的目标维度。 必需当你指定dimension_reduction_type参数。 有效值：大于 0 且 feature_dim 小于的正整数。
dimension_reduction_type	维度缩减方法的类型。 可选 有效值：对于随机投影为 sign；对于快速 Johnson-Lindenstrauss 变换为 fjlt。 默认值：无维度减少
faiss_index_ivf_nlists	当 index_type 为 faiss.IVFFlat 或 faiss.IVFPQ 时，在索引中构建的质心数。 可选 有效值：正整数 默认值：auto，这将解析为 sqrt(sample_size)。
faiss_index_pq_m	当 index_type 设置为 faiss.IVFPQ 时，要在索引中构造的向量子组件的数量。 FaceBook AI 相似性搜索 (FAISS) 库要求faiss_index_pq_m是的除数数据维度. 如果 faiss_index_pq_m 不是数据维度的除数，我们会将数据维度增加到可被 faiss_index_pq_m 整除的最小整数。如果未应用维度缩减，则算法会添加零填充。如果应用了维度缩减，算法 会增加 dimension_reduction_target 超级参数的值。 可选 有效值：以下正整数之一： 1、2、3、4、8、12、16、20、24、28、32、40、48、56、64、96
index_metric	在查找最近邻点时用于测量点之间距离的指标。使用训练时index_type设置为faiss.IVFPQ、这INNER_PRODUCT距离和COSINE不相似性支持。 可选 有效值：L2对于欧几里德-距离来说，INNER_ 产品对于内部产品的距离，余弦对于余弦相似性。 默认值：L2

参数名称	描述
index_type	索引类型。 可选 有效值：faiss.Flat、faiss.IVFFlat、faiss.IVFPQ。 默认值：faiss.Flat
mini_batch_size	用于数据迭代器的每个小批量的观察次数。 可选 有效值：正整数 默认值：5000

优化 k-NN 模型

Amazon SageMaker k 最近邻算法是一个指导式算法。该算法使用测试数据集，并发出关于分类任务的准确度或关于回归任务的均方误差的指标。这些准确度指标将其各自任务的模型预测与经验测试数据提供的真实值进行比较。要查找报告测试数据集中最高准确度或最低错误的最佳模型，请为 k-NN 运行超参数优化作业。

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以选择一个适合算法的预测任务的目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。超参数仅用于帮助估计模型参数，训练模型不使用超参数来进行预测。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

k-NN 算法计算的指标

k 最近邻算法在训练期间计算下表中的两个指标之一，具体取决于 predictor_type 超参数指定的任务类型。

- classifier 指定分类任务并计算 test:accuracy
- regressor 指定回归任务并计算 test:mse。

选择适合在优化模型时用来计算相关目标指标的任务类型的 predictor_type 值。

指标名称	描述	优化方向
test:accuracy	当 predictor_type 设置为 classifier 时，k-NN 会将基于 k 最近邻点标签的平均值的预测标签，与测试通道数据中提供的真实值标签进行比较。报告的准确度范围为从 0.0 (0%) 到 1.0 (100%)。	最大化
test:mse	当 predictor_type 设置为 regressor 时，k-NN 会将基于 k 最近邻点标签的平均值的预测标签，与测试通道数据中提供的真实值标签进行比较。通过比较两个标签来计算均方误差。	最小化

可优化的 k-NN 超参数

使用以下超参数优化 Amazon SageMaker k 最近邻模型。

参数名称	参数类型	建议的范围
k	IntegerParameterRanges	MinValue：1，MaxValue：1024
sample_size	IntegerParameterRanges	MinValue：256，MaxValue：20000000

k-NN 训练输入的数据格式

所有 Amazon SageMaker 内置算法遵守中所述的常见输入训练格式。常见数据格式 — 训练. 本主题包含 SageMaker k 最近邻算法的可用输入格式列表。

CSV 数据格式

content-type: text/csv; label_size=1

```
4,1.2,1.3,9.6,20.3
```

第一个 label_size 列解释为相应行的标签向量。

RECORDIO 数据格式

content-type: application/x-recordio-protobuf

```
[
    Record = {
        features = {
            'values': {
                values: [1.2, 1.3, 9.6, 20.3]  # float32
            }
        },
        label = {
            'values': {
                values: [4]  # float32
            }
        }
    }
]


}
```

k-NN 请求和响应格式

所有 Amazon SageMaker 内置算法遵守中所述的常见输入推理格式。常见数据格式 — 推理. 本主题包含 SageMaker k 最近邻算法的可用输出格式列表。

输入：CSV 请求格式

content-type: text/csv

```
1.2,1.3,9.6,20.3
```

这接受 label_size 或编码参数。它采用值为 0 的 label_size 和 utf-8 编码。

输入：JSON 请求格式

content-type: application/json

```
{
  "instances": [
    {"data": {"features": {"values": [-3, -1, -4, 2]}}},
    {"features": [3.0, 0.1, 0.04, 0.002]}]
}
```

输入：JSONLINES 请求格式

content-type: application/jsonlines

```
{"features": [1.5, 16.0, 14.0, 23.0]}
{"data": {"features": {"values": [1.5, 16.0, 14.0, 23.0]}}
```

输入：RECORDIO 请求格式

content-type: application/x-recordio-protobuf

```
[
    Record = {
        features = {
            'values': {
                values: [-3, -1, -4, 2]  # float32
            }
        },
        label = {}
    },
    Record = {
        features = {
            'values': {
                values: [3.0, 0.1, 0.04, 0.002]  # float32
            }
        },
        label = {}
    },
]
```

输出: JSON 响应格式

accept: application/json

```
{
  "predictions": [
    {"predicted_label": 0.0},
    {"predicted_label": 2.0}
  ]
}
```

输出: JSONLINES 响应格式

accept: application/jsonlines

```
{"predicted_label": 0.0}
{"predicted_label": 2.0}
```

输出: 详细 JSON 响应格式

在详细模式中，API 为搜索结果提供从最小到最大排序的距离向量，以及标签向量中的对应元素。在此示例中，k 设置为 3。

accept: application/json; verbose=true

```
{
  "predictions": [
    {
        "predicted_label": 0.0,
        "distances": [3.11792408, 3.89746071, 6.32548437],
        "labels": [0.0, 1.0, 0.0]
    },
    {
        "predicted_label": 2.0,
        "distances": [1.08470316, 3.04917915, 5.25393973],
        "labels": [2.0, 2.0, 0.0]
    }
  ]
}
```

输出: RECORDIO-PROTOBUF 响应格式

content-type: application/x-recordio-protobuf

```
[
    Record = {
        features = {},
        label = {
            'predicted_label': {
                values: [0.0]  # float32
            }
        }
    },
    Record = {
        features = {},
        label = {
            'predicted_label': {
                values: [2.0]  # float32
            }
        }
    }
]
```

输出: 详细 RECORDIO-PROTOBUF 响应格式

在详细模式中，API 为搜索结果提供从最小到最大排序的距离向量，以及标签向量中的对应元素。在此示例中，k 设置为 3。

accept: application/x-recordio-protobuf; verbose=true

```
[
    Record = {
        features = {},
        label = {
            'predicted_label': {
                values: [0.0]  # float32
            },
            'distances': {
                values: [3.11792408, 3.89746071, 6.32548437]  # float32
            },
```

```
            'labels': {
                values: [0.0, 1.0, 0.0]  # float32
            }
        }
    },
    Record = {
        features = {},
        label = {
            'predicted_label': {
                values: [0.0]  # float32
            },
            'distances': {
                values: [1.08470316, 3.04917915, 5.25393973]  # float32
            },
            'labels': {
                values: [2.0, 2.0, 0.0]  # float32
            }
        }
    }
]
```

k-NN 算法的示例输出

对于 regressor 任务：

```
[06/08/2018 20:15:33 INFO 140026520049408] #test_score (algo-1) : ('mse',
 0.013333333333333334)
```

对于 classifier 任务：

```
[06/08/2018 20:15:46 INFO 140285487171328] #test_score (algo-1) : ('accuracy',
 0.98666666666666669)
```

潜在狄利克雷分配 (LDA) 算法

Amazon SageMaker 潜在狄利克雷分配 (LDA) 算法是一种自主学习算法，它尝试将一组观察描述为不同类别的组合。LDA 最常用于发现通过文本语料库中的文档共享的用户指定数量的主题。这里，每个观察都是一个文档，特征是每个单词的存在 (或出现次数计数)，类别是主题。由于该方法是自主的，因此未预先指定主题，并且不能保证与人对文档进行自然分类的方式一致。以每个文档中单词的出现概率形式来学习主题。每个文档进而被描述为主题的混合。

具有类似主题混合的两个文档的确切内容将不相同。但总的来说，您希望这些文档更频繁地使用一个共享的单词子集，而不是与来自不同主题混合的文档进行比较。这将允许 LDA 发现这些单词组并使用它们来形成主题。举一个极其简单的例子，给定一组文档，其中出现的唯一单词是：eat、sleep、play、meow 和 bark，LDA 可能会生成类似下面内容的主题：

主题	eat	sleep	play	meow	bark
主题 1	0.1	0.3	0.2	0.4	0.0
主题 2	0.2	0.1	0.4	0.0	0.3

您可以推断出，更有可能归入主题 1 的文档与猫有关 (猫更有可能与 meow 和 sleep 相关)，归入主题 2 的文档与狗有关 (play 和 bark 更适合描述狗)。即便单词狗和猫从未在任何文本中出现，也可以找到这些主题。

主题

- 在潜在狄利克雷分配 (LDA) 和神经主题模型 (NTM) 之间进行选择 (p. 1653)

- DeepAR 算法的输入/输出接口 (p. 1653)
- LDA 算法的 EC2 实例建议 (p. 1653)
- LDA 示例笔记本 (p. 1653)
- LDA 工作原理 (p. 1654)
- LDA 超参数 (p. 1655)
- 优化 LDA 模型 (p. 1656)

在潜在狄利克雷分配 (LDA) 和神经主题模型 (NTM) 之间进行选择

主题模型通常用于从语料库中生成主题，这些主题：(1) 连贯地封装语义含义；(2) 很好地描述文档。因此，主题模型旨在尽量减少困惑并最大限度地提高主题的一致性。

Plexity 是一种内在的语言建模评估指标，用于衡量测试数据中每个字几何平均值可能性的逆数。困惑度得分较低表明泛化性能更好。研究表明，每个单词计算的可能性往往与人类判断不一致，可能完全不相关，因此引入了主题的连贯性。从模型中推断的每个主题都由单词组成，主题一致性计算为模型中该特定主题的前 N 个词。它通常被定义为该主题中单词的配对词相似度分数的平均值或中位数，例如 Pointtwise 互信息 (PMI)。有前途的模型可以生成具有较高主题一致性分数的连贯性主题

虽然目标是训练一种能够最大限度地减少困惑并最大限度地提高主题一致性的主题模型，但是与 LDA 和 NTM 往往存在权衡。亚马逊 Dinget 等人，2018 年的最近研究表明，NTM 在实现高主题一致性方面有希望，但接受崩溃的吉布斯抽样培训的 LDA 会带来更大的困惑。困惑和主题连贯性之间存在权衡。从硬件和计算能力的实用性角度来看，SageMaker NTM 硬件比 LDA 更灵活，并且可以更好地扩展，因为 NTM 可以在 CPU 和 GPU 上运行，并且可以跨多个 GPU 实例并行处理，而 LDA 只支持单实例 CPU 训练。

主题

- DeepAR 算法的输入/输出接口 (p. 1653)
- LDA 算法的 EC2 实例建议 (p. 1653)
- LDA 示例笔记本 (p. 1653)
- LDA 工作原理 (p. 1654)
- LDA 超参数 (p. 1655)
- 优化 LDA 模型 (p. 1656)

DeepAR 算法的输入/输出接口

LDA 预期在训练渠道上提供数据，并 (可选) 支持一个测试渠道，该渠道由最终模型进行评分。LDA 支持 `recordIO-wrapped-protobuf` (密集和稀疏) 和 `csv` 文件格式。对于 `csv`，数据必须是密集的，并且维度等于记录数 * 词汇表大小。使用 recordIO 包装的 protobuf 时，LDA 可以在文件或管道模式下训练，但仅在文件模式下支持 `csv` 格式。

对于推理，支持 `text/csv`、`application/json` 和 `application/x-recordio-protobuf` 内容类型。也可以为 `application/json` 和 `application/x-recordio-protobuf` 传递稀疏数据。LDA 推理返回 `application/json` 或 `application/x-recordio-protobuf` 预测，其中包括每个观察的 `topic_mixture` 向量。

有关训练和推理格式的更多详细信息，请参阅LDA 示例笔记本 (p. 1653)。

LDA 算法的 EC2 实例建议

LDA 当前仅支持单实例 CPU 训练。建议使用 CPU 实例进行托管/推理。

LDA 示例笔记本

有关该示例笔记本介绍了如何在数据集上训练 SageMaker 潜在狄利克雷分配算法，然后如何部署训练模型以对输入文档中的主题混合进行推理，请参阅SageMaker LDA 简介. 有关如何创建和访问可用于

在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 创建笔记本实例并将其打开后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。使用 NTM 算法的主题建模示例笔记本位于 Amazon 算法简介部分中。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

LDA 工作原理

Amazon SageMaker LDA 是一种自主学习算法，它尝试将一组观察描述为不同类别的组合。这些类别本身就是关于特征的概率分布。LDA 是一个生成概率模型，这意味着它尝试为基于潜在变量的输出和输入的分布提供一个模型。这与判别模型相反，这些模型尝试学习输入如何映射到输出。

您可以使用 LDA 执行各种任务，从根据产品购买对客户进行聚类分析到音乐中的自动和声分析。但是，它最常与文本语料库中的主题建模相关联。观察被称为文档。特性集被称为词汇表。特性被称为单词。生成的类别被称为主题。

> Note
>
> 词形还原显著提高了算法的性能和准确性。考虑预先处理任何输入文本数据。

一个 LDA 模型由两个参数定义：

- α - 对主题概率的预先估计 (换句话说，给定文档中每个主题出现的平均频率)。
- β - 一个 k 主题集合，在文档语料库中使用的词汇表中为每个主题提供一个概率分布，也称为“主题-单词分布”。

LDA 是一个“单词袋”模型，这意味着单词的顺序无关紧要。LDA 是一个生成模型，其中通过选择主题混合 θ ~ Dirichlet(α) 来逐词生成每个文档。

对于文档中的每个单词：

- 选择主题 z ~ Multinomial(θ)
- 选择相应的主题-单词分布 β_z。
- 绘制单词 w ~ Multinomial(β_z)。

在训练模型时，目标是找到参数 α 和 β，这两个参数可最大化模型生成的文本语料库的概率。

估计 LDA 模型的最常用方法是使用 Gibbs 抽样或期望最大化 (EM) 技术。Amazon SageMaker LDA 使用张量谱分解。这提供了几个优势：

- 理论保证结果。标准 EM 方法保证仅收敛到局部最优，通常质量较差。
- 尴尬平行。就输入文档而言，可在训练和推理方面对工作进行划分。EM 方法和 Gibbs 采样方法可以并行化，但这并不容易。
- 速度快。虽然 EM 方法的迭代成本较低，但其收敛速度较慢。Gibbs 采样也受到缓慢收敛速度的影响，而且还需要大量样本。

在较高级别，张量分解算法遵循此过程：

1. 目标是计算 V x V x V 张量的谱分解，这将总结语料库中的文档矩量。V 是词汇表大小 (换句话说，所有文档中不同单词的数目)。此张量的谱分量是 LDA 参数 α 和 β，这将最大化文档语料库的总体可能性。但是，由于词汇表比较大，因此该 V x V x V 张量过大，无法存储在内存中。
2. 相反，它使用 V x V 矩量矩阵 (步骤 1 中的张量的二维类比) 来查找维度 V x k 的白化矩阵。此矩阵可用于将 V x V 矩量矩阵转换为 k x k 单位矩阵。k 是模型中的主题数。
3. 之后，相同的白化矩阵可用于查找更小的 k x k x k 张量。在进行谱分析时，此张量具有与 V x V x V 张量的分量有简单关系的分量。
4. 交替最小二乘用于分解较小的 k x k x k 张量。这将大大减少内存消耗并加快速度。可以通过在谱分解中“取消白化”这些输出来查找参数 α 和 β。

在找到 LDA 模型的参数后，您可以查找每个文档的主题混合。您使用随机梯度下降法来最大化观察与这些数据相对应的给定主题混合的似然函数。

可以通过增加在训练中查找的主题数，然后筛选出质量较差的主题来改善主题质量。实际上，这是在 SageMaker LDA 中自动完成的：还将计算 25% 的主题，并且仅返回具有最大的关联狄利克雷先验的主题。要执行进一步的主题筛选和分析，您可以增加主题计数并修改生成的 LDA 模型，如下所示：

```
> import mxnet as mx
> alpha, beta = mx.ndarray.load('model.tar.gz')
> # modify alpha and beta
> mx.nd.save('new_model.tar.gz', [new_alpha, new_beta])
> # upload to S3 and create new SageMaker model using the console
```

有关 LDA 和 SageMaker 实施的算法的更多信息，请参阅：

- Animashree Anandkumar、Rong Ge、Daniel Hsu、Sham M Kakade 和 Matus Telgarsky。学习潜在变量模型的张量分解 (Tensor Decompositions for Learning Latent Variable Models)，机器学习研究杂志 (Journal of Machine Learning Research)，15:2773–2832，2014。
- David M Blei、Andrew Y Ng 和 Michael I Jordan。潜在狄利克雷分配。机器学习研究杂志 (Journal of Machine Learning Research)，3(Jan):993–1022，2003。
- Thomas L Griffiths 和 Mark Steyvers。发现科学主题 (Finding Scientific Topics)。美国国家科学院院刊 (Proceedings of the National Academy of Sciences)，101(suppl 1):5228–5235，2004。
- Tamara G Kolda 和 Brett W Bader。张量分解和应用程序 (Tensor Decompositions and Applications)。SIAM Review，51(3):455–500，2009。

LDA 超参数

在 `CreateTrainingJob` 请求中，您可以指定训练算法。您还可以指定算法特定的超参数作为字符串到字符串映射。下表列出了为 Amazon SageMaker 提供的 LDA 训练算法的超参数。有关更多信息，请参阅 LDA 工作原理 (p. 1654)。

参数名称	描述
`num_topics`	要在数据中查找的 LDA 的主题数。 必填 有效值：正整数
`feature_dim`	输入文档语料库的词汇表的大小。 必填 有效值：正整数
`mini_batch_size`	输入文档语料库中的文档的总数。 必填 有效值：正整数
`alpha0`	浓度参数的初始猜测：狄利克雷先验元素之和。小的值更有可能产生稀疏的主题混合，大的值 (大于 1.0) 会产生更均匀的混合。 可选 有效值：正值浮动

参数名称	描述
	默认值：1.0
max_restarts	在算法的交替最小二乘 (ALS) 谱分解阶段执行的重启次数。可用于通过额外计算来寻找更好的质量局部最小值，但通常不应进行调整。 可选 有效值：正整数 默认值：10
max_iterations	在算法的 ALS 阶段执行的迭代的最大次数。可用于通过额外计算来寻找更好的质量最小值，但通常不应进行调整。 可选 有效值：正整数 默认值：1000
tol	算法的 ALS 阶段的目标容错。可用于通过额外计算来寻找更好的质量最小值，但通常不应进行调整。 可选 有效值：正值浮动 默认值：1e-8

优化 LDA 模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

LDA 是一种自主型主题建模算法，它尝试将一组观察（文档）描述为不同类别（主题）的组合。“每单词对数似然”(PWLL) 指标评估一组学习主题（LDA 模型）准确描述测试文档数据集的可能性。较大的 PWLL 值表明测试数据更可能由 LDA 模型描述。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

LDA 算法计算的指标

LDA 算法在训练期间报告单个指标：`test:pwll`. 在优化模型时，选择此指标作为目标指标。

指标名称	描述	优化方向
test:pwll	测试数据集上的每单词对数似然。学习的 LDA 模型准确地描述测试数据集的可能性。	最大化

可优化的 LDA 超参数

您可以优化 LDA 算法的以下超参数。`alpha0` 和 `num_topics` 这两个超参数都会影响到 LDA 目标指标 (`test:pwll`)。如果您还不知道这些超参数的最佳值（即最大化每单词对数似然并生成准确的 LDA 模型），则自动模型调整功能可以帮助您找出这些最佳值。

参数名称	参数类型	建议的范围
alpha0	ContinuousParameterRanges	MinValue：0.1，MaxValue：10
num_topics	IntegerParameterRanges	MinValue：1，MaxValue：150

线性学习器算法

线性模型 是用于求解分类或回归问题的指导式学习算法。对于输入，向模型提供带标记的示例 (x, y)。x 是一个高维度向量，y 是一个数字标签。对于二元分类问题，该标签必须是 0 或 1。对于多类别分类问题，标签必须在 0 到 num_classes - 1 之间。对于回归问题，y 是一个实数。该算法学习线性函数或线性阈值函数（对于分类问题）进行分类，并将向量 x 映射到标签 y 的近似值。

Amazon SageMaker 线性学习器算法为分类和回归问题提供了一种解决方案。使用 SageMaker 算法，您可以同时探索不同的训练目标，并从验证集中选择最佳解决方案。您还可以探索大量模型并选择最佳模型。最佳模型优化以下任一项：

- 连续目标，如均方根误差、交叉熵损失、绝对误差。
- 适合分类的离散目标，例如 F1 度量、查准率、查全率或准确率。

与仅为连续目标提供解决方案的方法相比，SageMaker 线性学习器算法提供了显著高于原始超参数优化技术的速度提升。它也更为方便。

线性学习器算法需要一个数据矩阵，其中行表示观察，列表示特征的维度。它还需要一个包含与数据点匹配的标签的附加列。Amazon 至少 SageMaker 线性学习器要求您指定输入和输出数据位置，以及目标类型 (分类或回归) 作为参数。特征维度也是必需的。有关更多信息，请参阅 CreateTrainingJob。您可以在请求正文的 HyperParameters 字符串映射中指定其他参数。这些参数控制优化过程，或您训练的目标函数的细节。例如，纪元数、正则化和损失类型。

如果您使用的是托管的 Spot 训练，线性学习器算法支持用于快照模型状态的检查点.

主题

- 线性学习器算法的输入/输出接口 (p. 1657)
- 适用于线性学习器算法的 EC2 实例推荐 (p. 1658)
- 线性学习者样本笔记本 (p. 1658)
- 线性学习者的工作原理 (p. 1658)
- 线性学习器超参数 (p. 1659)
- 调整线性学习者模型 (p. 1667)
- 线性学员响应格式 (p. 1670)

线性学习器算法的输入/输出接口

Amazon SageMaker 线性学习器算法支持三个数据通道：训练、验证（可选）和测试（可选）。如果您提供验证数据，S3DataDistributionType 应该为 FullyReplicated。该算法记录每个纪元中的验证损失，并使用验证数据的样本来校准和选择最佳模型。如果您不提供验证数据，该算法会使用训练数据的样本来校准和选择该模型。如果您提供了测试数据，则算法日志会包含最终模型的测试分数。

对于训练，线性学习器算法支持 recordIO-wrapped protobuf 和 CSV 格式。对于 application/x-recordio-protobuf 输入类型，只支持 Float32 张量。对于 text/csv 输入类型，第一列假定为标签，即预测的目标变量。您可以使用文件模式或管道模式，针对格式为 recordIO-wrapped-protobuf 或 CSV 的数据训练线性学习器模型。

对于推理，线性学习器算法支持 application/json、application/x-recordio-protobuf 和 text/csv 格式。在对新数据进行预测时，响应格式取决于模型类型。对于回归 (predictor_type='regressor')，score 是模型生成的预测。对于分类（predictor_type='binary_classifier' 或 predictor_type='multiclass_classifier'），模型返回 score 以及 predicted_label。predicted_label 是模型预测的类别，score 测量该预测的强度。

- 对于二元分类，predicted_label 是 0 或 1，而 score 是单个浮点数，表示算法认为标签应为 1 的强度。
- 对于多类别分类，predicted_class 是从 0 到 num_classes-1 的整数，而 score 是一个浮点数列表，每个类别对应一个浮点数。

要解释分类问题中的 score，您必须考虑使用的损失函数。如果 loss 超参数值为 logistic（对于二元分类）或 softmax_loss（对于多类别分类），则可以将 score 解释为相应类别的概率。这些是在 loss 值为 auto 默认值时线性学习器使用的损失值。但如果将 loss 设置为 hinge_loss，则不能将 score 解释为概率。这是因为铰链损失对应于支持向量分类器，该分类器不会生成概率估计值。

有关输入和输出文件格式的更多信息，请参阅 线性学员响应格式 (p. 1670)。有关推理格式的更多信息，请参阅 线性学习者样本笔记本 (p. 1658)。

适用于线性学习器算法的 EC2 实例推荐

您可以在单机或多机 CPU 和 GPU 实例上训练线性学习器算法。在测试过程中，我们没有发现大量证据表明多 GPU 计算机比单 GPU 计算机更快。结果可能会有所不同，具体取决于您的使用案例。

线性学习者样本笔记本

下表概述了解决亚马逊不同使用案例的各种样本笔记本。 SageMaker 线性学习者算法。

笔记本标题	描述
MNIST 数据集简介	使用 MNIST 数据集，我们训练二进制分类器来预测一位数字。
预测乳腺癌	使用 UCI 的乳腺癌数据集，我们训练一个模型来预测乳腺癌。
如何构建多类分类器？	使用 UCI 的 Covertype 数据集，我们演示了如何训练多类分类器。
如何为推理构建 Machine Learning (ML) 管道？	使用 Scikit-Learn 容器，我们演示了如何构建 end-to-end ML 管道。

有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 创建笔记本实例并打开后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例。使用线性学习算法的主题建模示例笔记本位于 Amazon 算法简介部分中。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

线性学习者的工作原理

在线性学习器算法实施中涉及三个步骤：预处理、训练和验证。

第 1 步：预处理

标准化（或称为特征缩放）是特定损失函数的重要预处理步骤，可以确保在数据集上进行训练的模型不会由单个特征的权重所主导。Amazon SageMaker 线性学习器算法提供了一个标准化选项，用于协助此预处理步骤。如果已启用标准化，则该算法首先处理数据的一个小样本，用于了解各个特征以及标签的平均值和标准偏差。然后，完整数据集中的各个特征转换为具有平均值零并进行缩放以获得单位标准偏差。

Note

为了获得最佳结果，请确保在训练之前对您的数据进行随机排布。采用未随机排布的数据进行训练可能会导致训练失败。

您可以配置线性学习器算法是否分别使用 `normalize_data` 和 `normalize_label` 超参数来标准化特征数据和标签。默认情况下为特征和标签启用了标准化以进行回归。只有特征可以针对二进制分类进行标准化，这是默认行为。

第 2 步：训练

使用线性学习器算法，您可以使用随机梯度下降 (SGD) 的分布式实现进行训练。您可以通过选择优化算法来控制优化过程。例如，您可以选择使用 Adam、AdaGrad、随机梯度下降或其他优化算法。您还可以指定其超参数，例如动量、学习率和学习率计划。如果您不确定要使用哪个算法或超参数值，请选择适用于大多数数据集的默认值。

在训练期间，您可以同时优化多个模型，每个模型的目标略有不同。例如，您可以改变 L1 或 L2 正则化并尝试不同的优化程序设置。

第 3 步：验证并设置阈值

并行训练多个模型时，在训练完成后针对验证集来评估模型，以便选择最优的模型。对于回归，最优模型是在验证集上实现最佳损失的模型。对于分类，使用验证集的样本来校准分类阈值。所选最优模型是在验证集上实现了最佳二进制分类选择条件的模型。此类条件的示例包括 F1 度量、准确率和交叉熵损失。

Note

如果算法未提供验证集，则无法评估和选择最优模型。要利用并行训练和模型选择，请确保您向算法提供了验证集。

线性学习器超参数

下表包含线性学习器算法的超参数。这些是由用户设置的参数，以便于从数据中评估模型参数。首先，按字母顺序列出必须设置的所需超参数。接下来，也按字母顺序列出可以设置的可选超参数。当超参数设置为`auto`，Amazon SageMaker 将自动计算并设置该超级参数的值。

参数名称	说明
`num_classes`	响应变量的分类数量。该算法假定分类标记为 `0`、...、`num_classes - 1`。 必需什么时候`predictor_type`是`multiclass_classifier`. 否则，算法将忽略它。 有效值：从 3 到 1000000 的整数
`predictor_type`	将目标变量的类型指定为二元分类、多类别分类或回归。 必填 有效值：`binary_classifier`、`multiclass_classifier` 或 `regressor`
`accuracy_top_k`	当计算多类别分类的前 k 个最大数准确率指标时，为 k 的值。如果模型将前 k 个最大数分数中的一个分配给实际标签，则将示例评分为正确。 可选 有效值：正整数 默认值：3

参数名称	说明
balance_multiclass_weights	指定是否使用分类权重，这使得每个分类在损失函数中具有相同的重要性。仅当 predictor_type 为 multiclass_classifier 时使用。 可选 有效值：true，false 默认值：false
beta_1	一阶矩估计的指数衰减率。仅当 optimizer 值为 adam 时适用。 可选 有效值：auto 或 0 和 1.0 之间的浮点值 默认值：auto
beta_2	二阶矩估计的指数衰减率。仅当 optimizer 值为 adam 时适用。 可选 有效值：auto 或 0 和 1.0 之间的浮点整数值 默认值：auto
bias_lr_mult	允许偏移项有不同的学习率。偏移的实际学习率是 learning_rate * bias_lr_mult。 可选 有效值：auto 或正的浮点整数 默认值：auto
bias_wd_mult	允许偏移项有不同的正则化。偏移的实际 L2 正则化权重是 wd * bias_wd_mult。默认情况下，偏移项上没有正则化。 可选 有效值：auto 或非负浮点整数 默认值：auto

参数名称	说明
binary_classifier_model_selection_criteria	当 predictor_type 设置为 binary_classifier 时，验证数据集的模型评估标准（或者，如果未提供验证数据集，则为训练数据集）。标准包括： • accuracy— 具有最高准确率的模型。 • f_beta— F1 分数最高的模型。默认值为 F1。 • precision_at_target_recall— 在给定的召回目标上具有最高准确率的模型。 • recall_at_target_precision— 在给定的准确率目标上具有最高召回率的模型。 • loss_function— 具有训练中使用的最低损失函数值的模型。 可选 有效值：accuracy、f_beta、precision_at_target_recall、recall_at_target_pr或 loss_function 默认值：accuracy
early_stopping_patience	如果在相关指标中没有改进，则在结束训练前等待的纪元数。如果您提供了 binary_classifier_model_selection_criteria 的值，则指标就是该值。否则，指标与为 loss 超参数指定的值相同。 该指标是在验证数据上评估的。如果未提供验证数据，则该指标始终与为 loss 超参数指定的值相同，并在训练数据上进行评估。要禁用提前停止，请将 early_stopping_patience 设置为大于为 epochs 指定的值。 可选 有效值：正整数 默认值：3
early_stopping_tolerance	用于评估损失改善的相对容差。如果损失改善除以上一个最佳损失的比率小于此值，则提前停止逻辑会认为改善是零。 可选 有效值：正浮点整数 默认值：0.001
epochs	扫描训练数据的最大次数。 可选 有效值：正整数 默认值：15

参数名称	说明
f_beta	在计算二元分类或多类别分类的 F 分数指标时使用的 Beta 值。如果为 binary_classifier_model_selection_criteria 指定的值为 f_beta，也会使用此选项。 可选 有效值：正浮点整数 默认值：1.0
feature_dim	输入数据中的特征数。 可选 有效值：auto 或正整数 默认值：auto
huber_delta	用于 Huber 损失的参数。在训练和指标评估过程中，对小于增量的错误计算 L2 损失，对大于增量的错误计算 L1 损失。 可选 有效值：正浮点整数 默认值：1.0
init_bias	偏移项的初始权重。 可选 有效值：浮点整数 默认值：0
init_method	设置用于模型权重的初始分布函数。函数包括： • uniform— 均匀分布在（-scale，+ scale）之间均匀分布 • normal— 正态分布，平均值 0 和 sigma 可选 有效值：uniform 或 normal 默认值：uniform
init_scale	扩展模型权重的初始均匀分布。仅当 init_method 超参数设置为 uniform 时适用。 可选 有效值：正浮点整数 默认值：0.07

参数名称	说明
init_sigma	正态分布的初始标准差。仅当 init_method 超参数设置为 normal 时适用。 可选 有效值：正浮点整数 默认值：0.01
l1	L1 正则化参数。如果您不希望使用 L1 正则化，将此值设置为 0。 可选 有效值：auto 或非负浮点值 默认值：auto
learning_rate	优化程序用于参数更新的步长。 可选 有效值：auto 或正的浮点整数 默认值：auto，其值取决于选择的优化程序。
loss	指定损失函数。 可用的损失函数及其默认值取决于 predictor_type 的值： • 如果 predictor_type 设置为 regressor，则可用的选项为 auto、squared_loss、absolute_loss、eps_insensitive_squared_loss、ep 和 huber_loss。auto 的默认值为 squared_loss。 • 如果 predictor_type 设置为 binary_classifier，则可用的选项为 auto、logistic 和 hinge_loss。auto 的默认值为 logistic。 • 如果 predictor_type 设置为 multiclass_classifier，则可用的选项为 auto 和 softmax_loss。auto 的默认值为 softmax_loss。 有效 值：auto、logistic、squared_loss、absolute_loss、hinge_loss、eps_inse 或 huber_loss 可选 默认值：auto
loss_insensitivity	epsilon 不敏感损失类型的参数。在训练和指标评估过程中，任何小于此值的误差都被认为是零。 可选 有效值：正浮点整数 默认值：0.01

参数名称	说明
lr_scheduler_factor	对于每个 lr_scheduler_step 超参数，学习率按此数量减少。仅当 use_lr_scheduler 超参数设置为 true 时适用。 可选 有效值：auto 或 0 和 1 之间正的浮点整数值 默认值：auto
lr_scheduler_minimum_lr	学习率降低到的值永远不会低于为 lr_scheduler_minimum_lr 设置的值。仅当 use_lr_scheduler 超参数设置为 true 时适用。 可选 有效值：auto 或正的浮点整数 默认值：auto
lr_scheduler_step	学习率下降之间的步骤数。仅当 use_lr_scheduler 超参数设置为 true 时适用。 可选 有效值：auto 或正整数 默认值：auto
margin	hinge_loss 函数的间隔。 可选 有效值：正浮点整数 默认值：1.0
mini_batch_size	用于数据迭代器的每个小批量的观察次数。 可选 有效值：正整数 默认值：1000
momentum	sgd 优化程序的动量。 可选 有效值：auto 或 0 和 1.0 之间的浮点整数值 默认值：auto
normalize_data	在训练之前标准化特征数据。数据标准化将各个特征的数据转换为具有平均值零并进行缩放以获得单位标准偏差。 可选 有效值：auto、true 或 false 默认值：true

参数名称	说明
normalize_label	对标签进行标准化。标签标准化将标签移动到平均值零，并将其缩放为具有单位标准偏差。 auto 默认值标准化回归问题的标签，但不标准化分类问题的标签。对于分类问题，如果您将 normalize_label 超参数设置为 true，该算法将忽略它。 可选 有效值：auto、true 或 false 默认值：auto
num_calibration_sample	验证数据集中用于模型校准的观察次数（在查找最佳阈值时）。 可选 有效值：auto 或正整数 默认值：auto
num_models	并行训练的模型数。对于默认值 auto，算法决定训练的并行模型数。一个模型根据给定的训练参数（正则化、优化程序、损失）进行训练，其余模型根据接近参数进行训练。 可选 有效值：auto 或正整数 默认值：auto
num_point_for_scaler	用于计算标准化或取消偏移项的数据点数量。 可选 有效值：正整数 默认值：10000
optimizer	要使用的优化算法。 可选 有效值： • auto— 默认值。 • sgd— 随机梯度下降。 • adam—适应性动量估计. • rmsprop— 基于梯度的优化技术，使用平方梯度的移动平均值来标准化梯度。 默认值：auto. 的默认设置auto是adam.

参数名称	说明
positive_example_weight_mult	在训练二元分类器时分配给正示例的权重。负示例的权重固定为 1。如果您希望算法选择一个权重，以使分类负示例与 正示例时发生的错误对训练损失有相同的影响，请指定 balanced。如果希望算法选择优化性能的权重，请指定 auto。 可选 有效值：balanced、auto 或正的浮点整数 默认值：1.0
quantile	分位数损失的分位数。对于分位数 q，模型将尝试生成预测，以便 true_label 的值大于概率 q 的预测。 可选 有效值：0 和 1 之间的浮点整数 默认值：0.5
target_precision	目标查准率。如果 binary_classifier_model_selection_criteria 是 recall_at_target_precision，则查准率保持为此值，同时查全率最大化。 可选 有效值：0 和 1.0 之间的浮点整数 默认值：0.8
target_recall	目标查全率。如果 binary_classifier_model_selection_criteria 是 precision_at_target_recall，则查全率保持在此值，同时查准率最大化。 可选 有效值：0 和 1.0 之间的浮点整数 默认值：0.8
unbias_data	在训练前对特性取消偏移以使平均值为 0。默认情况下，数据不带偏移，use_bias超参数设置为true. 可选 有效值：auto、true 或 false 默认值：auto
unbias_label	在训练前对标签取消偏移以使平均值为 0。仅当 use_bias 超参数设置为 true 时才适用于回归。 可选 有效值：auto、true 或 false 默认值：auto

参数名称	说明
use_bias	指定模型是否应包含偏移项，即线性等式中的截距项。 可选 有效值：true 或 false 默认值：true
use_lr_scheduler	是否对学习率使用计划程序。如果要使用计划程序，请指定 true。 可选 有效值：true 或 false 默认值：true
wd	权重衰减参数，也称为 L2 正则化参数。如果不希望使用 L2 正则化，请将此值设置为 0。 可选 有效值：auto 或非负浮点整数 默认值：auto

调整线性学习者模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

线性学习器算法还具有用于优化超参数的内部机制，与此处描述的自动模型优化功能无关。默认情况下，线性学习器算法通过并行训练多个模型来优化超参数。当您使用自动模型优化时，线性学习器内部优化机制功能自动关闭。这会将并行模型的数量 num_models 设置为 1。该算法会忽略您为 num_models 设置的任何值。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

由线性学习者算法计算的指标

线性学习器算法报告下表中的指标，这些指标是在训练期间计算的。选择其中一个作为目标指标。为避免过拟合，我们建议根据验证指标而不是训练指标优化模型。

指标名称	说明	优化方向
test:absolute_loss	最终模型在测试数据集上的绝对损失。此客观指标仅对回归有效。	最小化
test:binary_classification_accuracy	最终模型在测试数据集上的准确率。此客观指标仅对二进制分类有效。	最大化
test:binary_f_beta	最终模型在测试数据集上的 F-beta 分数。默认情况下，它是 F1 分数，这是查准率和查全率的调和平均数。此客观指标仅对二进制分类有效。	最大化
test:dcg	最终模型在测试数据集上的折扣累计收益。此客观指标仅适用于多类别分类。	最大化

指标名称	说明	优化方向
test:macro_f_beta	最终模型在测试数据集上的 F-beta 分数。此客观指标仅适用于多类别分类。	最大化
test:macro_precision	最终模型在测试数据集上的准确率分数。此客观指标仅适用于多类别分类。	最大化
test:macro_recall	最终模型在测试数据集上的查全率。此客观指标仅适用于多类别分类。	最大化
test:mse	最终模型在测试数据集上的均方误差。此客观指标仅对回归有效。	最小化
test:multiclass_accuracy	最终模型在测试数据集上的准确率。此客观指标仅适用于多类别分类。	最大化
test:multiclass_top_k_accuracy	在测试数据集上预测的前 k 个标签之间的准确率。如果您选择此指标作为目标，我们建议您使用accuracy_top_k超参数。此客观指标仅适用于多类别分类。	最大化
test:objective_loss	训练模型后测试数据集上的目标损失函数的平均值。默认情况下，损失是二元分类的逻辑损失和回归的平方损失。要将损失设置为其他类型，请使用 loss 超参数。	最小化
test:precision	最终模型在测试数据集上的查准率。如果您选择此指标作为目标，我们建议您通过将 binary_classifier_model_selection 超参数设置为 precision_at_target_recall 以及设置 target_recall 超参数的值，来设置目标查全率。此客观指标仅对二进制分类有效。	最大化
test:recall	最终模型在测试数据集上的查全率。如果您选择此指标作为目标，我们建议您通过将 binary_classifier_model_selection 超参数设置为 recall_at_target_precision 以及设置 target_precision 超参数的值，来设置目标准确率。此客观指标仅对二进制分类有效。	最大化
test:roc_auc_score	最终模型在测试数据集上的正在接收的操作特征曲线（ROC 曲线）。此客观指标仅对二进制分类有效。	最大化
validation:absolute_loss	最终模型在验证数据集上的绝对损失。此客观指标仅对回归有效。	最小化
validation:binary_classification_accuracy	最终模型在验证数据集上的准确率。此客观指标仅对二进制分类有效。	最大化
validation:binary_f_beta	最终模型在验证数据集上的 F-beta 分数。默认情况下，F-beta 分数是 F1 分数，这是validation:precision和validation:recall指标。此客观指标仅对二进制分类有效。	最大化
validation:dcg	最终模型在验证数据集上的折扣累计收益。此客观指标仅适用于多类别分类。	最大化

指标名称	说明	优化方向
validation:macro_f_beta	最终模型在验证数据集上的 F-beta 分数。此客观指标仅适用于多类别分类。	最大化
validation:macro_precision	最终模型在验证数据集上的准确率分数。此客观指标仅适用于多类别分类。	最大化
validation:macro_recall	最终模型在验证数据集上的查全率。此客观指标仅适用于多类别分类。	最大化
validation:mse	最终模型在验证数据集上的平方误差。此客观指标仅对回归有效。	最小化
validation:multiclass_accuracy	最终模型在验证数据集上的准确率。此客观指标仅适用于多类别分类。	最大化
validation:multiclass_top_k_accuracy	前 k 个标签在验证数据集上的准确率。如果您选择此指标作为目标，我们建议您使用accuracy_top_k超参数。此客观指标仅适用于多类别分类。	最大化
validation:objective_loss	每个纪元验证数据集上目标损失函数的平均值。默认情况下，损失是二元分类的逻辑损失和回归的平方损失。要将损失设置为其他类型，请使用 loss 超参数。	最小化
validation:precision	最终模型在验证数据集上的查准率。如果您选择此指标作为目标，我们建议您通过将 binary_classifier_model_selection 超参数设置为 precision_at_target_recall 以及设置 target_recall 超参数的值，来设置目标查全率。此客观指标仅对二进制分类有效。	最大化
validation:recall	最终模型在验证数据集上的查全率。如果您选择此指标作为目标，我们建议您通过将 binary_classifier_model_selection 超参数设置为 recall_at_target_precision 以及设置 target_precision 超参数的值，来设置目标准确率。此客观指标仅对二进制分类有效。	最大化
validation:roc_auc_score	最终模型在验证数据集上的接收特征曲线（ROC 曲线）下的区域。此客观指标仅对二进制分类有效。	最大化

调整线性学习者超参数

您可以使用以下超参数优化线性学习器模型。

参数名称	参数类型	建议的范围
wd	ContinuousParameterRanges	MinValue: 1e-7, MaxValue: 1
l1	ContinuousParameterRanges	MinValue: 1e-7, MaxValue: 1
learning_rate	ContinuousParameterRanges	MinValue: 1e-5, MaxValue: 1

参数名称	参数类型	建议的范围
mini_batch_size	IntegerParameterRanges	MinValue: 100, MaxValue: 5000
use_bias	CategoricalParameterRanges	[True, False]
positive_example_weight_mult	ContinuousParameterRanges	MinValue：1e-5，MaxValue：1e5

线性学员响应格式

JSON 响应格式

Amazon SageMaker 内置算法遵守中所述的常见输入推理格式。常见数据格式-推理. 下面是可用于 SageMaker 线性学习者算法。

二元分类

```
let response =   {
    "predictions":    [
        {
            "score": 0.4,
            "predicted_label": 0
        }
    ]
}
```

多类别分类

```
let response =   {
    "predictions":    [
        {
            "score": [0.1, 0.2, 0.4, 0.3],
            "predicted_label": 2
        }
    ]
}
```

回归

```
let response =   {
    "predictions":    [
        {
            "score": 0.4
        }
    ]
}
```

JSONLINES 响应格式

二元分类

```
{"score": 0.4, "predicted_label": 0}
```

多类别分类

```
{"score": [0.1, 0.2, 0.4, 0.3], "predicted_label": 2}
```

回归

```
{"score": 0.4}
```

RECORDIO 响应格式

二元分类

```
[
    Record = {
        features = {},
        label = {
            'score': {
                keys: [],
                values: [0.4]  # float32
            },
            'predicted_label': {
                keys: [],
                values: [0.0]  # float32
            }
        }
    }
]
```

多类别分类

```
[
    Record = {
    "features": [],
    "label":    {
            "score":  {
                    "values":   [0.1, 0.2, 0.3, 0.4]
            },
            "predicted_label":  {
                    "values":   [3]
            }
       },
    "uid":  "abc123",
    "metadata": "{created_at: '2017-06-03'}"
   }
]
```

回归

```
[
    Record = {
        features = {},
        label = {
            'score': {
                keys: [],
                values: [0.4]  # float32
            }
        }
    }
]
```

神经主题模型 (NTM) 算法

Amazon SageMaker NTM 是一种自主型学习算法，用于将文档语料库组织成话题其中包含基于统计分布的词组。例如，如果文档包含诸如“自行车”、“汽车”、“火车”、“里程”和“速度”之类的频繁出现的单词，则文档可

能共享关于“运输”的主题。主题建模可用于基于检测到的主题对文档进行分类或汇总，或者基于主题相似性检索信息或推荐内容。来自 NTM 学习的文档的主题被表征为潜在表示，因为主题是从语料库中观察到的单词分布推断出来的。主题的语义通常通过检查它们包含的最高排名词来推断。因为该方法是自主型的，所以只有主题的数量，而不是主题本身是预设的。此外，这些主题不能保证与人类自然地对文档进行分类的方式一致。

主题建模提供了一种根据所学主题可视化大型文档语料库内容的方法。与每个主题相关的文档可以根据其软主题标签进行索引或搜索。文档的潜在表示还可用于在主题空间中查找类似的文档。您还可以将主题模型学习的文档的潜在表示用于另一受监督算法 (如文档分类器) 的输入。因为预计文档的潜在表示会捕获底层文档的语义，所以部分基于这些表示的算法预计比仅基于词汇特征的算法表现更好。

虽然您可以同时使用 Amazon SageMaker NTM 和 LDA 算法进行主题建模，但它们是不同的算法，可能会在相同的输入数据上产生不同的结果。

有关 NTM 背后的数学的更多信息，请参阅文本处理的神经变分推理。

主题

- NTM 算法的输入/输出接口 (p. 1672)
- NTM 算法的 EC2 实例建议 (p. 1672)
- NTM 示例笔记本 (p. 1673)
- NTM 超参数 (p. 1673)
- 优化 NTM 模型 (p. 1675)
- NTM 响应格式 (p. 1676)

NTM 算法的输入/输出接口

Amazon SageMaker 神经主题模型支持四种数据通道：训练、验证、测试和辅助。验证、测试和辅助数据通道是可选的。如果您指定这些可选通道中的任何一个，则将这些通道的 `S3DataDistributionType` 参数值设置为 `FullyReplicated`。如果您提供验证数据，则会在每个纪元记录此数据的损失，一旦检测到验证损失未改善，模型就会停止训练。如果您不提供验证数据，算法将根据训练数据提早停止，但这可能会降低效率。如果您提供测试数据，则该算法将从最终模型中报告测试损失。

NTM 的训练、验证和测试数据通道同时支持 `recordIO-wrapped-protobuf`（密集和稀疏）和 `CSV` 文件格式。如果是 `CSV` 格式，对于不存在于相应文档中的单词，必须以零计数密集地表示每行，并且每行必须具有等于 (记录数) * (词汇表大小) 的维度。您可以使用文件模式或管道模式，针对格式为 `recordIO-wrapped-protobuf` 或 `CSV` 的数据训练模型。辅助通道用于提供包含词汇表的文本文件。通过提供词汇表文件，用户可以查看日志中输出的每个主题的顶级单词，而不是整数 ID。使用词汇表文件，还允许 NTM 计算 Word 嵌入主题一致性 (WETC) 分数，这是一个在日志中显示的新指标，可以有效地捕获每个主题中顶级单词之间的相似性。辅助通道的 `ContentType` 是 `text/plain`，每一行包含一个单词，顺序对应于数据中提供的整数 ID。词汇表文件必须命名为 `vocab.txt`，目前仅支持 UTF-8 编码。

对于推理，支持 `text/csv`、`application/json`、`application/jsonlines` 和 `application/x-recordio-protobuf` 内容类型。也可以为 `application/json` 和 `application/x-recordio-protobuf` 传递稀疏数据。NTM 推理返回 `application/json` 或 `application/x-recordio-protobuf` 预测，其中包括每个观察的 `topic_weights` 向量。

有关使用辅助通道和 WETC 分数的更多详细信息，请参阅博客帖子和配套的笔记本。有关如何计算 WETC 分数的更多信息，请参阅相干感知神经主题建模。我们将本文中所述的成对 WETC 用于 Amazon SageMaker 神经主题模型。

有关输入和输出文件格式的更多信息，请参阅NTM 响应格式 (p. 1676)（对于推理）和NTM 示例笔记本 (p. 1673)。

NTM 算法的 EC2 实例建议

NTM 训练支持 GPU 和 CPU 实例类型。我们推荐 GPU 实例，但是对于某些工作负载，CPU 实例可能会使训练成本降低。CPU 实例应足以进行推理。

NTM 示例笔记本

有关使用 SageMaker NTM 算法从已知主题分布的合成数据源中的文档内发现主题的示例笔记本，请参阅NTM 的基本功能简介. 有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 创建笔记本实例并打开之后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。使用 NTM 算法的主题建模示例笔记本位于 Amazon 算法简介部分中。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

NTM 超参数

参数名称	描述
`feature_dim`	数据集的词汇表大小。 必填 有效值：正整数（最小值：1，最大值：1,000,000）
`num_topics`	所需的主题数。 必填 有效值：正整数（最小值：2，最大值：1000）
`batch_norm`	在训练过程中是否使用批处理标准化。 可选 有效值：true 或 false 默认值：false
`clip_gradient`	每个梯度组件的最大幅度。 可选 有效值：浮点型（最小值：1e-3）
`encoder_layers`	编码器中的层数和每个层的输出大小。当设置为 auto 时，算法使用各自大小为 3 x `num_topics` 和 2 x `num_topics` 的两个层。 可选 有效值：以逗号分隔的正整数列表或自动 默认值：auto
`encoder_layers_activation`	要在编码器层中使用的激活函数。 可选 有效值： • `sigmoid`：Sigmoid 函数 • `tanh`：双曲正切 • `relu`：修正线性单元

参数名称	描述
	默认值：sigmoid
epochs	扫描训练数据的最大次数。 可选 有效值：正整数（最小值：1） 默认值：50
learning_rate	优化程序的学习率。 可选 有效值：浮点型（最小值：1e-6，最大值：1.0） 默认值：0.001
mini_batch_size	每个小批量中的示例数。 可选 有效值：正整数（最小值：1，最大值：10000） 默认值：256
num_patience_epochs	计算提前停止标准所依据的连续纪元数。当损失函数的变化在最近 num_patience_epochs 个纪元内低于指定 tolerance 时，将触发提前停止。要禁用提早停止，请将 num_patience_epochs 设置为一个大于 epochs 的值。 可选 有效值：正整数（最小值：1） 默认值：3
optimizer	用于训练的优化程序。 可选 有效值： • sgd：随机梯度下降 • adam：适应性动量估计 • adagrad：自适应梯度算法 • adadelta：自适应学习率算法 • rmsprop： 均方根传播 默认值：adadelta

参数名称	描述
rescale_gradient	梯度的缩放因子。 可选 有效值：浮点型（最小值：1e-3，最大值：1.0） 默认值：1.0
sub_sample	每个纪元中要采样用于训练的训练数据的一部分。 可选 有效值：浮点型（最小值：0.0，最大值：1.0） 默认值：1.0
tolerance	损失函数的最大相对变化。当损失函数的变化在最近 num_patience_epochs 个纪元内低于此值时，将触发提前停止。 可选 有效值：浮点型（最小值：1e-6，最大值：0.1） 默认值：0.001
weight_decay	权重衰减系数。添加 L2 正则化。 可选 有效值：浮点型（最小值：0.0，最大值：1.0） 默认值：0.0

优化 NTM 模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

Amazon SageMaker NTM 是一种自主型学习算法，它学习大量离散数据集合 (如文档语料库) 的潜在表示。潜在表示使用未直接测量的推理变量来对数据集中的观察值进行建模。NTM 上的自动模型优化可帮助您找到最小化训练或验证数据损失的模型。训练损失 用于衡量模型适合训练数据的程度。验证损失 用于衡量模型可推广到未经过训练的数据的程度。低训练损失表明模型非常适合训练数据。低验证损失表明模型没有过拟合训练数据，因此应该能够成功对尚未训练的文档进行建模。通常，两个损失最好都很小。但是，将训练损失降到最低可能会导致过拟合并增加验证损失，这会降低模型的通用性。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

NTM 算法计算的指标

NTM 算法报告在训练期间计算的单个指标：validation:total_loss. 总损失是重建损失和 Kullback-Leibler 分歧. 优化超参数值时，请选择此指标作为目标。

指标名称	描述	优化方向
validation:total_los	验证集上的总损失	最小化

可优化 NTM 超参数

您可以优化 NTM 算法的以下超参数。通常设置低 `mini_batch_size` 和小 `learning_rate` 值会导致较低的验证损失，尽管它可能需要更长的时间来训练。低验证损失不一定产生人类解释的更连贯的主题。其他超级参数对训练和验证损失的影响可能因数据集而异。要了解哪些值是兼容的，请参阅NTM 超参数 (p. 1673)。

参数名称	参数类型	建议的范围
`encoder_layers_activation`	CategoricalParameterRanges	['sigmoid'、'tanh'、'relu']
`learning_rate`	ContinuousParameterRange	最 MinValue：1e-4，MaxValue：0.1
`mini_batch_size`	IntegerParameterRanges	MinValue：16，最大值：2048
`optimizer`	CategoricalParameterRanges	['sgd'、'adam'、'adadelta']
`rescale_gradient`	ContinuousParameterRange	MinValue：0.1，MaxValue：1.0
`weight_decay`	ContinuousParameterRange	MinValue：0.0，MaxValue：1.0

NTM 响应格式

所有 Amazon SageMaker 内置算法遵循中所述的常见输入推理格式。常见数据格式-推理. 本主题包含 SageMaker NTM 算法的可用输出格式列表。

JSON 响应格式

```
{
    "predictions":    [
        {"topic_weights": [0.02, 0.1, 0,...]},
        {"topic_weights": [0.25, 0.067, 0,...]}
    ]
}
```

JSONLINES 响应格式

```
{"topic_weights": [0.02, 0.1, 0,...]}
{"topic_weights": [0.25, 0.067, 0,...]}
```

RECORDIO 响应格式

```
[
    Record = {
        features = {},
        label = {
            'topic_weights': {
                keys: [],
                values: [0.25, 0.067, 0, ...]  # float32
            }
        }
    },
    Record = {
```

```
        features = {},
        label = {
            'topic_weights': {
                keys: [],
                values: [0.25, 0.067, 0, ...]  # float32
            }
        }
    }
]
```

Object2Vec 算法

Amazon SageMaker Object2Vec 算法是高度可自定义的通用神经嵌入算法。它可以学习高维对象的低维密集嵌入。将以某种方式学习嵌入，以便在嵌入空间中保留原始空间中的对象对之间的关系语义。例如，您可以使用学到的嵌入有效地计算对象的最近邻点，以及可视化低维空间中的相关对象的自然聚类。您还可以将嵌入用作下游指导式任务中相应对象的功能，例如分类或回归。

Object2Vec 概括了众所周知的 Word2Vec 嵌入技术，用于 SageMaker 中优化的单词。BlazingText 算法 (p. 1589). 有关论述如何将 Object2Vec 应用到一些实际使用案例的博客文章，请参阅Amazon SageMaker Object2VEC 简介.

主题

- Object2Vec 算法的 I/O 接口 (p. 1677)
- Object2Vec 算法的 EC2 实例建议 (p. 1678)
- Object2Vec 示例笔记本 (p. 1678)
- Object2Vec 工作原理 (p. 1678)
- Object2Vec 超参数 (p. 1680)
- 优化 Object2Vec 模型 (p. 1687)
- 用于 Object2Vec 训练的数据格式 (p. 1689)
- 用于 Object2Vec 推理的数据格式 (p. 1689)
- Object2Vec 的编码器嵌入 (p. 1690)

Object2Vec 算法的 I/O 接口

您可以在很多输入数据类型上使用 Object2Vec，包括以下示例。

输入数据类型	示例
语句-语句对	“一款有多个男性玩的足球比赛。” “有些男人在玩运动。”
标签-序列对	电影 “泰坦尼克号” 的流派标签，例如 “浪漫” 和 “戏剧” 以及简短描述：“詹姆斯·卡梅隆的泰坦尼克号是一个史诗般的，动感十足的浪漫，背对 R.M.S. Titanic. She was the most luxurious liner of her era, a ship of dreams, which ultimately carried over 1,500 people to their death in the ice cold waters of the North Atlantic in the early hours of April 15, 1912.”
客户-客户对	Jane 的客户 ID 和 Jackie 的客户 ID。
产品-产品对	足球的产品 ID 和篮球的产品 ID。
项审查用户-项对	用户的 ID 以及她购买的商品，例如苹果、梨和橙子。

要将输入数据转换为支持的格式，必须对其进行预处理。目前，Object2Vec 本身支持两种类型的输入：

- 离散标记，它表示为单个 integer-id 的列表。例如：[10]。
- 离散标记序列，它表示为 integer-ids 列表。例如：[0,12,10,13]。

每个对中的对象可以是不对称的。例如，这些对可以是 (标记, 序列)、(标记, 标记) 或 (序列, 序列)。对于标记输入，该算法支持简单嵌入作为兼容编码器。对于标记向量序列，该算法支持以下编码器：

- 平均池化嵌入
- 分层卷积神经网络 (CNN)
- 多层双向长短期记忆 (BiLSTM)

每个对的输入标签可以是以下内容之一：

- 分类标签，表示对中的对象之间的关系
- 分数，表示两个对象之间的相似性强度

对于分类中使用的分类标签，该算法支持交叉熵损失函数。对于回归中使用的基于评级/分数的标签，该算法支持均方误差 (MSE) 损失函数。在创建模型训练作业时，请使用 output_layer 超参数指定这些损失函数。

Object2Vec 算法的 EC2 实例建议

您使用的 Amazon Elastic Compute Cloud (Amazon EC2) 实例的类型取决于是训练还是运行推理。

用于训练的实例建议

在 CPU 上使用 Object2Vec 算法训练模型时，请从 ml.m5.2xlarge 实例开始。要在 GPU 上进行训练，请从 ml.p2.xlarge 实例开始。如果该实例上的训练时间太长，您可以使用更大的实例，例如 ml.m5.4xlarge 或 ml.m5。12xlarge实例目前，Object2Vec 算法只能在单个机器上训练。但是，它提供对多个 GPU 的支持。

用于推理的实例建议

对于使用经过训练的 Object2Vec 模型（具有深层神经网络）的推理，我们建议使用 ml.p3.2xlarge GPU 实例。由于 GPU 内存稀缺，可以指定 INFERENCE_PREFERRED_MODE 环境变量来优化是将 the section called “GPU 优化：GPU 优化 分类或回归” (p. 1689)还是将 the section called “GPU 优化：GPU 优化 嵌入编码器” (p. 1690)推理网络加载到 GPU 中。

Object2Vec 示例笔记本

- 使用 Object2VEC 将句子编码为固定长度嵌入
- 使用 Object2VEC 学习文档嵌入

> **Note**
>
> 要在笔记本实例上运行笔记本，请参阅示例笔记本 (p. 158). 要在 Studio 上运行笔记本电脑，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86).

Object2Vec 工作原理

在使用 Amazon SageMaker Object2Vec 算法时，请需要采用标准工作流程：处理数据，训练模型以及生成推理。

主题

- 第 1 步：处理数据 (p. 1679)
- 第 2 步：训练模型 (p. 1679)
- 第 3 步：生成推理 (p. 1679)

第 1 步：处理数据

在预处理过程中，将数据转换为JSON 行在中指定的文本文件格式用于 Object2Vec 训练的数据格式 (p. 1689). 要在训练期间获得最高的准确性，还需要在为模型提供数据之前将数据随机排序。如何生成随机排序取决于语言。对于 python，您可以使用 `np.random.shuffle`；对于 Unix，您可以使用 `shuf`。

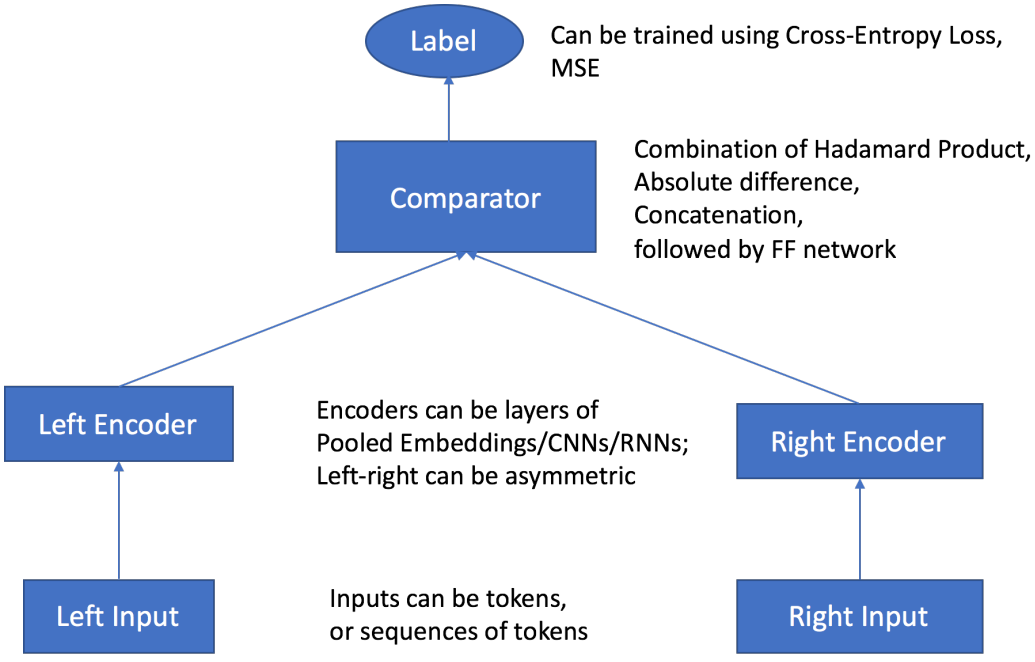
第 2 步：训练模型

SageMaker Object2Vec 算法具有以下主要组件：

- 两个输入通道-输入通道将一对相同或不同类型的对象作为输入，并将它们传递给独立的可自定义编码器。
- 两个编码器— enc0 和 enc1 两个编码器将每个对象转换为固定长度的嵌入向量。然后，将对中的对象的编码嵌入传递到比较器中。
- 比较器— 比较器以不同方式比较嵌入，并输出分数以表示配对对象之间的关系强度。例如，在句子对的输出分数中，1 表示句子对之间的关系较强，0 表示关系较弱。

在训练期间，该算法接受成对的对象及其关系标签或分数以作为输入。每个对中的对象可以具有不同的类型，如上所述。如果两个编码器的输入由相同的标记级单元组成，您可以在创建训练作业时将 `tied_token_embedding_weight` 超参数设置为 `True` 以使用共享标记嵌入层。例如，在比较两个具有单词标记级单元的句子时，可以执行该操作。要以指定的速率生成负样本，请将 `negative_sampling_rate` 超参数设置为所需的负样本/正样本比率。该超参数加快学习如何区分在训练数据中观察到的正样本和不太可能观察到的负样本。

对象对是通过独立的可自定义编码器传递的，这些编码器与相应对象的输入类型兼容。这些编码器将一个对中的每个对象转换为长度相等的固定长度嵌入向量。向量对传递到比较器运算符，该运算符使用 `comparator_list` 超参数中指定的值将向量合并为单个向量。然后，合并的向量通过多层感知机 (MLP) 层进行传递，该层生成输出，损失函数将输出与您提供的标签进行比较。这种比较评估对中的对象之间的关系强度，如模型中所预测的一样。下图显示了该工作流程。



从数据输入到分数的 Object2Vec 算法架构

第 3 步：生成推理

训练模型后，您可以使用训练的编码器预处理输入对象或执行两种类型的推理：

- 使用相应的编码器将单个输入对象转换为固定长度的嵌入
- 预测一对输入对象之间的关系标签或分数

推理服务器根据输入数据自动确定请求哪种类型。要将嵌入作为输出，请仅提供一个输入。要预测关系标签或分数，请在该对中同时提供两个输入。

Object2Vec 超参数

在 `CreateTrainingJob` 请求中，您可以指定训练算法。您还可以指定算法特定的超参数作为字符串到字符串映射。下表列出了 Object2Vec 训练算法的超参数。

参数名称	描述
`enc0_max_seq_len`	enc0 编码器的最大序列长度。 必填 有效值：1 ≤ 整数 ≤ 5000
`enc0_vocab_size`	enc0 标记的词汇表大小。 必填 有效值：2 ≤ 整数 ≤3000000
`bucket_width`	启用分桶时数据序列长度之间允许的差异。要启用分桶，请为该参数指定非零值。 可选 有效值：0 ≤ 整数 ≤ 100 默认值：0（不分桶）
`comparator_list`	用于自定义比较两个嵌入的方式的列表。Object2Vec 比较器运算符层将来自两个编码器的编码作为输入，并输出单个向量。该向量是将子向量串联在一起的结果。传递到 `comparator_list` 的字符串值及其传递顺序决定了如何合并这些子向量。例如，如果 `comparator_list="hadamard, concat"`，则比较器运算符串联两个编码的 Hadamard 乘积和两个编码的串联以构造向量。另一方面，如果 `comparator_list="hadamard"`，则比较器运算符将向量构造为仅两个编码的 Hadamard 乘积。 可选 有效值：包含三个二元运算符名称的任意组合的字符串：`hadamard`、`concat`，或者`abs_diff`. Object2Vec 算法目前要求两个向量编码具有相同的维度。这些运算符生成子向量，如下所示： • `hadamard`：将向量构造为两个编码的 Hadamard 乘积（元素积）。 • `concat`：将向量构造为两个编码的串联。 • `abs_diff`：将向量构造为两个编码之间的绝对差值。 默认值：`"hadamard, concat, abs_diff"`

参数名称	描述
dropout	网络层的丢弃概率。丢弃法 是神经网络中使用的一种正规化形式，通过修剪相关的神经元来减少过拟合。 可选 有效值：0.0 ≤浮动≤ 1.0 默认值：0.0
early_stopping_patience	在应用提前停止之前，允许的无改进的连续纪元数。改进是使用 early_stopping_tolerance 超参数定义的。 可选 有效值：1 ≤ 整数 ≤ 5 默认值：3
early_stopping_tolerance	算法必须在连续纪元之间实现损失函数减少，以避免在 early_stopping_patience 超参数中指定的连续纪元数量结束之后提前停止。 可选 有效值：0.000001 ≤ 浮点值 ≤ 0.1 默认值：0.01
enc_dim	嵌入层的输出的维。 可选 有效值：4 ≤ 整数 10000 默认值：4096
enc0_network	enc0 编码器的网络模型。 可选 有效值：hcnn、bilstm 或 pooled_embedding • hcnn：分层卷积神经网络。 • bilstm：双向长短期记忆网络 (biLSTM)，其中信号向后面和前面的时间点传播。这是适合用于顺序学习任务的循环神经网络 (RNN) 架构。 • pooled_embedding：对输入中的所有标记的嵌入取平均值。 默认值：hcnn
enc0_cnn_filter_width	卷积神经网络 (CNN) enc0 编码器的滤波器宽度。 条件 有效值：1 ≤ 整数 ≤ 9 默认值：3

参数名称	描述
enc0_freeze_pretrained_embedding	是否冻结 enc0 预训练嵌入权重。 条件 有效值：True 或 False 默认值：True
enc0_layers	enc0 编码器中的层数。 条件 有效值：auto或者 1 ≤ 整数 ≤ 4 • 对于 hcnn，auto 表示 4。 • 对于 bilstm，auto 表示 1。 • 对于 pooled_embedding，auto 忽略层数。 默认值：auto
enc0_pretrained_embedding_file	辅助数据通道中的预训练 enc0 标记嵌入文件的文件名。 条件 有效值：包含字母数字字符、下划线或句点的字符串。[A-Z0-9\.\ _] 默认值：""（空字符串）
enc0_token_embedding_dim	enc0 标记嵌入层的输出维。 条件 有效值：2 ≤ 整数 ≤ 1000 默认值：300
enc0_vocab_file	用于将预训练的 enc0 标记嵌入向量映射到数字词汇表 ID 的词汇表文件。 条件 有效值：包含字母数字字符、下划线或句点的字符串。[A-Z0-9\.\ _] 默认值：""（空字符串）

参数名称	描述
enc1_network	enc1 编码器的网络模型。如果您希望 enc1 编码器使用与 enc0 相同的网络模型（包括超参数值），请将该值设置为enc0. Note 即使 enc0 和 enc1 编码器网络具有对称的架构，您也无法在这些网络中共享参数值。 可选 有效值：enc0、hcnn、bilstm 或 pooled_embedding • enc0：enc0 编码器的网络模型。 • hcnn：分层卷积神经网络。 • bilstm：双向 LSTM，其中信号向后面和前面的时间点传播。这是适合用于顺序学习任务的循环神经网络 (RNN) 架构。 • pooled_embedding：输入中的所有标记的嵌入的平均值。 默认值：enc0
enc1_cnn_filter_width	CNN enc1 编码器的滤波器宽度。 条件 有效值：1 ≤ 整数 ≤ 9 默认值：3
enc1_freeze_pretrained_embedding	是否冻结 enc1 预训练嵌入权重。 条件 有效值：True 或 False 默认值：True
enc1_layers	enc1 编码器中的层数。 条件 有效值：auto或者 1 ≤ 整数 ≤ 4 • 对于 hcnn，auto 表示 4。 • 对于 bilstm，auto 表示 1。 • 对于 pooled_embedding，auto 忽略层数。 默认值：auto
enc1_max_seq_len	enc1 编码器的最大序列长度。 条件 有效值：1 ≤ 整数 ≤ 5000

参数名称	描述
enc1_pretrained_embedding_file	辅助数据通道中的 enc1 预训练标记嵌入文件的名称。 条件 有效值：包含字母数字字符、下划线或句点的字符串。[A-Z0-9\.\ _] 默认值：""（空字符串）
enc1_token_embedding_dim	enc1 标记嵌入层的输出维度。 条件 有效值：2 ≤ 整数 ≤ 1000 默认值：300
enc1_vocab_file	用于将预训练的 enc1 标记嵌入映射到词汇表 ID 的词汇表文件。 条件 有效值：包含字母数字字符、下划线或句点的字符串。[A-Z0-9\.\ _] 默认值：""（空字符串）
enc1_vocab_size	enc0 标记的词汇表大小。 条件 有效值：2 ≤ 整数 ≤3000000
epochs	要运行用于训练的纪元数。 可选 有效值：1 ≤ 整数 ≤ 100 默认值：30
learning_rate	训练的学习率。 可选 有效值：1.0E-6 ≤ 浮点值 ≤ 1.0 默认值：0.0004
mini_batch_size	在训练期间将 optimizer 的数据集拆分成的批次大小。 可选 有效值：1 ≤ 整数 10000 默认值：32

参数名称	描述
mlp_activation	多层感知机 (MLP) 层的激活函数的类型。 可选 有效值：tanh、relu 或 linear • tanh：双曲正切 • relu：修正线性单元 (ReLU) • linear：线性函数 默认值：linear
mlp_dim	MLP 层的输出维度。 可选 有效值：2 ≤ 整数 10000 默认值：512
mlp_layers	网络中的 MLP 层数。 可选 有效值：0 ≤ 整数 ≤ 10 默认值：2
negative_sampling_rate	为帮助训练算法而生成的负样本与用户提供的正样本的比率。负样本表示实际不太可能出现的数据，并在训练时标记为不正确的样本。它们有助于训练模型，以区分观察到的正样本和未观察到的负样本。要指定用于训练的负样本与正样本的比率，请将该值设置为正整数。例如，如果在所有样本都是正样本的输入数据上训练算法，并将 negative_sampling_rate 设置为 2，则 Object2Vec 算法在内部为每个正样本生成两个负样本。如果您不希望在训练期间生成或使用负样本，请将该值设置为 0。 可选 有效值：0 ≤ 整数 默认值：0 (关闭)
num_classes	用于分类训练的分类数。Amazon SageMaker 忽略该超参数以解决回归问题。 可选 有效值：2 ≤ 整数 ≤ 30 默认值：2

参数名称	描述
optimizer	优化程序类型。 可选 有效值：adadelta、adagrad、adam、sgd，或者rmsprop. • adadelta：梯度下降的每个维的学习率方法 • adagrad：自适应梯度算法 • adam：自适应矩估计算法 • sgd：随机梯度下降 • rmsprop： 均方根传播 默认值：adam
output_layer	输出层的类型，您可以在其中指定任务是回归还是分类。 可选 有效值：softmax 或 mean_squared_error • softmax：Softmax 函数用于分类。 • mean_squared_error：MSE 用于回归。 默认值：softmax
tied_token_embedding_weight	是否将共享嵌入层用于两个编码器。如果两个编码器的输入使用相同的标记级单元，请使用共享标记嵌入层。例如，对于文档集合，如果一个编码器对句子进行编码，另一个编码器对整个文档进行编码，则可以使用共享标记嵌入层。这是因为，句子和文档都是由同一词汇表中的单词标记组成的。 可选 有效值：True 或 False 默认值：False
token_embedding_storage_type	在训练期间使用的梯度更新模式：在使用 dense 模式时，优化程序计算标记嵌入层的完整梯度矩阵，即使梯度的大多数行的值为零。在使用 sparse 模式时，优化程序仅存储实际在小批次中使用的梯度行。如果您希望算法执行延迟梯度更新（仅计算非零行中的梯度并加速训练），请指定 row_sparse。如果将该值设置为 row_sparse，则会限制其他超参数的可用值，如下所示： • optimizer 超参数必须设置为 adam、adagrad 或 sgd。否则，该算法将引发 CustomerValueError。 • 该算法自动禁用分桶，从而将 bucket_width 超参数设置为 0。 可选 有效值：dense 或 row_sparse 默认值：dense

参数名称	描述
weight_decay	用于优化的权重衰减参数。 可选 有效值：0 ≤ 浮点值 ≤10000 默认值：0（无衰减）

优化 Object2Vec 模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。对于目标指标，您可以使用该算法计算的指标之一。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

Object2Vec 算法计算的指标

该 Object2Vec 算法同时具有分类和回归指标。output_layer 类型确定可用于自动模型优化的指标。

Object2Vec 算法计算的回归量指标

该算法报告均方误差回归量指标，该指标在测试和验证期间计算。在为回归任务优化模型时，请选择此指标作为目标。

指标名称	描述	优化方向
test:mean_squared_error	均方错误差	最小化
validation:mean_squared_error	均方错误差	最小化

Object2Vec 算法计算的分类指标

Object2Vec 算法报告在测试和验证期间计算的准确度和交叉熵分类指标。在为分类任务优化模型时，请选择其中之一作为目标。

指标名称	说明	优化方向
test:accuracy	Accuracy	最大化
test:cross_entropy	交叉熵	最小化
validation:accuracy	Accuracy	最大化
validation:cross_entropy	交叉熵	最小化

可优化 Object2Vec 超参数

您可为 Object2Vec 算法优化以下超参数。

超参数名称	超参数类型	建议的范围和值	
dropout	ContinuousParameterRange	MinValue： 0.0，MaxValue：1.0	

超参数名称	超参数类型	建议的范围和值	
early_stopping_patience	IntegerParameterRange	MinValue：1，MaxValue：5	
early_stopping_tolerance	ContinuousParameterRange	MinValue：0.001，MaxValue：0.1	
enc_dim	IntegerParameterRange	MinValue：4，MaxValue：4096	
enc0_cnn_filter_width	IntegerParameterRange	MinValue：1，MaxValue：5	
enc0_layers	IntegerParameterRange	MinValue：1，MaxValue：4	
enc0_token_embedding_dim	IntegerParameterRange	MinValue：5，MaxValue：300	
enc1_cnn_filter_width	IntegerParameterRange	MinValue：1，MaxValue：5	
enc1_layers	IntegerParameterRange	MinValue：1，MaxValue：4	
enc1_token_embedding_dim	IntegerParameterRange	MinValue：5，MaxValue：300	
epochs	IntegerParameterRange	MinValue：4，MaxValue：20	
learning_rate	ContinuousParameterRange	最 MinValue：1e-6，最大值：最大值：1e-6 1.0	
mini_batch_size	IntegerParameterRange	MinValue：1，MaxValue：8192	
mlp_activation	CategoricalParameterRanges	[tanh, relu, linear]	
mlp_dim	IntegerParameterRange	MinValue：16，MaxValue：1024	
mlp_layers	IntegerParameterRange	MinValue：1，MaxValue：4	
optimizer	CategoricalParameterRanges	[adagrad, adam, rmsprop, sgd, adadelta]	
weight_decay	ContinuousParameterRange	MinValue：0.0，MaxValue：1.0	

用于 Object2Vec 训练的数据格式

输入: JSON 行请求格式

C内容-类型：应用程序/jsonlines

```
{"label": 0, "in0": [6, 17, 606, 19, 53, 67, 52, 12, 5, 10, 15, 10178, 7, 33, 652, 80, 15,
 69, 821, 4], "in1": [16, 21, 13, 45, 14, 9, 80, 59, 164, 4]}
{"label": 1, "in0": [22, 1016, 32, 13, 25, 11, 5, 64, 573, 45, 5, 80, 15, 67, 21, 7, 9,
 107, 4], "in1": [22, 32, 13, 25, 1016, 573, 3252, 4]}
{"label": 1, "in0": [774, 14, 21, 206], "in1": [21, 366, 125]}
```

“in0”和“in1”分别是 encoder0 和 encoder1 的输入。相同的格式对分类和回归问题都有效。对于回归，字段`"label"`可以接受实际价值的投入.

用于 Object2Vec 推理的数据格式

GPU 优化：GPU 优化 分类或回归

由于 GPU 内存稀缺，`INFERENCE_PREFERRED_MODE`可以指定环境变量来优化是分类/回归还是这 the section called “输出：嵌入编码器” (p. 1691)推理网络已加载到 GPU 中。如果您的大多数推理适用于分类或回归，请指定 `INFERENCE_PREFERRED_MODE=classification`。下面是一个批量转换示例，用于说明如何使用 4 个针对分类/回归推理进行优化的 p3.2xlarge 实例：

```
transformer = o2v.transformer(instance_count=4,
                              instance_type="ml.p2.xlarge",
                              max_concurrent_transforms=2,
                              max_payload=1,  # 1MB
                              strategy='MultiRecord',
                              env={'INFERENCE_PREFERRED_MODE': 'classification'},  # only
 useful with GPU
                              output_path=output_s3_path)
```

输入: 分类或回归请求格式

Content-type：application/json

```
{
  "instances" : [
    {"in0": [6, 17, 606, 19, 53, 67, 52, 12, 5, 10, 15, 10178, 7, 33, 652, 80, 15, 69, 821,
 4], "in1": [16, 21, 13, 45, 14, 9, 80, 59, 164, 4]},
    {"in0": [22, 1016, 32, 13, 25, 11, 5, 64, 573, 45, 5, 80, 15, 67, 21, 7, 9, 107, 4],
 "in1": [22, 32, 13, 25, 1016, 573, 3252, 4]},
    {"in0": [774, 14, 21, 206], "in1": [21, 366, 125]}
  ]
}
```

Content-type：application/jsonlines

```
{"in0": [6, 17, 606, 19, 53, 67, 52, 12, 5, 10, 15, 10178, 7, 33, 652, 80, 15, 69, 821, 4],
 "in1": [16, 21, 13, 45, 14, 9, 80, 59, 164, 4]}
{"in0": [22, 1016, 32, 13, 25, 11, 5, 64, 573, 45, 5, 80, 15, 67, 21, 7, 9, 107, 4], "in1":
 [22, 32, 13, 25, 1016, 573, 3252, 4]}
{"in0": [774, 14, 21, 206], "in1": [21, 366, 125]}
```

对于分类问题，分数向量的长度对应于 `num_classes`。对于回归问题，长度为 1。

输出：分类或回归响应格式

Accept: application/json

```
{
    "predictions": [
        {
            "scores": [
                0.6533935070037842,
                0.07582679390907288,
                0.2707797586917877
            ]
        },
        {
            "scores": [
                0.026291321963071823,
                0.6577019095420837,
                0.31600672006607056
            ]
        }
    ]
}
```

Accept: application/jsonlines

```
{"scores":[0.195667684078216,0.395351558923721,0.408980727195739]}
{"scores":[0.251988261938095,0.258233487606048,0.489778339862823]}
{"scores":[0.280087798833847,0.368331134319305,0.351581096649169]}
```

在分类和回归格式中，分数应用于单个标签。

Object2Vec 的编码器嵌入

GPU 优化：GPU 优化 嵌入编码器

嵌入是从离散对象（例如单词）到实数向量的映射。

由于 GPU 内存稀缺，可以指定 `INFERENCE_PREFERRED_MODE` 环境变量来优化是将the section called “推理格式：得分” (p. 1689)还是将编码器嵌入推理网络加载到 GPU 中。如果您的绝大多数推理用于进行编码器嵌入，则指定 `INFERENCE_PREFERRED_MODE=embedding`。下面是一个批量转换示例，用于说明如何使用 4 个针对编码器嵌入推理进行优化的 p3.2xlarge 实例：

```
transformer = o2v.transformer(instance_count=4,
                              instance_type="ml.p2.xlarge",
                              max_concurrent_transforms=2,
                              max_payload=1,  # 1MB
                              strategy='MultiRecord',
                              env={'INFERENCE_PREFERRED_MODE': 'embedding'},  # only useful
 with GPU
                              output_path=output_s3_path)
```

输入：嵌入编码器

Content-type: application/json; infer_max_seqlens=<FWD-LENGTH>,<BCK-LENGTH>

其中 <FWD-LENGTH> 和 <BCK-LENGTH> 是 [1,5000] 范围内的整数，并定义向前和向后编码器的最大序列长度。

```
{
  "instances" : [
    {"in0": [6, 17, 606, 19, 53, 67, 52, 12, 5, 10, 15, 10178, 7, 33, 652, 80, 15, 69, 821,
 4]},
    {"in0": [22, 1016, 32, 13, 25, 11, 5, 64, 573, 45, 5, 80, 15, 67, 21, 7, 9, 107, 4]},
```

```
    {"in0": [774, 14, 21, 206]}
  ]
}
```

Content-type: application/jsonlines; infer_max_seqlens=<FWD-LENGTH>,<BCK-LENGTH>

其中 <FWD-LENGTH> 和 <BCK-LENGTH> 是 [1,5000] 范围内的整数，并定义向前和向后编码器的最大序列长度。

```
{"in0": [6, 17, 606, 19, 53, 67, 52, 12, 5, 10, 15, 10178, 7, 33, 652, 80, 15, 69, 821, 4]}
{"in0": [22, 1016, 32, 13, 25, 11, 5, 64, 573, 45, 5, 80, 15, 67, 21, 7, 9, 107, 4]}
{"in0": [774, 14, 21, 206]}
```

在这两种格式中，您可以仅指定一种输入类型：“in0” 或 “in1.”。然后，推理服务调用相应的编码器，并输出每个实例的嵌入。

输出：嵌入编码器

Content-type：application/json

```
{
  "predictions": [
    {"embeddings":
[0.057368703186511,0.030703511089086,0.099890425801277,0.063688032329082,0.026327300816774,0.0036375711
    {"embeddings":
[0.150190666317939,0.05145975202322,0.098204270005226,0.064249359071254,0.056249320507049,0.01513972133
  ]
}
```

Content-type：application/jsonlines

```
{"embeddings":
[0.057368703186511,0.030703511089086,0.099890425801277,0.063688032329082,0.026327300816774,0.0036375711
{"embeddings":
[0.150190666317939,0.05145975202322,0.098204270005226,0.064249359071254,0.056249320507049,0.01513972133
```

推理服务输出的嵌入的向量长度等于在训练时指定的以下超参数之一的值：enc0_token_embedding_dim、enc1_token_embedding_dim 或 enc_dim。

对象检测算法

AmazonSageMaker对象检测算法使用单个深度神经网络检测和分类图像中的对象。它是一种指导式学习算法，将图像作为输入并识别图像场景中的所有对象实例。该对象被归类到指定集合中的一个分类，具有属于该分类的置信度分数。它在图像中的位置和比例由矩形边界框表示。它使用单拍多箱探测器 (SSD)框架并支持两个基本网络：VGG和ResNet. 可以从头开始训练网络，也可以使用在ImageNet数据集。

主题

- 对象检测算法的输入/输出接口 (p. 1692)
- 对象检测算法的 EC2 实例建议 (p. 1694)
- 对象检测示例笔记本 (p. 1694)
- 对象检测的工作原理 (p. 1694)
- 对象检测超参数 (p. 1694)
- 优化对象检测模型 (p. 1698)
- 对象检测请求和响应格式 (p. 1699)

对象检测算法的输入/输出接口

在文件模式下，SageMaker 对象检测算法同时支持 RecordIO (`application/x-recordio`) 和图像（`image/png`、`image/jpeg` 和 `application/x-image`）内容类型以进行训练；在管道模式下，该算法支持 RecordIO `application/x-recordio` 内容类型以进行训练。但是，您还可以使用图像文件（`image/png`、`image/jpeg` 和 `application/x-image`）在管道模式下进行训练，无需使用扩增清单格式创建 RecordIO 文件。对于 Amazon，建议的输入格式SageMaker对象检测算法是Apache MXNet RecordIO. 但是，您也可以使用 .jpg 或 .png 格式的原始图像。该算法仅支持 `application/x-image` 用于推理。

> **Note**
>
> 为了与现有深度学习框架保持更好的互操作性，这不同于其他 Amazon 常用的 protobuf 数据格式。SageMaker算法。

有关数据格式的更多详细信息，请参阅对象检测示例笔记本 (p. 1694)。

使用 RecordIO 格式进行训练

如果您使用 RecordIO 格式进行训练，则同时指定 train 和 validation通道作为 `CreateTrainingJob` 请求的 `InputDataConfig` 参数值。在 train 通道中指定一个 RecordIO (.rec) 文件，在 validation 通道中指定一个 RecordIO 文件。将两个通道的内容类型设置为 `application/x-recordio`。一个示例，介绍如何生成可在对象检测示例笔记本中找到的 RecordIO 文件。您也可以使用中的工具MXNet 的 GluonCV为常见的数据集生成 RecordIO 文件，例如PASCAL 可视对象类和上下文中的常见对象 (COCO).

使用图像格式进行训练

如果您使用图像格式进行训练，则指定 `train`、`validation`、`train_annotation` 和 `validation_annotation` 通道作为 `CreateTrainingJob` 请求的 `InputDataConfig` 参数值。为 train 或 validation 通道指定相应的图像数据（.jpg 或 .png）文件。对于注释数据，您可以使用 JSON 格式。在 `train_annotation` 和 `validation_annotation` 通道中指定相应的 .json 文件。根据图像类型，将所有四个通道的内容类型设置为 `image/png` 或 `image/jpeg`。当数据集同时包含 .jpg 和 .png 图像时，您还可以使用内容类型 `application/x-image`。以下是 .json 文件的示例。

```
{
   "file": "your_image_directory/sample_image1.jpg",
   "image_size": [
      {
         "width": 500,
         "height": 400,
         "depth": 3
      }
   ],
   "annotations": [
      {
         "class_id": 0,
         "left": 111,
         "top": 134,
         "width": 61,
         "height": 128
      },
      {
         "class_id": 0,
         "left": 161,
         "top": 250,
         "width": 79,
         "height": 143
      },
      {
         "class_id": 1,
         "left": 101,
         "top": 185,
         "width": 42,
```

```
            "height": 130
        }
    ],
    "categories": [
        {
            "class_id": 0,
            "name": "dog"
        },
        {
            "class_id": 1,
            "name": "cat"
        }
    ]
}
```

每个图像都需要一个 .json 文件进行注释，而 .json 文件应该与相应的图像具有相同的名称。上面的 .json 文件的名称应该是“sample_image1.json”。注释 .json 文件中有四个属性。属性“file”指定图像文件的相对路径。例如，如果您的训练图像和相应的 .json 文件存储在 s3://*your_bucket*/train/sample_image 和 s3://*your_bucket*/train_annotation 中，请将您的训练和训练注释通道的路径分别指定为 s3://*your_bucket*/train 和 s3://*your_bucket*/train_annotation。

在 .json 文件中，名为 sample_image1.jpg 的图像的相对路径应为 sample_image / sample_image1.jpg。`"image_size"` 属性指定整个图像的尺寸。这些区域有：SageMaker对象检测算法目前只支持 3 通道图像。`"annotations"` 属性指定图像中对象的分类和边界框。每个对象都由一个 `"class_id"` 索引和四个边界框坐标（`"left"`、 `"top"`、`"width"` 和 `"height"`）注释。`"left"`（x 坐标）和 `"top"`（y 坐标）值表示边界框的左上角。`"width"`（x 坐标）和 `"height"`（y 坐标）值表示边界框的维度。原点 (0, 0) 是整个图像的左上角。如果您在一个图像内有多个对象，则所有注释都应包含在单个 .json 文件中。`"categories"` 属性存储类索引与类名称之间的映射。类索引应采用连续编号，并且编号应从 0 开始。`"categories"` 属性对于注释 .json 文件是可选的

使用扩增清单图像格式进行训练

扩增清单格式使您可以使用图像文件在管道模式下进行训练，而无需创建 RecordIO 文件。您需要将训练通道和验证通道指定为 `CreateTrainingJob` 请求的 `InputDataConfig` 参数的值。使用该格式时，需要生成包含图像列表及其相应注释的 S3 清单文件。清单文件格式应为 JSON 行格式，其中每行代表一个样本。使用指向图像 S3 位置的 `'source-ref'` 标签来指定图像。将在 `"AttributeNames"` 参数值下面提供注释，如 `CreateTrainingJob` 请求中指定。它还可以在 `metadata` 标签下包含其他元数据，但这些会被算法忽略。在以下示例中，`"AttributeNames` 包含在列表 `["source-ref", "bounding-box"]` 中：

```
{"source-ref": "s3://your_bucket/image1.jpg", "bounding-box":{"image_size":[{ "width":
 500, "height": 400, "depth":3}], "annotations":[{"class_id": 0, "left": 111, "top":
 134, "width": 61, "height": 128}, {"class_id": 5, "left": 161, "top": 250, "width": 80,
 "height": 50}]}, "bounding-box-metadata":{"class-map":{"0": "dog", "5": "horse"}, "type":
 "groundtruth/object-detection"}}
{"source-ref": "s3://your_bucket/image2.jpg", "bounding-box":{"image_size":[{ "width":
 400, "height": 300, "depth":3}], "annotations":[{"class_id": 1, "left": 100, "top": 120,
 "width": 43, "height": 78}]}, "bounding-box-metadata":{"class-map":{"1": "cat"}, "type":
 "groundtruth/object-detection"}}
```

在训练对象检测算法时，`"AttributeNames"` 在输入文件中的顺序很重要。它按特定的顺序接受管道数据，先接受 `image`，然后接受 `annotations`。因此，”AttributeNames“在这个例子中提供了`"source-ref"`首先，然后是`"bounding-box"`. 在将对象检测与增强清单一起使用时，`RecordWrapperType` 参数值必须设置为 `"RecordIO"`。

有关增强清单文件的更多信息，请参阅通过增强清单文件，将数据集元数据提供给训练作业 (p. 2090)。

增量训练

您还可以使用以前用 SageMaker 训练的模型中的构件作为种子来训练新模型。当您想训练具有相同或相似数据的新模型时，增量训练可节省训练时间。SageMaker对象检测模型只能使用在中训练的另一个内置对象检测模型植入数据。SageMaker.

要使用预训练模型，请在 `CreateTrainingJob` 请求中，在 `InputDataConfig` 参数中将 `ChannelName` 指定为“model”。将模型通道的 `ContentType` 设置为 `application/x-sagemaker-model`。您上传到模型通道的新模型和预训练模型的输入超参数必须与 `base_network` 和 `num_classes` 输入参数具有相同的设置。这些参数定义了网络架构。对于预训练模型文件，使用 SageMaker 输出的压缩模型构件（采用 .tar.gz 格式）。您可以对输入数据使用 RecordIO 或图像格式。

有关增量训练及其使用方式说明的更多信息，请参阅Amazon SageMaker 中的增量训练 (p. 2081)。

对象检测算法的 EC2 实例建议

对于对象检测，我们支持以下 GPU 实例用于训练：`ml.p2.xlarge`、`ml.p2.8xlarge`、`ml.p2.16xlarge`、`ml.p3.2xlarge`、`ml.p3.8xlarge` 和 `ml.p3.16xlarge`。对于大批量训练，建议使用具有更多内存的 GPU 实例。您也可以在多 GPU 和多机器设置上运行该算法以进行分布式训练。不过，CPU（例如 C5 和 M5）和 GPU（例如 P2 和 P3）实例均可用于推理。在上逐项列出了所有支持用于推理的实例类型。亚马逊SageMakerML 实例类型.

对象检测示例笔记本

对于演示如何使用SageMaker对象检测算法用于在

加州理工大学鸟类（古巴 200 2011）使用单拍多盒检测器算法的数据集，请参阅亚马逊SageMaker鸟类物种的物体检测. 有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147)。创建笔记本实例并将其打开后，选择SageMaker示例选项卡以查看所有SageMaker示例。使用对象检测算法的对象检测示例笔记本位于 Amazon 算法简介 部分。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

对象检测的工作原理

对象检测算法根据已知的对象类别集合来识别和定位图像中的对象的所有实例。该算法接受图像作为输入并输出对象所属的类别，以及它属于该类别的置信度分数。该算法还使用矩形边界框预测对象的位置和比例。亚马逊SageMaker对象检测使用单拍多箱探测器 (SSD)该算法采用卷积神经网络 (CNN) 作为基础网络用于分类任务预训练。SSD 使用中间层输出作为检测特征。

各种 CNN 例如VGG和ResNet在图像分类任务上都有很好的表现。Amazon 中的对象检测SageMaker同时支持 VGG-16 和ResNet-50 作为 SSD 的基础网络。该算法可以在完全训练模式或迁移学习模式中训练。在完全训练模式下，基础网络使用随机权重进行初始化，然后根据用户数据进行训练。在迁移学习模式下，基础网络和权重是从预训练模型加载的。

对象检测算法在内部动态使用标准数据扩增操作，例如翻转、重新调整和抖动，以帮助避免过度拟合。

对象检测超参数

在 `CreateTrainingJob` 请求中，您指定要使用的训练算法。您还可以指定特定于算法的超参数，用于帮助根据训练数据集估计模型的参数。下表列出了 Amazon 提供的超参数。SageMaker用于训练对象检测算法。有关对象训练工作原理的更多信息，请参阅对象检测的工作原理 (p. 1694)。

参数名称	说明
`num_classes`	输出类的数量。此参数定义网络输出的维度，通常设置为数据集中的类别数。 必填 有效值：正整数
`num_training_samples`	输入数据集中的训练样本数。

参数名称	说明
	Note 如果此值与训练设置中的样本数不匹配，则 `lr_scheduler_step` 参数的行为未定义，且分布式训练准确度可能会受影响。 必填 有效值：正整数
`base_network`	要使用的基础网络架构。 可选 有效值：“vgg-16”或“resnet-50” 默认值：“vgg-16”
`early_stopping`	`True` 表示在训练期间使用提前停止逻辑。`False` 表示不使用它。 可选 有效值：`True` 或 `False` 默认值：`False`
`early_stopping_min_epochs`	可以调用提前停止逻辑之前必须运行的最小纪元数。仅当 `early_stopping = True` 时使用它。 可选 有效值：正整数 默认值：10
`early_stopping_patience`	如果在相关指标中没有改进，则在结束训练前等待的纪元数（如 `early_stopping_tolerance` 中的定义）。仅当 `early_stopping = True` 时使用它。 可选 有效值：正整数 默认值：5
`early_stopping_tolerance`	`validation:mAP` 的相对改进的容差值，要求超过平均精度 (mAP) 以避免提前停止。如果 mAP 变化除以上一个最佳 mAP 的比率小于 `early_stopping_tolerance` 值集，则提早停止会认为没有任何改进。仅当 `early_stopping = True` 时使用它。 可选 有效值：0 ≤ 浮点型 ≤ 1 默认值：0.0

参数名称	说明
image_shape	输入图像的图像大小。我们将输入图像重新缩放为具有此大小的方形图像。我们建议使用 300 和 512 来提高性能。 可选 有效值：大于等于 300 的正整数 默认值：300
epochs	训练纪元数。 可选 有效值：正整数 默认值：30
freeze_layer_pattern	用于冻结基础网络中的层的正则表达式。例如，如果我们设置 freeze_layer_pattern = "^(conv1_|conv2_).*"，则名称中包含 "conv1_" 或 "conv2_" 的任何层都将冻结，这意味着这些层的权重在训练期间不会更新。图层名称可以在网络符号文件中找到。vgg16 符号 .json和resnet-50 符号 .json. 冻结层意味着其权重无法进一步修改。这可以显著减少训练时间，而不利的一面是准确性会有适度损失。该技术通常用于迁移学习，其中基础网络中的较低层不需要重新训练。 可选 有效值：字符串 默认值：没有层冻结。
kv_store	分布式训练期间使用的权重更新同步模式。可对各机器同步或异步更新权重。同步更新通常比异步更新提供更好的准确率，但速度较慢。请参阅 MXNet 教程中的分布式训练，以了解详细信息。 Note 此参数不适用于单个机器训练。 可选 有效值：'dist_sync' 或 'dist_async' • 'dist_sync'：每次批处理后，梯度会与所有工作线程同步。对于 'dist_sync'，批次大小现在表示每台机器上使用的批次大小。因此，如果有 n 台机器，并且我们使用批次大小 b，则 dist_sync 行为类似于单个机器且批次大小为 n*b。 • 'dist_async'：执行异步更新。每当从任何机器收到梯度时，权重就会更新，并且权重更新是原子更新。然而，该顺序并不能得到保证。 默认值：-

参数名称	说明
label_width	用于跨训练和验证数据同步的强制填充标签宽度。例如，如果数据中的一个图像包含最多 10 个对象，并且每个对象的注释都指定了 5 个数字 [class_id、left、top、width、height]，那么 label_width 应不小于（10 * 5 + 标头信息长度）。标头信息长度通常是 2。我们建议使用稍大的 label_width 进行训练，例如本示例中为 60。 可选 有效值：正整数足够大以容纳数据中最大的注释信息长度。 默认值：350
learning_rate	初始学习率。 可选 有效值：(0, 1] 中的浮点值 默认值：0.001
lr_scheduler_factor	用于减少学习率的比率。与定义为 lr_new 的 lr_scheduler_step 参数结合使用 = lr_old * lr_scheduler_factor。 可选 有效值：(0, 1) 中的浮点值 默认值：0.1
lr_scheduler_step	降低学习率的纪元。在逗号分隔的字符串中列出的纪元，按照 lr_scheduler_factor 减少学习率：“epoch1，epoch2...”。例如，如果将值设置为“10, 20”并且 lr_scheduler_factor 设置为 1/2，则学习率在第 10 纪元之后减半，然后在第 20 纪元之后再减半。 可选 有效值：字符串 默认值：空字符串
mini_batch_size	训练的批次大小。在单机器多 GPU 设置中，每个 GPU 处理 mini_batch_size/num_gpu 个训练样本。对于 dist_sync 模式下的多机器训练，实际批次大小为 mini_batch_size*机器数量。大的 mini_batch_size 值通常可加快训练速度，但它可能会导致内存不足问题。内存使用率与 mini_batch_size、image_shape 和 base_network 架构相关。例如，在单个 p3.2xlarge 实例上，不出现内存不足错误的最大 mini_batch_size 为 32，其中 base_network 设置为“resnet-50”并且 image_shape 为 300。使用相同的实例，您可以将 mini_batch_size 设置为 64，使用基础网络 vgg-16，并将 image_shape 设置为 300。 可选 有效值：正整数 默认值：32

参数名称	说明
momentum	sgd 的动量。对其他优化程序则忽略。 可选 有效值：(0, 1] 中的浮点值 默认值：0.9
nms_threshold	非最大抑制阈值。 可选 有效值：(0, 1] 中的浮点值 默认值：0.45
optimizer	优化程序类型。有关优化程序值的详细信息，请参阅 MXNet 的 API。 可选 有效值：['sgd', 'adam', 'rmsprop', 'adadelta'] 默认值：'sgd'
overlap_threshold	评估重叠阈值。 可选 有效值：(0, 1] 中的浮点值 默认值：0.5
use_pretrained_model	指明是否使用预先训练的模型进行训练。如果设置为 1，则加载具有架构的预先训练的模型并将其用于训练。否则，从头训练网络。 可选 有效值：0 或 1 默认值：1
weight_decay	sgd 和 rmsprop 的权重衰减系数。对其他优化程序则忽略。 可选 有效值：(0, 1) 中的浮点值 默认值：0.0005

优化对象检测模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

对象检测算法计算的指标

对象检测算法在训练期间报告单个指标：validation:mAP. 在优化模型时，选择此指标作为目标指标。

指标名称	说明	优化方向
validation:mAP	对验证集计算的平均精度 (mAP)。	最大化

可优化对象检测超参数

优化 AmazonSageMaker具有以下超参数的对象检测模型。对对象检测目标指标影响最大的超参数包括：mini_batch_size、learning_rate 和 optimizer。

参数名称	参数类型	建议的范围
learning_rate	ContinuousParameterRange	MinValue: 1e-6MaxValue：0.5
mini_batch_size	IntegerParameterRANGE	MinValue：8,MaxValue：64
momentum	ContinuousParameterRange	MinValue：0.0,MaxValue：0.999
optimizer	CategoricalParameterRANGE	['sgd'、'adam'、'rmsprop'、'adadelta']
weight_decay	ContinuousParameterRange	MinValue：0.0,MaxValue：0.999

对象检测请求和响应格式

请求格式

使用模型的终端节点查询经过训练的模型。终端节点接受具有 image/jpeg 和 image/png 内容类型的 .jpg 和 .png 图像格式。

响应格式

响应是具有置信度分数的分类索引以及以 JSON 格式编码的图像中的所有对象的边界框坐标。以下是响应 .json 文件的示例：

```
{"prediction":[
  [4.0, 0.86419455409049988, 0.3088374733924866, 0.07030484080314636, 0.7110607028007507,
 0.9345266819000244],
  [0.0, 0.73376623392105103, 0.5714187026023865, 0.40427327156066895, 0.827075183391571,
 0.9712159633636475],
  [4.0, 0.32643985450267792, 0.3677481412887573, 0.034883320331573486, 0.6318609714508057,
 0.5967587828636169],
  [8.0, 0.22552496790885925, 0.6152569651603699, 0.5722782611846924, 0.882301390171051,
 0.8985623121261597],
  [3.0, 0.42260299175977707, 0.019305512309074402, 0.08386176824569702,
 0.39093565940856934, 0.9574796557426453]
]}
```

此 .json 文件中的每一行都包含一个表示检测到的对象的数组。这些对象数组中的每一个都包含六个数字的列表。第一个数字是预测分类标签。第二个数字是检测的关联置信度分数。最后四个数字代表边界框坐标 [xmin, ymin, xmax, ymax]。这些输出边界框角索引由整体图像大小标准化。请注意，此编码与输入 .json 格

式使用的编码不同。例如，在检测结果的第一个条目中，0.3088374733924866 是边界框的左坐标（左上角的 x 坐标）与整体图像宽度的比例，0.07030484080314636 是顶部坐标（y 坐标）边界框的左上角与整体图像高度的比例，0.7110607028007507 是边界框的右坐标（右下角的 x 坐标）与整体图像宽度的比例，而 0.9345266819000244 是边界框的底部坐标（右下角的 y 坐标）与整体图像高度的比率。

为避免检测结果不可靠，您可能需要筛选掉具有低置信度分数的检测结果。在物体检测样本笔记本，我们提供了使用阈值来删除低置信度检测和在原始图像上绘制边界框的脚本示例。

对于批量转换，响应采用 JSON 格式，其格式与上述 JSON 格式相同。每个图像的检测结果表示为 JSON 文件。例如：

```
{"prediction": [[label_id, confidence_score, xmin, ymin, xmax, ymax], [label_id,
 confidence_score, xmin, ymin, xmax, ymax]]}
```

有关训练和推理的更多详细信息，请参阅对象检测示例笔记本 (p. 1694)。

：JSON 响应格式

accept: application/json;annotation=1

```
{
   "image_size": [
      {
         "width": 500,
         "height": 400,
         "depth": 3
      }
   ],
   "annotations": [
      {
         "class_id": 0,
         "score": 0.943,
         "left": 111,
         "top": 134,
         "width": 61,
         "height": 128
      },
      {
         "class_id": 0,
         "score": 0.0013,
         "left": 161,
         "top": 250,
         "width": 79,
         "height": 143
      },
      {
         "class_id": 1,
         "score": 0.0133,
         "left": 101,
         "top": 185,
         "width": 42,
         "height": 130
      }
   ]
}
```

主成分分析 (PCA) 算法

PCA 是一种自主型机器学习算法，它试图在数据集内减少维数 (特征个数)，同时仍保留尽可能多的信息。这是通过查找称为成分的新特征集完成的，这些功能是原始特征的复合体，它们彼此不相关。它们还受到约束，因此第一个成分在数据中可能存在最大的可变性，第二个成分是第二个最易变的，以此类推。

在 Amazon SageMaker 中，PCA 以两种模式运行，具体取决于场景：

- 定期的：针对具有稀疏数据以及适度数量的观察和特征的数据集。
- 随机化：针对具有大量观察和特征的数据集。此模式使用近似算法。

PCA 使用表格数据。

这些行表示您想要嵌入到较低维度空间中的观察。这些列表示您想要为其查找的缩小近似的特征。该算法计算协方差矩阵 (或在分布方式下的近似)，然后对此摘要执行单值分解以产生主成分。

主题

- PCA 算法的输入/输出接口 (p. 1701)
- PCA 算法的 EC2 实例建议 (p. 1701)
- PCA 示例笔记本 (p. 1701)
- PCA 工作原理 (p. 1701)
- PCA 超参数 (p. 1702)
- PCA 响应格式 (p. 1703)

PCA 算法的输入/输出接口

在训练时，PCA 期望在训练通道上提供数据，并（可选）支持一个传递到测试数据集的数据集，该数据集由最终算法进行评分。`recordIO-wrapped-protobuf` 和 `CSV` 格式均支持用于训练。您可以使用文件模式或管道模式，针对格式为 `recordIO-wrapped-protobuf` 或 `CSV` 的数据训练模型。

对于推理，PCA 支持 `text/csv`、`application/json` 和 `application/x-recordio-protobuf`。结果以 `application/json` 或 `application/x-recordio-protobuf` 格式返回，且具有向量“投影”。

有关输入和输出文件格式的更多信息，请参阅PCA 响应格式 (p. 1703)（对于推理）和PCA 示例笔记本 (p. 1701)。

PCA 算法的 EC2 实例建议

PCA 支持 GPU 和 CPU 计算。哪个实例类型具有最高性能取决于输入数据的具体程度。

PCA 示例笔记本

有关演示如何使用 SageMaker 主成分分析算法来分析 MNIST 数据集中从 0 到 9 的手写数字图像的示例笔记本，请参阅。将 PCA 与 MNIST 结合使用简介. 有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147). 创建笔记本实例并将其打开后，选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。使用 NTM 算法的主题建模示例笔记本位于 Amazon 算法简介部分中。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

PCA 工作原理

主成分分析 (PCA) 是一种自主型机器学习算法，它在数据集内减少维数 (特征个数)，同时仍保留尽可能多的信息。

PCA 通过查找称为成分的新特征集减少维数，这些功能是原始特征的复合体，但彼此不相关。第一个成分在数据中可能存在最大的可变性，第二个成分是第二个最易变的，以此类推。

它是一种自主型维数减少算法。在自主学习中，不使用可能与训练数据集中的对象相关联的标签。

假定输入中的矩阵包含行 $x_1, \ldots, x_n$，每个行具有维度 `1 * d`，数据将被分成小批量行，并分布到训练节点（工作线程）中。然后，每个工作线程计算其数据的摘要。然后在计算结束时，不同工作线程的摘要统一为一个解决方案。

模式

Amazon SageMaker PCA 算法使用两种模式之一来计算这些摘要，具体取决于情况：

- 常规：针对具有稀疏数据以及适度数量的观察和特征的数据集。
- 随机：针对具有大量观察和特征的数据集。此模式使用近似算法。

作为算法的最后一步，它在统一解决方案上执行单值分解，然后将会从中导出主成分。

模式 1：定期

工作线程联合计算 $\sum x_i^T x_i$ 和 $\sum x_i$。

Note

由于 x_i 为 `1 * d` 个行向量，$x_i^T x_i$ 是一个矩阵（非标量）。在代码中使用行向量可以使我们获得高效的缓存。

协方差矩阵的计算方式为 $\sum x_i^T x_i - (1/n)(\sum x_i)^T \sum x_i$，其最前面的 `num_components` 个奇异向量构成模型。

Note

如果 `subtract_mean` 是 `False`，我们会避免计算和减去 $\sum x_i$。

当满足以下条件时使用此算法：向量的维度 `d` 足够小，以致 d^2 可以放入内存中。

模式 2：随机

当输入数据集中的特征数量较大时，我们使用一个方法来近似计算协方差指标。对于维度 `b * d` 的每个小批量 X_t，我们随机初始化一个我们与每个小批量相乘的 `(num_components + extra_components) * b` 矩阵，从而创建一个 `(num_components + extra_components) * d` 矩阵。这些矩阵的总和由工作线程计算，服务器在最终 `(num_components + extra_components) * d` 矩阵上执行 SVD。其右上方的 `num_components` 单向量是输入矩阵的顶部单向量的近似值。

指定 ℓ = `num_components + extra_components`。假定一个维度 `b * d` 的小批量 X_t，工作线程会提取维度 $\ell * b$ 的随机矩阵 H_t。根据环境是使用 GPU 还是 CPU 以及维度大小，矩阵是随机签名矩阵 (其中每个条目是 `+-1`) 或 FJLT (快速 Johnson Lindenstrauss 转换；有关信息，请参阅 FJLT 转换跟进文章)。随后工作线程会计算 $H_t X_t$ 并维护 $B = \sum H_t X_t$。工作线程还维护 h^T、$H_1, \ldots, H_T$（`T` 为小批量的总数）的列总和以及 `s`（所有输入行的总和）。在处理完整个数据碎片后，工作线程会向服务器发送 `B`、`h`、`s` 和 `n` (输入行的数量)。

将不同输入对服务器表示为 B^1, h^1, s^1, n^1。服务器会计算 `B`、`h`、`s`、`n` 的各自输入的总和。然后，它计算 $C = B - (1/n) h^T s$，并查找其奇异值分解。`C` 的右上单向量和单值被用作解决问题的近似方法。

PCA 超参数

在 `CreateTrainingJob` 请求中，您可以指定训练算法。您还可以指定算法特定的超参数作为字符串到字符串映射。下表列出了 Amazon SageMaker 为 PCA 训练算法提供的超参数。有关 PCA 的工作方式的更多信息，请参阅PCA 工作原理 (p. 1701)。

参数名称	描述
`feature_dim`	输入维度。

参数名称	描述
	必填 有效值：正整数
mini_batch_size	小批中的行数。 必填 有效值：正整数
num_components	要计算的主成分数量。 必填 有效值：正整数
algorithm_mode	用于计算主成分的模式。 可选 有效值：常规或随机 默认值：常规
extra_components	随着值增加，答案变得更准确，但运行时和内存消耗呈线性增长。默认值 -1 表示最大值为 10 和 num_components。仅对随机模式有效。 可选 有效值：非负整数或 -1 默认值：–1
subtract_mean	指示在训练期间和在进行推理时数据是否应该是无偏移的。 可选 有效值：其中之一真的要么假的 默认值：true

PCA 响应格式

所有 Amazon SageMaker 内置算法遵守中所述的常见输入推理格式。常见数据格式-推理. 本主题包含 SageMaker PCA 算法的可用输出格式列表。

JSON 响应格式

ACCEPT — applic/JSON

```
{
    "projections": [
        {
            "projection": [1.0, 2.0, 3.0, 4.0, 5.0]
        },
        {
            "projection": [6.0, 7.0, 8.0, 9.0, 0.0]
        },
        ....
```

```
    ]
}
```

JSONLINES 响应格式

ACCEPT — applicition /jsonlines

```
{ "projection": [1.0, 2.0, 3.0, 4.0, 5.0] }
{ "projection": [6.0, 7.0, 8.0, 9.0, 0.0] }
```

RECORDIO 响应格式

ACCEPT — applicition /x-recordio-protobuf

```
[
    Record = {
        features = {},
        label = {
            'projection': {
                keys: [],
                values: [1.0, 2.0, 3.0, 4.0, 5.0]
            }
        }
    },
    Record = {
        features = {},
        label = {
            'projection': {
                keys: [],
                values: [1.0, 2.0, 3.0, 4.0, 5.0]
            }
        }
    }
]
```

随机森林砍伐 (RCF) 算法

亚马逊 SageMaker 随机森林砍伐 (RCF) 是一种自主算法，用于检测数据集中的异常数据点。这些数据点是与良好结构或模式化数据存在偏差的观察数据。异常可以表现为时间序列数据中意外峰值、周期性中断或无法分类的数据点。它们很容易描述，表现为在图中查看时，它们通常很容易与“常规”数据区分。数据集中包括这些异常会显著增加机器学习任务的复杂性，因为“常规”数据通常使用简单模型描述。

对于每个数据点，RCF 关联一个异常分数。低分数值表明该数据点被视为“正常”。高分数值表明数据中存在异常。“低”和“高”的定义取决于应用，但常见做法建议将落在平均分数的三个标准偏差之外的分数视为异常。

异常检测算法在一维时间序列数据上有很多应用，例如流量分析或音量峰值检测，而 RCF 设计用于任意维度的输入。亚马逊 SageMaker RCF 可以很好地随着特征数、数据集大小和实例数缩放。

主题

- RCF 算法的输入/输出接口 (p. 1705)
- RCF 算法的实例建议 (p. 1705)
- RCF 示例笔记本 (p. 1705)
- RCF 工作原理 (p. 1706)
- RCF 超参数 (p. 1708)
- 优化 RCF 模型 (p. 1709)
- RCF 响应格式 (p. 1709)

RCF 算法的输入/输出接口

亚马逊 SageMaker 随机森林砍伐支持train和test数据通道。可选测试通道用于计算所标记数据的准确度、精度、召回率和 F1 分数指标。训练和测试数据内容类型可为 application/x-recordio-protobuf 或 text/csv 格式。对于测试数据，在使用文本/csv 格式时，内容必须指定为 text/csv;label_size=1，其中每行的第一列表示异常标签：1 表示异常数据点，0 表示普通数据点。您可以使用文件模式或管道模式，针对格式为 recordIO-wrapped-protobuf 或 CSV 的数据训练 RCF 模型。

火车频道只支持S3DataDistributionType=ShardedByS3Key而且测试频道只支持S3DataDistributionType=FullyReplicated. 以下示例使用亚马逊 SageMaker Python 开发工具包.

> Note
>
> 这些区域有：sagemaker.inputs.s3_input方法已重命名为sagemaker.inputs.TrainingInput在SageMaker Python 开发工具包 v2.

```
import sagemaker

# specify Random Cut Forest training job information and hyperparameters
rcf = sagemaker.estimator.Estimator(...)

# explicitly specify "ShardedByS3Key" distribution type
train_data = sagemaker.inputs.TrainingInput(
     s3_data=s3_training_data_location,
     content_type='text/csv;label_size=0',
     distribution='ShardedByS3Key')

# run the training job on input data stored in S3
rcf.fit({'train': train_data})
```

为避免执行角色方面的常见错误，请确保您拥有所需的执行角色，AmazonSageMakerFullAccess和AmazonEC2ContainerRegistryFullAccess. 为避免映像不存在的常见错误或其权限不正确，请确保 ECR 映像不大于训练实例上分配的磁盘空间。为避免这种情况，请在具有足够磁盘空间的实例上运行训练作业。此外，如果你的 ECR 图像来自不同Amazon账户的弹性容器服务 (ECS) 存储库，并且您没有设置存储库权限来授予访问权限，这将导致错误。请参阅ECR 存储库权限了解有关设置存储库策略声明的更多信息。

请参阅S3DataSource有关自定义 S3 数据源属性的更多信息。最后，为了利用多实例训练，训练数据必须至少分区为与实例相同数量的文件。

对于推理，RCF 支持 application/x-recordio-protobuf、text/csv 和 application/json 输入数据内容类型。有关更多信息，请参阅内置算法的常见数据格式 (p. 1579)文档。RCF 推理返回 application/x-recordio-protobuf 或 application/json 格式的输出。这些输出数据中的每个记录包含各个输入数据点对应的异常分数。有关更多信息，请参阅常见数据格式 -- 推理。

有关输入和输出文件格式的更多信息，请参阅RCF 响应格式 (p. 1709)（对于推理）和RCF 示例笔记本 (p. 1705)。

RCF 算法的实例建议

对于训练，我们建议使用 ml.m4、ml.c4 和 ml.c5 实例系列。对于推理，我们特别建议使用 ml.c5.xl 实例类型，用于实现最佳性能以及最低的每小时使用成本。尽管在技术上，算法可以在 GPU 实例类型上运行，但它没有充分利用 GPU 硬件。

RCF 示例笔记本

有关如何训练 RCF 模型并使用该模型执行推理的示例，请参阅简介 SageMaker 随机森林砍伐笔记本。有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon

SageMaker 笔记本实例 (p. 147). 创建笔记本实例并将其打开后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

RCF 工作原理

亚马逊 SageMaker 随机森林砍伐 (RCF) 是一种自主算法，用于检测数据集中的异常数据点。这些数据点是与良好结构或模式化数据存在偏差的观察数据。异常可以表现为时间序列数据中意外峰值、周期性中断或无法分类的数据点。它们很容易描述，表现为在图中查看时，它们通常很容易与“常规”数据区分。数据集中包括这些异常会显著增加机器学习任务的复杂性，因为“常规”数据通常使用简单模型描述。

RFC 算法的主要理念是创建森林，其中的每棵树使用训练数据样本的一个分区获得。例如，首先确定输入数据的随机样本。然后，根据森林中树的数量对随机样本分区。向每个树提供一个这样的分区，并将这些点的子集整理到 k-d 树中。由树分配给数据点的异常分数定义为将该点添加到树中导致的树的复杂性变化；这大致与点在树中生成的深度成反比。随机森林砍伐通过从组成的每棵树计算平均分数，并根据样本大小缩放结果，从而分配异常分数。RCF 算法基于引用 [1] 中所述的算法之一。

随机采样数据

RCF 算法的第一步是获取训练数据的随机样本。具体而言，假设我们希望从总共 N 个数据点中提取 K 个数据点的样本。如果训练数据集足够小，可以使用整个数据集，并且我们可以从此数据集随机提取 K 个元素。但是，训练数据经常会太大，无法一次使用，此方法不可行。我们改用一种称为水库采样的技术。

水库采样是一种算法，可用于从一个数据集 $S = \{S_1, \ldots, S_N\}$ 中有效地提取随机样本，其中数据集中的元素只能观察一次或者批量观察。事实上，即使 N 不是事先知道，水库采样仍可工作。如果只请求一个样本，例如当 $K = 1$ 时，算法类似于：

算法：储存器采样

- 输入：数据集或数据流 $S = \{S_1, \ldots, S_N\}$
- 初始化随机样本 $X = S_1$
- 对于每个观察到的采样 $S_n, n = 2, \ldots, N$：
  - 选取统一随机号码 $\xi \in [0, 1]$
  - 如果 $\xi < 1/n$
    - Set $X = S_n$
- 返回值 X

此算法选择随机采样，这样对于所有 $n = 1, \ldots, N$，可计算 $P(X = S_n) = 1/N$。当 $K > 1$ 时，算法更复杂。此外，必须区分使用替换和不使用替换的随机采样。RCF 基于 [2] 中所述的算法，对训练数据执行增强水库采样而不使用替换。

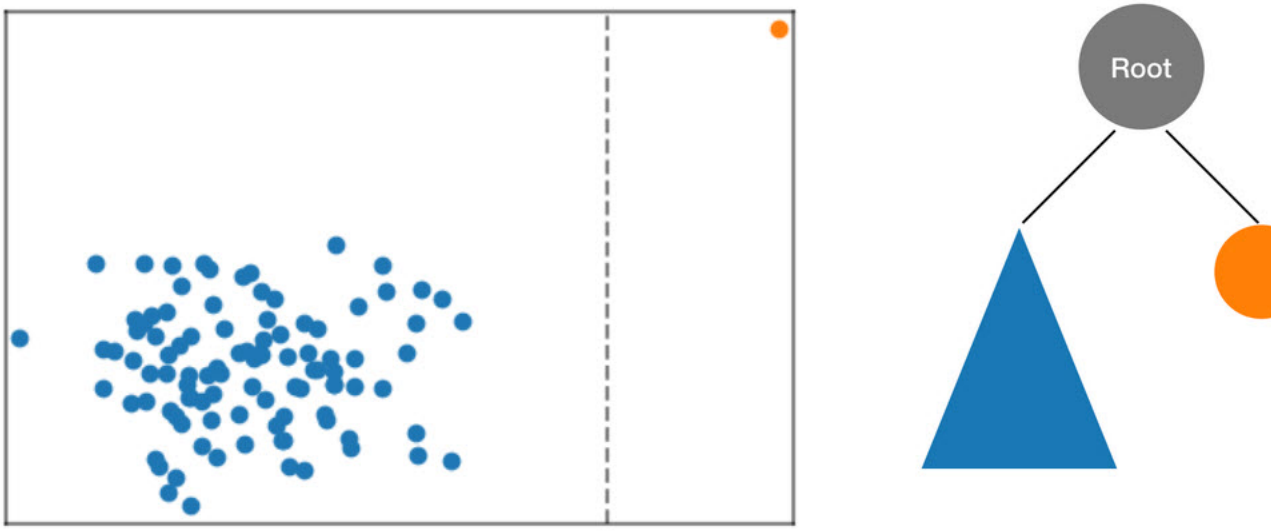
训练 RFC 模型和生成推理

RCF 中的下一步是使用随机采样的数据构造随机森林砍伐。首先，该样本分为多个大小相等的分区，其数量与森林中树的数量相同。然后，每个分区发送到单个树。树通过将数据域分区到边界框，以递归方式将其分区组织到二进制树中。

这个过程可以通过一个例子来很好地说明。假设向一颗树提供了以下二维数据集。对应的树初始化到根节点：

一个二维数据集，大多数数据位于集群 (蓝色) 中，但一个异常数据点 (橙色) 除外。使用根节点初始化树。

RCF 算法首先计算数据的边界框，选择随机维 (向具有较高“方差”的维提供更多权重)，然后通过该维随机确定 Hyperplane“砍伐”的位置，从而将这些数据组织到树中。生成的两个子空间定义自己的子树。在本示例中，通过砍伐将单一点与样本中其余数据分离开。生成的二进制树的第一级包含两个节点，其中一个由初始砍伐左侧点的子树组成，另一个表示右侧的单一点。



随机砍伐对二维数据集分区。异常数据点更可能隔离在边界框中，相比其他点具有较小的树深度。

然后，对两等分的数据计算左侧和右侧的边界框，重复该过程，直至树的每个叶表示来自样本的一个数据点。请注意，如果单一点足够远，则随机砍伐更可能产生点隔离。此观察提供的直觉感受是，大致来说，树深度与异常分数成反比。

使用经过训练的 RCF 模型执行推理时，按照每棵树报告的分数的平均值报告最终异常分数。请注意，经常会出现新的数据点不在树中的情况。要确定与新数据点关联的分数，请将数据点插入指定树中，按照与上述训

练过程相等的方式，高效 (并且临时) 重组树。也就是说，就像输入数据点是最初用于构造树时所用样本的成员那样生成了树。报告的分数与输入点在树中的深度成反比。

选择超参数

用于优化 RCF 模型的主要超参数为 num_trees 和 num_samples_per_tree。增加 num_trees 具有减少在异常分数中观察到的噪音的效果，因为最终分数是各个树报告的分数的平均值。尽管最优值取决于应用，我们建议使用 100 个树开始，以在分数噪声和模型复杂性之间取得平衡。请注意，推理时间与树的数量成正比。虽然训练时间也会受影响，但主要由上述的水库采样算法决定。

参数 num_samples_per_tree 与数据集中的预期异常密度相关。具体而言，num_samples_per_tree 的选择应该使得 1/num_samples_per_tree 大致相当于异常数据与普通数据的比率。例如，如果每个树中使用 256 个样本，则我们预期数据包含 1/256 的异常，或者大约为 0.4% 的时间。同样，此超参数的最优值取决于应用。

参考

1. Sudipto Guha、Nina Mishra、Gourav Roy 和 Okke Schrijvers。“基于流上异常检测的可靠随机森林砍伐。” InMachine Learning 国际会议，第 2712-2721 页。2016 年。
2. Byung-Hoon Park、George Ostrouchov、Nagiza F. Samatova 和 Al Geist。“可在数据流中进行替换的基于水库的随机采样。” 2004 年 SIAM 国际数据挖掘会议的会议记录，第 492-496 页。Society for Industrial and Applied Mathematics，2004 年。

RCF 超参数

在 CreateTrainingJob 请求中，您可以指定训练算法。您还可以指定算法特定的超参数作为字符串到字符串映射。下表列出了 Amazon 超参数。 SageMaker RCF 算法。有关更多信息，包括如何选择超参数的建议，请参阅 RCF 工作原理 (p. 1706)。

参数名称	描述
feature_dim	数据集中的特征数。（如果使用随机森林砍伐评估程序，则会为您计算此值而无需指定。） 必填 有效值：正整数 (最小值：1，最大值：10000)
eval_metrics	用于对标记的测试数据集评分的指标列表。可以为输出选择以下指标： • accuracy- 返回正确预测的比例。 • precision_recall_fscore - 返回正精度、负精度、召回率和 F1 分数。 可选 有效值：可能值列表，获取自 accuracy 或 precision_recall_fscore。 默认值：二者accuracy、precision_recall_fscore计算。
num_samples_per_tree	从训练数据集向每个树提供的随机样本数。 可选 有效值：正整数 (最小值：1，最大值：2048)

参数名称	描述
	默认值：256
num_trees	森林中树的数量。 可选 有效值：正整数 (最小值：50，最大值：1000) 默认值：100

优化 RCF 模型

自动模型优化（也称作超参数调整或超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

Amazon SageMaker RCF 算法是一种无监督异常检测算法，需要标记的测试数据集进行超参数优化。RCF 计算测试数据点的异常分数，然后当数据分数超出平均分数三个标准偏差时，将数据点标记为异常。这称为三西格玛极限启发式。F1 分数基于计算标签和实际标签之差。超参数优化作业可找出最大化该分数的模型。超参数优化的成功取决于三西格玛极限启发式对测试数据集的适用性。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

RCF 算法计算的指标

RCF 算法在训练期间计算以下指标。优化模型时，请选择此指标作为目标指标。

指标名称	描述	优化方向
test:f1	测试数据集的 F1 分数基于计算标签和实际标签之差。	最大化

可优化 RCF 超参数

您可以使用以下超参数优化 RCF 模型。

参数名称	参数类型	建议的范围
num_samples_per_tree	IntegerParameterRanges	MinValue：1，最大值：2048
num_trees	IntegerParameterRanges	MinValue：50，最大值：1000

RCF 响应格式

Amazon 所有 SageMaker 内置算法遵循中所述的常见输入推理格式。常见数据格式-推理. 请注意，SageMaker 随机森林砍伐同时支持密集和稀疏 JSON 和 RecordIO 格式。本主题包含的可用输出格式列表。SageMaker RCF 算法。

JSON 响应格式

ACCEPT：application/json。

```
    {


        "scores":    [


            {"score": 0.02},


            {"score": 0.25}


        ]


    }
```

JSONLINES 响应格式

ACCEPT: application/jsonlines。

```
{"score": 0.02},
{"score": 0.25}
```

RECORDIO 响应格式

ACCEPT：application/x-recordio-protobuf。

```
    [


         Record = {


             features = {},


             label = {


                 'score': {


                     keys: [],


                     values: [0.25]  # float32


                 }
```

```
            }

        },

        Record = {

            features = {},

            label = {

                'score': {

                    keys: [],

                    values: [0.23]  # float32

                }

            }

        }

]
```

语义分割算法

SageMaker 语义分割算法为开发计算机视觉应用程序提供了一种细粒度的像素级方法。它使用预定义类集中的类标签来标记图像中的每个像素。标记是理解场景的基础，这对于越来越多的计算机视觉应用（例如自动驾驶车辆、医学成像诊断和机器人感知）来说至关重要。

为了比较，SageMaker图像分类算法 (p. 1620)是一种仅分析整个图像并将图像分类为多个输出类别之一的监督学习算法。对象检测算法 (p. 1691) 是一种指导式学习算法，用于检测和分类图像中对象的所有实例。它使用矩形边界框指示图像中每个对象的位置和比例。

由于语义分割算法对图像中的每个像素进行分类，因此，它还提供有关图像中包含的对象形状的信息。分割输出表示为灰度图像，称作分割掩膜. 分割遮罩是与输入图像形状相同的灰度图像。

SageMaker 语义分割算法是使用MxNet Gluon 框架和 Gluon 简历工具包. 它为您提供三种可选择用来训练深度神经网络的内置算法。您可以使用全卷积网络 (FCN) 算法、金字塔场景解析 (PSP) 算法或 DeepLabV3。

这三种算法均具有两个不同的组件：

- 这些区域有：骨干（或编码器）— 生成可靠的要素激活映射的网络。
- 这些区域有：解码器— 从编码的激活映射构造分割掩膜的网络。

您还可以选择 FCN、PSP 和 DeepLabv3 算法的主干：resnet50 或 Resnet101. 这些主干包括最初在 ImageNet 分类任务中训练的预训练构件。您可以使用自己的数据微调这些主干以进行分割。或者，您可以仅使用自己的数据从头开始初始化和训练这些网络。解码器从未预先训练过。

要部署训练后的模型进行推理，请使用 SageMaker 托管服务。在推理期间，您可以 PNG 图像形式或以每个像素的每个类的一组概率的形式请求分割掩膜。您可以将这些掩膜用作包含其他下游图像处理或其他应用程序的较大管道的一部分。

主题

- 语义分割示例笔记本 (p. 1712)
- 语义分割算法的输入/输出接口 (p. 1712)
- 语义分割算法的 EC2 实例建议 (p. 1714)
- 语义分割超参数 (p. 1715)
- 调整语义分割模型 (p. 1719)

语义分割示例笔记本

有关使用 SageMaker 语义分割算法来训练模型并部署它以执行推理的示例 Jupyter 笔记本，请参阅语义分割示例. 有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147).

要查看所有 SageMaker 示例的列表，请创建并打开笔记本实例，并选择SageMaker 示例选项卡。示例语义分割笔记本位于 Amazon 算法简介下。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

语义分割算法的输入/输出接口

SageMaker 语义分割预计客户的训练数据集将启用Amazon Simple Storage Service (Amazon S3). 训练后，它会在 Amazon S3 上生成结果模型构件。SageMaker 语义分割的输入接口格式与大多数标准化语义分割基准数据集的输入接口格式类似。预计 Amazon S3 中的数据集将以两个通道呈现（一个通道用于）`train`还有一个用于`validation`使用四个目录，两个用于图像，两个用于注释。注释预计是未压缩的 PNG 图像。数据集还可能具有一个标签映射，此映射描述了如何建立注释映射。否则，算法会使用默认值。它还支持增强的清单图像格式(`application/x-image`）用于直接从 Amazon S3 进行管道输入模式的训练。对于推理，终端节点接受具有 `image/jpeg` 内容类型的图像。

训练的工作方式

训练数据分为四个目录：`train`、`train_annotation`、`validation` 和 `validation_annotation`。上述目录均有一个通道。对于 `train_annotation` 和 `validation_annotation`，数据集还预计每个通道有一个 `label_map.json` 文件。如果您不提供这些 JSON 文件，SageMaker 会提供默认设置标签映射。

指定这些文件的数据集应类似于以下示例：

```
s3://bucket_name
    |
    |- train
                |
                | - 0000.jpg
                | - coffee.jpg
    |- validation
                |
                | - 00a0.jpg
```

```
                | - bananna.jpg
    |- train_annotation
                |
                | - 0000.png
                | - coffee.png
    |- validation_annotation
                |
                | - 00a0.png
                | - bananna.png
    |- label_map
                | - train_label_map.json
                | - validation_label_map.json
```

训练和验证目录中的每个 JPG 图像都在 `train_annotation` 和 `validation_annotation` 目录中具有带相同名称的相应 PNG 标签图像。此命名约定有助于算法在训练期间将标签与其对应的图像关联。`train`、`train_annotation`、`validation` 和 `validation_annotation` 通道是必需的。注释是单通道 PNG 图像。只要图像中的元数据（模式）帮助算法将注释图像读取为单通道 8 位无符号整数，格式就有效。有关我们对模式的支持的更多信息，请参阅 Python 图像库文档。我们建议使用 8 位像素，真彩色 `P` 模式。

使用模式时，编码的图像是一个简单的 8 位整数。为了从该映射到标签的映射，此算法对每个通道使用一个映射文件，称作标签映射。标签映射用于将图像中的值与实际标签索引进行映射。在默认标签映射中（如果不提供标签映射，则默认提供），注释矩阵（图像）中的像素值直接索引标签。这些图像可以是灰度 PNG 文件或 8 位索引 PNG 文件。未缩放默认情况下的标签映射文件如下：

```
{
  "scale": "1"
}
```

为了提供一些对比度以供查看，一些注释软件将标签图像按一定的比例缩放。为了支持这一点，SageMaker 语义分割算法提供了重新缩放选项，以将值缩小到实际标签值。当向下缩放不将值转换为适当的整数时，算法默认为小于或等于缩放值的最大整数。以下代码说明如何设置缩放值来重新缩放标签值：

```
{
  "scale": "3"
}
```

以下示例说明如何使用 `"scale"` 值重新缩放输入注释图像的 `encoded_label` 值（当这些值映射到用于训练的 `mapped_label` 值时）。输入注释图像中的标签值为 0、3、6，标度为 3，以便将其映射到 0、1、2 来进行训练：

```
encoded_label = [0, 3, 6]
mapped_label = [0, 1, 2]
```

在某些情况下，您可能需要为每个类指定特定的颜色映射。使用标签映射中的映射选项，如下面的 `label_map` 文件示例所示：

```
{
    "map": {
        "0": 5,
        "1": 0,
        "2": 2
    }
}
```

此示例的该标签映射为：

```
encoded_label = [0, 5, 2]
```

```
mapped_label = [1, 0, 2]
```

使用标签映射，您可以使用不同的注释系统和注释软件来获取数据，而无需进行大量预处理。您可以为每个通道提供一个标签映射。label_map 通道中标签映射的文件必须遵循四个目录结构的命名约定。如果您未提供标签映射，则算法假定比例为 1（默认值）。

使用增强清单格式进行训练

扩增清单格式使您可以使用图像文件在管道模式下进行训练，而无需创建 RecordIO 文件。增强清单文件包含数据对象并且应位于JSON 行格式，如CreateTrainingJob请求. 清单中的每一行都是一个条目，其中包含映像的 Amazon S3 URI 和注释映像的 URI。

清单文件中的每个 JSON 对象必须包含一个 source-ref 键。这些区域有：source-ref键应包含映像的 Amazon S3 URI 的值。将在 AttributeNames 参数值下面提供标签，如 CreateTrainingJob 请求中指定。它还可以在元数据标签下包含其他元数据，但算法会忽略这些元数据。在下面的示例中，AttributeNames 包含在图像和注释引用 ["source-ref", "city-streets-ref"] 列表中。这些名称必须附加了 -ref。在将语义分割算法与增强清单一起使用时，RecordWrapperType 参数的值必须为 "RecordIO"，ContentType 参数的值必须为 application/x-recordio。

```
{"source-ref": "S3 bucket location", "city-streets-ref": "S3 bucket location", "city-
streets-metadata": {"job-name": "label-city-streets", }}
```

有关增强清单文件的更多信息，请参阅通过增强清单文件，将数据集元数据提供给训练作业 (p. 2090)。

增量训练

您还可以使用以前使用 SageMaker 训练的模型为新模型的训练做种。当您想训练具有相同或类似数据的新模型时，此增量训练可节省训练时间。目前，仅使用内置 SageMaker 语义分割训练的模型支持增量训练。

要使用您自己的预训练模型，请为 CreateTrainingJob 请求将 ChannelName 指定为 InputDataConfig 中的“模型”。将模型通道的 ContentType 设置为 application/x-sagemaker-model。必须在新模型和上传到模型通道的预训练模型的输入超参数中一致地指定用于定义网络架构的 backbone、algorithm、crop_size 和 num_classes 输入参数。对于预训练的模型文件，您可以使用 SageMaker 输出中的压缩 (.tar.gz) 构件。您只能对输入数据使用图像格式。有关增量训练及其使用方式说明的更多信息，请参阅Amazon SageMaker 中的增量训练 (p. 2081)。

生成推理

要查询部署到终端节点的训练后的模型，您需要提供一个图像和一个 AcceptType，后者表示所需输出的类型。终端节点接受 image/jpeg 内容类型的 JPEG 图像。如果您请求的 AcceptType 为 image/png，该算法将输出一个带有分割掩膜的 PNG 文件，其格式与标签本身相同。如果您请求的 AcceptType 为 application/x-recordio-protobuf，该算法将返回以 recordio-protobuf 格式编码的类概率。后一种格式输出一个 3D 张量，其中第三维的大小与类的数量相同。该分量表示每个像素的每个类标签的概率。

语义分割算法的 EC2 实例建议

SageMaker 语义分割算法仅支持 GPU 实例进行训练，我们建议使用具有更多内存的 GPU 实例进行大批量训练。可以使用单机配置中的 P2/P3 EC2 Amazon Elastic Compute Cloud (Amazon EC2) 实例训练该算法。它支持以下用于训练的 GPU 实例：

- ml.p2.xlarge
- ml.p2.8xlarge
- ml.p2.16xlarge
- ml.p3.2xlarge
- ml.p3.8xlarge
- ml.p3.16xlarge

对于推理，您可以使用 CPU 实例（例如 c5 和 m5）和/或 GPU 实例（例如 p2 和 p3）。有关为推理提供不同的 CPU、GPU、内存和网络容量组合的实例类型的信息，请参阅 Amazon SageMaker ML 实例类型。

语义分割超参数

以下各表列出了 Amazon SageMaker 语义分割算法支持的用于网络架构、数据输入和训练的超级参数。您可以在 `CreateTrainingJob` 请求的 `AlgorithmName` 中为训练指定语义分割。

网络架构超参数

参数名称	描述
`backbone`	要用于算法的编码器组件的主干。 可选 有效值：`resnet-50`，`resnet-101` 默认值：`resnet-50`
`use_pretrained_model`	是否将预训练模型用于主干。 可选 有效值：`True`，`False` 默认值：`True`
`algorithm`	要用于语义分割的算法。 可选 有效值： • `fcn`：全卷积网络 (FCN) 算法 • `psp`：金字塔场景解析 (PSP) 算法 • `deeplab`：DeepLab V3 算法 默认值：`fcn`

数据超参数

参数名称	描述
`num_classes`	要分割的类的数量。 必填 有效值：2 ≤ 正整数 ≤ 254
`num_training_samples`	训练数据中的示例数。该算法使用此值来设置学习率计划程序。 必填 有效值：正整数
`base_size`	定义裁剪前如何重新缩放图像。图像被重新缩放，以便将长尺寸长度设置为 `base_size` 乘以 0.5 到 2.0 之间的随机数，并计算短尺寸以保持长宽比。

参数名称	描述
	可选 有效值：正整数 > 16 默认值：520
crop_size	训练期间输入的图像大小。我们将根据 base_size 随机重新缩放输入图像，然后按照边长等于 crop_size 进行随机的方形裁剪。crop_size 将自动舍入到 8 的倍数。 可选 有效值：正整数 > 16 默认值：240

训练超参数

参数名称	描述
early_stopping	是否在训练期间使用提前停止逻辑。 可选 有效值：True，False 默认值：False
early_stopping_min_epochs	必须运行的纪元的最小数量。 可选 有效值：整数 默认值：5
early_stopping_patience	在算法强制提前停止之前满足较低性能容差的纪元数。 可选 有效值：整数 默认值：4
early_stopping_tolerance	如果训练作业分数 mIOU 的相对改善小于此值，则提前停止将纪元视为未改善。仅当 early_stopping = True 时使用。 可选 有效值：0 ≤ 浮点型 ≤ 1 默认值：0.0
epochs	用于训练的纪元的数量。 可选 有效值：正整数

参数名称	描述
	默认值：10
gamma1	rmsprop 的平方梯度的移动平均值的衰减系数。仅用于 rmsprop。 可选 有效值：0 ≤ 浮点型 ≤ 1 默认值：0.9
gamma2	rmsprop 的动量因子。 可选 有效值：0 ≤ 浮点型 ≤ 1 默认值：0.9
learning_rate	初始学习率。 可选 有效值：0 < 浮点型 ≤ 1 默认值：0.001
lr_scheduler	学习速率计划的形状，可控制其随时间的推移而减少。 可选 有效值： • step：逐步衰减，其中学习率减少（乘以）lr_scheduler_factor在指定的纪元之后lr_scheduler_step. • poly：使用多项式函数的平滑衰减。 • cosine：使用余弦函数的平滑衰减。 默认值：poly
lr_scheduler_factor	如果lr_scheduler设置为step，减少（多个）的比率learning_rate在指定的每个时代之后lr_scheduler_step. 否则，请忽略。 可选 有效值：0 ≤ 浮点型 ≤ 1 默认值：0.1
lr_scheduler_step	一个逗号分隔的时代列表，之后learning_rate减少（乘以）lr_scheduler_factor. 例如，如果值设置为"10, 20"，然后learning-rate减少了lr_scheduler_factor在 10 时代之后，20 世纪之后再次受到这个因素的影响。 有条件必需如果lr_scheduler设置为step. 否则，请忽略。 有效值：字符串 默认值：（没有默认值，因为使用时需要该值。）

参数名称	描述
mini_batch_size	训练的批次大小。使用大型 mini_batch_size 通常会加快训练速度，但可能会导致内存不足。内存使用率受 mini_batch_size 和 image_shape 参数的值以及主干架构的影响。 可选 有效值：正整数 默认值：16
momentum	sgd 优化程序的动量。当您使用其他优化程序时，语义分割算法将忽略此参数。 可选 有效值：0 < 浮点型 ≤ 1 默认值：0.9
optimizer	优化程序的类型。有关优化程序的更多信息，请选择相应的链接： • adam：适应性动量估计 • adagrad：自适应梯度下降 • nag：Nesterov 加速梯度 • rmsprop： 均方根传播 • sgd：随机梯度下降 可选 有效值：adam，adagrad，nag，rmsprop，sgd 默认值：sgd
syncbn	如果设置为 True，则为 GPU 中处理的所有样本计算批量标准化平均值和方差。 可选 有效值：True，False 默认值：False

参数名称	描述
validation_mini_batch_size	验证的批次大小。大型 mini_batch_size 通常会加快训练速度，但可能会导致内存不足。内存使用率受 mini_batch_size 和 image_shape 参数的值以及主干架构的影响。 • 要在不裁剪图像的情况下对整个图像的验证评分，请将此参数设置为 1。如果要整体测量整个图像的性能，请使用此选项。 Note 将 validation_mini_batch_size 参数设置为 1 会导致算法为每个图像创建一个新的网络模型。这可能会减慢验证和训练的速度。 • 要将图像裁剪为 crop_size 参数中指定的大小，即使在评估期间，也要将此参数设置为大于 1 的值。 可选 有效值：正整数 默认值：16
weight_decay	sgd 优化程序的权重衰减系数。当您使用其他优化程序时，算法将忽略此参数。 可选 有效值：0 < 浮点型 < 1 默认值：0.0001

调整语义分割模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

语义分割算法计算的指标

语义分割算法报告了两个验证指标。优化超参数值时，请选择这些指标之一作为目标。

指标名称	描述	优化方向
validation:mIOU	预测分割与地面真相交点的区域除以验证集中的图像之间的并集区域。也被称为 Jaccard 指数。	最大化
validation:pixel_accuracy	验证集中图像中正确分类的像素百分比。	最大化

可调谐语义分割超参数

您可以优化语义分割算法的以下超参数。

参数名称	参数类型	建议的范围
learning_rate	ContinuousParameterRange	最 MinValue：1e-4，MaxValue：1e-1
mini_batch_size	IntegerParameterRanges	MinValue：1，MaxValue：128
momentum	ContinuousParameterRange	MinValue：0.9，MaxValue：0.999
optimzer	CategoricalParameterRanges	['sgd'、'adam'、'adadelta']
weight_decay	ContinuousParameterRange	最 MinValue：1e-5，MaxValue：1e-3

序列到序列算法

Amazon SageMaker 序列到序列是一种受监管的学习算法，其中的输入是一系列标记（例如，文本、音频），生成的输出是另一序列标记。示例应用程序包括：机器翻译 (从一种语言输入一个句子，并预测该句子在另一种语言中将是什么)、文本摘要 (输入较长的单词串并预测较短的摘要单词串)、语音到文本 (在标记中转换为输出语句的音频剪辑)。最近，这个领域中的问题已经通过深度神经网络成功地建模，这些网络显示比以前的方法有了显著的性能提升。Amazon SageMaker seq2seq 将循环神经网络 (RNN) 和卷积神经网络 (CNN) 模型用于编码解码器体系结构。

主题

- 序列到序列算法的输入/输出接口 (p. 1720)
- 序列到序列算法的 EC2 实例建议 (p. 1721)
- 序列到序列示例笔记本 (p. 1721)
- 序列到序列的工作原理 (p. 1721)
- 序列到序列超参数 (p. 1722)
- 优化序列到序列模型 (p. 1729)

序列到序列算法的输入/输出接口

训练

SageMaker seq2seq 需要采用 RecordIO-Protobuf 格式的数据。但是，通常情况下，需要整数而不是浮点数的标记。

seq2seq 示例笔记本中包括一个将数据从标记化文本文件转换为 protobuf 格式的脚本。通常，它将数据打包成 32 位整数张量，并生成必要的词汇表文件，这些文件是指标计算和推理所需的。

在进行预处理后，可以调用该算法进行训练。该算法需要有三个通道：

- train：它应包含训练数据 (例如，预处理脚本生成的 train.rec 文件)。
- validation：它应包含验证数据 (例如，预处理脚本生成的 val.rec 文件)。
- vocab：它应包含两个词汇表文件（vocab.src.json 和 vocab.trg.json）

如果算法在这三个通道中都找不到数据，则训练会导致错误。

推理

对于托管终端节点，推理支持两种数据格式。要使用空格分隔的文本标记执行推理，请使用 `application/json` 格式。否则，使用 `recordio-protobuf` 格式，以便使用整数编码的数据。这两种模式均支持对输入数据的批处理。`application/json`格式还允许您可视化关注矩阵。

- `application/json`：需要 JSON 格式的输入，返回 JSON 格式的输出。内容和接受类型都应该是 `application/json`。每个序列都将是一个带有空格分隔标记的字符串。当批中的源序列数较小时，建议使用此格式。它还支持以下附加配置选项：

  `configuration: {attention_matrix: true}`：返回特定输入序列的关注矩阵。
- `application/x-recordio-protobuf`：需要 `recordio-protobuf` 格式的输入，返回 `recordio-protobuf format` 格式的输出。内容和接受类型都应该是 `applications/x-recordio-protobuf`。对于此格式，源序列必须转换为整数列表以便进行后续的 protobuf 编码。建议使用此格式进行批量推理。

对于批量转换，推理支持 JSON 行格式。批量转换需要 JSON 行格式的输入，返回 JSON 行格式的输出。内容和接受类型都应该是 `application/jsonlines`。输入的格式如下所示：

```
content-type: application/jsonlines

{"source": "source_sequence_0"}
{"source": "source_sequence_1"}
```

响应的格式如下：

```
accept: application/jsonlines

{"target": "predicted_sequence_0"}
{"target": "predicted_sequence_1"}
```

有关如何将输入和输出序列化和反序列化为特定格式以进行推理的其他详细信息，请参阅序列到序列示例笔记本 (p. 1721)。

序列到序列算法的 EC2 实例建议

目前，Amazon SageMaker seq2seq 仅在 GPU 实例类型上支持，并且仅设置为在单个机器上训练。但它也提供对多个 GPU 的支持。

序列到序列示例笔记本

有关说明如何使用 SageMaker 序列到序列算法训练英语到德语翻译模型的示例笔记本，请参阅使用 SageMaker Seq2Seq 的机器翻译英语-德语示例. 有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147). 在您创建笔记本实例并打开之后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。使用 NTM 算法的主题建模示例笔记本位于 Amazon 算法简介部分中。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

序列到序列的工作原理

通常，序列到序列建模的神经网络由几个层组成，包括：

- 一个嵌入层。在此层中，输入矩阵是以稀疏的方式编码的输入标记 (例如，单热点编码)，映射到密集的特征层。这是必需的，因为高维特征向量比简单的单热点编码向量更能编码有关特定标记 (用于文本语料库的单词) 的信息。用预先训练过的单词向量 (如 FastText 或 Glove) 初始化此嵌入层或者在训练过程中随机初始化它并学习参数也是一种标准做法。
- 一个编码器层。将输入标记映射到高维特征空间之后，序列会通过编码器层传递，从而将 (整个序列的) 输入嵌入层中的所有信息压缩为固定长度的特征向量。通常，编码器由 RNN 型网络组成，如长短期内存 (LSTM) 或门控循环单元 (GRU)。(Colah 的博客非常详细地解释了 LSTM。)

- 一个解码器层。解码层采用此编码的特征向量，并生成标记的输出序列。此层通常也用 RNN 架构 (LSTM 和 GRU) 构建。

在给定源序列的情况下，对整个模型进行联合训练，使目标序列的概率最大化。该模型首先由 Sutskever 等人在 2014 年提出。

关注机制。编码器-解码器框架的缺点是，由于固定长度编码的特征向量可以包含多少信息的限制，模型性能会随着源序列长度的增加而降低。为了解决此问题，2015 年 Bahdanau 等人提出了关注机制。在关注机制中，解码器试图在编码器序列中查找最重要信息所在的位置，并使用该信息和以前解码的单词来预测序列中的下一个标记。

有关更多详细信息，请参阅 Luong 等人编写的白皮书基于关注的神经机器翻译的有效方法，该书解释并简化了各种关注机制的计算。此外，白皮书谷歌的神经机器翻译系统：弥合人工翻译和机器翻译之间的差距Wu 等人描述了 Google 机器翻译的架构，即在编码器和解码层之间使用跳过连接。

序列到序列超参数

参数名称	描述
`batch_size`	用于梯度下降的小批次大小。 可选 有效值：正整数 默认值：64
`beam_size`	光束搜索的光束的长度。在训练期间用于计算 `bleu`，以及在推理期间使用。 可选 有效值：正整数 默认值：5
`bleu_sample_size`	在训练过程中从验证数据集中选取以解码并计算 `bleu` 分数的实例数。设置为 -1 可使用完整验证集（如果选择 `bleu` 作为 `optimized_metric`）。 可选 有效值：整数 默认值：0
`bucket_width`	返回最多 (`max_seq_len_source`、`max_seq_len_target`) 个 (源、目标) 存储桶。数据的较长一侧使用 `bucket_width` 的步骤，而较短的一侧使用按平均目标/源长度比例缩减的步骤。如果一侧的最大长度超过另一侧，则该侧额外存储桶的宽度固定在 `max_len` 的那一侧。 可选 有效值：正整数 默认值：10

参数名称	描述
bucketing_enabled	设置为 false 可禁用分桶，展开到最大长度。 可选 有效值：true 或 false 默认值：true
checkpoint_frequency_num_batches	检查点并评估每个 x 批处理。此检查点超参数传递给 SageMaker 的 seq2seq 算法，以便提前停止和检索最佳模型。算法的检查点在算法的训练容器中本地运行，并且与 SageMaker 检查点不兼容。该算法暂时将检查点保存到本地路径，并在训练作业停止后将最佳模型工件存储到 S3 中的模型输出路径中。 可选 有效值：正整数 默认值：1000
checkpoint_threshold	在停止训练之前，允许在验证数据集上的 optimized_metric 中不改进检查点模型的最大数量。此检查点超参数传递给 SageMaker 的 seq2seq 算法，以便提前停止和检索最佳模型。算法的检查点在算法的训练容器中本地运行，并且与 SageMaker 检查点不兼容。该算法暂时将检查点保存到本地路径，并在训练作业停止后将最佳模型工件存储到 S3 中的模型输出路径中。 可选 有效值：正整数 默认值：3
clip_gradient	大于此的剪辑绝对梯度值。设置为负表示禁用。 可选 有效值：浮点值 默认值：1
cnn_activation_type	要使用的 cnn 激活类型。 可选 有效值：字符串。以下值之一：glu、relu、softrelu、sigmoid 或 tanh。 默认值：glu
cnn_hidden_dropout	卷积层之间的退出的丢弃概率。 可选 有效值：Float。范围为 [0,1]。 默认值：0

参数名称	描述
cnn_kernel_width_decoder	cnn 解码器的内核宽度。 可选 有效值：正整数 默认值：5
cnn_kernel_width_encoder	cnn 编码器的内核宽度。 可选 有效值：正整数 默认值：3
cnn_num_hidden	编码器和解码器的 cnn 隐藏单元的数量。 可选 有效值：正整数 默认值：512
decoder_type	解码器类型。 可选 有效值：字符串。rnn 或 cnn。 默认值：rnn
embed_dropout_source	源端嵌入的丢弃概率。 可选 有效值：Float。范围为 [0,1]。 默认值：0
embed_dropout_target	目标端嵌入的丢弃概率。 可选 有效值：Float。范围为 [0,1]。 默认值：0
encoder_type	编码器类型。rnn 架构基于 Bahdanau 等人提出的关注机制，cnn 架构基于 Gehring 等人提出的关注机制。 可选 有效值：字符串。rnn 或 cnn。 默认值：rnn

参数名称	描述
fixed_rate_lr_half_life	就 fixed_rate_* 计划程序的检查点数量来说，学习率的半生命周期。 可选 有效值：正整数 默认值：10
learning_rate	初始学习率。 可选 有效值：浮点值 默认值：0.0003
loss_type	训练的损失函数。 可选 有效值：字符串。cross-entropy 默认值：cross-entropy
lr_scheduler_type	学习率计划程序类型。plateau_reduce 表示每当 validation_accuracy 稳定状态上有 optimized_metric 时，就降低学习率。inv_t 是反时衰减。learning_rate/(1+decay_rate*t) 可选 有效值：字符串。plateau_reduce、fixed_rate_inv_t 或 fixed_rate_inv_sqrt_t。 默认值：plateau_reduce
max_num_batches	要处理的更新/批处理的最大数量。-1 表示不限次数。 可选 有效值：整数 默认值：–1
max_num_epochs	在停止调整之前通过训练数据传递的最大纪元数。如果传递此参数，则即使验证准确性没有提高，训练仍将持续到此数量纪元。如果不传递，则会忽略。 可选 有效值：小于或等于 max_num_epochs 的正整数。 默认值：none

参数名称	描述
max_seq_len_source	源序列长度的最大长度。比此长度长的序列被截断为此长度。 可选 有效值：正整数 默认值：100
max_seq_len_target	目标序列长度的最大长度。比此长度长的序列被截断为此长度。 可选 有效值：正整数 默认值：100
min_num_epochs	在通过 early_stopping 条件停止之前，训练必须运行的最少纪元数。 可选 有效值：正整数 默认值：0
momentum	用于 sgd 的动量常量。如果您使用的是 adam 或 rmsprop，请不要传递此参数。 可选 有效值：浮点值 默认值：none
num_embed_source	源标记的嵌入大小。 可选 有效值：正整数 默认值：512
num_embed_target	目标标记的嵌入大小。 可选 有效值：正整数 默认值：512
num_layers_decoder	解码器 rnn 或 cnn 的层数。 可选 有效值：正整数 默认值：1

参数名称	描述
num_layers_encoder	编码器 rnn 或 cnn 的层数。 可选 有效值：正整数 默认值：1
optimized_metric	用于使用早期停止进行优化的指标。 可选 有效值：字符串。perplexity、accuracy 或 bleu。 默认值：perplexity
optimizer_type	可供从中选择的优化程序。 可选 有效值：字符串。adam、sgd 或 rmsprop。 默认值：adam
plateau_reduce_lr_factor	与学习率相乘的系数 (对于 plateau_reduce). 可选 有效值：浮点值 默认值：0.5
plateau_reduce_lr_threshold	对于 plateau_reduce 计划程序，如果 optimized_metric 未改善这很多检查点，则用学习率乘以降低系数。 可选 有效值：正整数 默认值：3
rnn_attention_in_upper_layers	将关注传递到 rnn 的上层，如 Google NMT 文章中所述。仅在使用多个层时才适用。 可选 有效值：布尔值 (true 或 false) 默认值：true
rnn_attention_num_hidden	关注层的隐藏单元数量。默认值为 rnn_num_hidden。 可选 有效值：正整数 默认值：rnn_num_hidden

参数名称	描述
rnn_attention_type	编码器的关注模型。mlp 是指 concat，bilinear 是指 general（源自 Luong 等人的论文）。 可选 有效值：字符串。以下值之一：dot、fixed、mlp 或 bilinear。 默认值：mlp
rnn_cell_type	特定类型的 rnn 架构。 可选 有效值：字符串。lstm 或 gru。 默认值：lstm
rnn_decoder_state_init	如何从编码器初始化 rnn 解码器状态。 可选 有效值：字符串。last、avg 或 zero。 默认值：last
rnn_first_residual_layer	第一个具有剩余连接的 rnn 层，只有在编码器或解码器中的层数超过 1 时才适用。 可选 有效值：正整数 默认值：2
rnn_num_hidden	编码器和解码器的 rnn 隐藏单元的数量。这必须是 2 的倍数，因为默认情况下该算法使用双向长短期记忆 (LSTM)。 可选 有效值：正偶数 默认值：1024
rnn_residual_connections	将剩余连接添加到堆叠的 rnn。层数应超过 1。 可选 有效值：布尔值（true 或 false） 默认值：false

参数名称	描述
rnn_decoder_hidden_dropout	将上下文与解码器中的 rnn 隐藏状态结合的隐藏状态的丢弃概率。 可选 有效值：Float。范围为 [0,1]。 默认值：0
training_metric	用于跟踪验证数据训练的指标。 可选 有效值：字符串。perplexity 或 accuracy。 默认值：perplexity
weight_decay	权重衰减常量。 可选 有效值：浮点值 默认值：0
weight_init_scale	权重初始化规模（对于 uniform 和 xavier 初始化）。 可选 有效值：浮点值 默认值：2.34
weight_init_type	权重初始化的类型。 可选 有效值：字符串。uniform 或 xavier。 默认值：xavier
xavier_factor_type	Xavier 系数类型。 可选 有效值：字符串。in、out 或 avg。 默认值：in

优化序列到序列模型

自动模型优化（也称作超参数优化）通过运行很多在数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择可优化超参数、每个超参数的值范围和一个目标指标。您可以从算法计算的指标中选择目标指标。自动模型优化将搜索所选超参数以找到导致优化目标指标的模型的值组合。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

由序列到序列算法计算的指标

序列到序列算法报告在训练期间所计算的三个指标。在优化超参数值时，选择其中之一作为优化的目标。

指标名称	描述	优化方向
validation:accuracy	在验证数据集上计算的准确率。	最大化
validation:bleu	在验证数据集上计算的 Bleu 分数。由于 BLEU 计算成本高昂，您可以选择在验证数据集的随机子样本上计算 BLEU，以加速整体训练过程。使用 bleu_sample_size 参数指定子样本。	最大化
validation:perplexity	困惑度，是在验证数据集上计算的损失函数。困惑度评估经验样本与模型预测分布之间的交叉熵，这可用于评估模型预测样本值的效果有多好。能够很好地预测样本的模型具有低困惑度。	最小化

可优化序列到序列超级参数

您可为 SageMaker 序列到序列算法优化以下超参数。对序列到序列目标指标影响最大的超级参数包括：batch_size、optimizer_type、learning_rate、num_layers_encoder 和 num_layers_decoder。

参数名称	参数类型	建议的范围
num_layers_encoder	IntegerParameterRange	[1-10]
num_layers_decoder	IntegerParameterRange	[1-10]
batch_size	CategoricalParameterRange	[16,32,64,128,256,512,1024,2048]
optimizer_type	CategoricalParameterRange	['adam', 'sgd', 'rmsprop']
weight_init_type	CategoricalParameterRange	['xavier', 'uniform']
weight_init_scale	ContinuousParameterRange	对于 xavier 类型：MinValue：2.0，MaxValue：3.0 对于制服类型：MinValue：-1.0，MaxValue：1.0
learning_rate	ContinuousParameterRange	MinValue：0.00005，MaxValue：0.2
weight_decay	ContinuousParameterRange	MinValue：0.0，MaxValue：0.1
momentum	ContinuousParameterRange	MinValue：0.5，MaxValue：0.9
clip_gradient	ContinuousParameterRange	MinValue：1.0，MaxValue：5.0

参数名称	参数类型	建议的范围
rnn_num_hidden	CategoricalParameterRange	仅适用于循环神经网络 (RNN)。[128,256,512,1024,2048]
cnn_num_hidden	CategoricalParameterRange	仅适用于卷积神经网络 (CNN)。[128,256,512,1024,2048]
num_embed_source	IntegerParameterRange	[256-512]
num_embed_target	IntegerParameterRange	[256-512]
embed_dropout_source	ContinuousParameterRange	MinValue：0.0，MaxValue：0.5
embed_dropout_target	ContinuousParameterRange	MinValue：0.0，MaxValue：0.5
rnn_decoder_hidden_dropout	ContinuousParameterRange	MinValue：0.0，MaxValue：0.5
cnn_hidden_dropout	ContinuousParameterRange	MinValue：0.0，MaxValue：0.5
lr_scheduler_type	CategoricalParameterRange	['plateau_reduce', 'fixed_rate_inv_t', 'fixed_rate_inv_sqrt_t']
plateau_reduce_lr_factor	ContinuousParameterRange	MinValue：0.1，MaxValue：0.5
plateau_reduce_lr_threshold	IntegerParameterRange	[1-5]
fixed_rate_lr_half_life	IntegerParameterRange	[10-30]

XGBoost 算法

XGBoost (eXtreme Gradient Boosting) 是梯度提升树算法的一种流行且高效的开源实施。梯度提升是一种指导式学习算法，它尝试将一组较简单且较弱模型的一系列估计值结合在一起，从而准确地预测目标变量。XGBoost 算法在机器学习竞赛中表现良好，因为它可以强大地处理各种数据类型、关系、分布以及您可以微调的各种超参数。您可以使用 XGBoost 来处理回归、分类（二进制和多类）和排名问题。

您可以使用新版本的 XGBoost 算法作为 Amazon。 SageMaker 内置算法或作为框架以在本地环境中运行训练脚本。与原始版本相比，此实施占用内存更少，并且具有更好的日志记录、改进的超参数验证以及一组扩展的指标。它提供了一个 XGBoost `estimator`，用于在托管的 XGBoost 环境中执行训练脚本。当前版本 SageMaker XGBoost 基于原始 XGBoost 版本 0.90、1.0、1.2 和 1.3。

支持的版本

- 框架（开源）模式：0.90-1、0.90-2、1.0-1、1.2-1、1.2-2、1.3-1
- 算法模式：0.90-1、0.90-2、1.0-1、1.2-1、1.2-2、1.3-1

Warning

xgBoost 0.90 版本将于 2021 年 12 月 31 日弃用。对 xgBoost 0.90 的安全更新或错误修复的支持将在此日期之后停止。强烈建议在弃用之前将 xgBoost 版本升级到其中一个较新版本（1.0-1 或更高版本）。

Warning

SageMaker Python SDK v1 支持 xgBoost 的最高版本 1.0-1。对于 xgBoost 的更高版本，请使用 SageMaker Python 开发工具包 v2。有关更多信息，请参阅来自的开发工具包 v1 到 SDK v2.

Note

不支持 xgBoost 1.1 SageMaker 因为当测试输入的功能少于 LIBSVM 输入中的训练数据时，xgBoost 1.1 的运行预测功能不佳。该功能已在 xgBoost 1.2 中恢复。考虑使用 SageMaker xgBoost 1.2-2 或更高版本。

如何使用 SageMaker XGBoost

使用 SageMaker，您可以使用 XGBoost 作为内置算法或框架。通过将 XGBoost 用作框架，您可以自定义自己的训练脚本，从而具有更大的灵活性并可以访问更高级的方案（例如 k 折交叉验证）。

- 使用 XGBoost 作为框架

  使用 XGBoost 作为框架来运行可以将附加数据处理合并到训练作业中的自定义训练脚本。在下面的代码示例中，您可以找到如何 SageMaker Python 开发工具包以它提供其他框架 API（如 TensorFlow、MXNet 和 PyTorch）的方式提供 XGBoost API 作为框架。

  SageMaker Python SDK v1

```
import boto3
import sagemaker
from sagemaker.xgboost.estimator import XGBoost
from sagemaker.session import s3_input, Session

# initialize hyperparameters
hyperparameters = {
        "max_depth":"5",
        "eta":"0.2",
        "gamma":"4",
        "min_child_weight":"6",
        "subsample":"0.7",
        "verbosity":"1",
        "objective":"reg:linear",
        "num_round":"50"}

# set an output path where the trained model will be saved
bucket = sagemaker.Session().default_bucket()
prefix = 'DEMO-xgboost-as-a-framework'
output_path = 's3://{}/{}/{}/output'.format(bucket, prefix, 'abalone-xgb-framework')

# construct a SageMaker XGBoost estimator
# specify the entry_point to your xgboost training script
estimator = XGBoost(entry_point = "your_xgboost_abalone_script.py",
                    framework_version='1.0-1',
                    hyperparameters=hyperparameters,
                    role=sagemaker.get_execution_role(),
                    train_instance_count=1,
                    train_instance_type='ml.m5.2xlarge',
                    output_path=output_path)

# define the data type and paths to the training and validation datasets
content_type = "libsvm"
train_input = s3_input("s3://{}/{}/{}/".format(bucket, prefix, 'train'),
 content_type=content_type)
```

```
validation_input = s3_input("s3://{}/{}/{}/".format(bucket, prefix, 'validation'),
 content_type=content_type)

# execute the XGBoost training job
estimator.fit({'train': train_input, 'validation': validation_input})
```

SageMaker Python SDK v2

```
import boto3
import sagemaker
from sagemaker.xgboost.estimator import XGBoost
from sagemaker.session import Session
from sagemaker.inputs import TrainingInput

# initialize hyperparameters
hyperparameters = {
        "max_depth":"5",
        "eta":"0.2",
        "gamma":"4",
        "min_child_weight":"6",
        "subsample":"0.7",
        "verbosity":"1",
        "objective":"reg:linear",
        "num_round":"50"}

# set an output path where the trained model will be saved
bucket = sagemaker.Session().default_bucket()
prefix = 'DEMO-xgboost-as-a-framework'
output_path = 's3://{}/{}/{}/output'.format(bucket, prefix, 'abalone-xgb-framework')

# construct a SageMaker XGBoost estimator
# specify the entry_point to your xgboost training script
estimator = XGBoost(entry_point = "your_xgboost_abalone_script.py",
                    framework_version='1.2-2',
                    hyperparameters=hyperparameters,
                    role=sagemaker.get_execution_role(),
                    instance_count=1,
                    instance_type='ml.m5.2xlarge',
                    output_path=output_path)

# define the data type and paths to the training and validation datasets
content_type = "libsvm"
train_input = TrainingInput("s3://{}/{}/{}/".format(bucket, prefix, 'train'),
 content_type=content_type)
validation_input = TrainingInput("s3://{}/{}/{}/".format(bucket, prefix,
 'validation'), content_type=content_type)

# execute the XGBoost training job
estimator.fit({'train': train_input, 'validation': validation_input})
```

对于 end-to-end 使用示例 SageMaker XGBoost 作为框架，请参阅Amazon 的回归 SageMaker XGBoost

- 使用 XGBoost 作为内置算法

使用 XGBoost 内置算法构建 XGBoost 训练容器，如以下代码示例所示。您可以使用 SageMaker `image_uris.retrieve`API（或`get_image_uri`如果使用 API亚马逊 SageMaker Python 开发工具包版本 1)。如果要确保 `image_uris.retrieve` API 找到正确的 URI，请参阅内置算法的常用参数，并从内置算法映像 URI 和可用区域的完整列表中查找 `xgboost`。

指定 XGBoost 映像 URI 后，您可以使用 XGBoost 容器构建估算程序。 SageMaker 评估程序 API 并启动训练作业。此 XGBoost 内置算法模式不包含您自己的 XGBoost 训练脚本，而是直接在输入数据集上运行。

SageMaker Python SDK v1

```
import sagemaker
import boto3
from sagemaker.amazon.amazon_estimator import get_image_uri
from sagemaker.session import s3_input, Session

# initialize hyperparameters
hyperparameters = {
        "max_depth":"5",
        "eta":"0.2",
        "gamma":"4",
        "min_child_weight":"6",
        "subsample":"0.7",
        "objective":"reg:squarederror",
        "num_round":"50"}

# set an output path where the trained model will be saved
bucket = sagemaker.Session().default_bucket()
prefix = 'DEMO-xgboost-as-a-built-in-algo'
output_path = 's3://{}/{}/{}/output'.format(bucket, prefix, 'abalone-xgb-built-in-
algo')

# this line automatically looks for the XGBoost image URI and builds an XGBoost
 container.
# specify the repo_version depending on your preference.
xgboost_container = get_image_uri(boto3.Session().region_name,
                          'xgboost',
                          repo_version='1.0-1')

# construct a SageMaker estimator that calls the xgboost-container
estimator = sagemaker.estimator.Estimator(image_name=xgboost_container,
                                          hyperparameters=hyperparameters,
                                          role=sagemaker.get_execution_role(),
                                          train_instance_count=1,
                                          train_instance_type='ml.m5.2xlarge',
                                          train_volume_size=5, # 5 GB
                                          output_path=output_path)

# define the data type and paths to the training and validation datasets
content_type = "libsvm"
train_input = s3_input("s3://{}/{}/{}/".format(bucket, prefix, 'train'),
 content_type=content_type)
validation_input = s3_input("s3://{}/{}/{}/".format(bucket, prefix, 'validation'),
 content_type=content_type)

# execute the XGBoost training job
estimator.fit({'train': train_input, 'validation': validation_input})
```

SageMaker Python SDK v2

```
import sagemaker
import boto3
from sagemaker import image_uris
from sagemaker.session import Session
from sagemaker.inputs import TrainingInput

# initialize hyperparameters
hyperparameters = {
        "max_depth":"5",
        "eta":"0.2",
        "gamma":"4",
        "min_child_weight":"6",
```

```
        "subsample":"0.7",
        "objective":"reg:squarederror",
        "num_round":"50"}

# set an output path where the trained model will be saved
bucket = sagemaker.Session().default_bucket()
prefix = 'DEMO-xgboost-as-a-built-in-algo'
output_path = 's3://{}/{}/{}/output'.format(bucket, prefix, 'abalone-xgb-built-in-
algo')

# this line automatically looks for the XGBoost image URI and builds an XGBoost
 container.
# specify the repo_version depending on your preference.
xgboost_container = sagemaker.image_uris.retrieve("xgboost", region, "1.2-2")

# construct a SageMaker estimator that calls the xgboost-container
estimator = sagemaker.estimator.Estimator(image_uri=xgboost_container,
                                          hyperparameters=hyperparameters,
                                          role=sagemaker.get_execution_role(),
                                          instance_count=1,
                                          instance_type='ml.m5.2xlarge',
                                          volume_size=5, # 5 GB
                                          output_path=output_path)

# define the data type and paths to the training and validation datasets
content_type = "libsvm"
train_input = TrainingInput("s3://{}/{}/{}/".format(bucket, prefix, 'train'),
 content_type=content_type)
validation_input = TrainingInput("s3://{}/{}/{}/".format(bucket, prefix,
 'validation'), content_type=content_type)

# execute the XGBoost training job
estimator.fit({'train': train_input, 'validation': validation_input})
```

有关如何将 XGBoost 设置为内置算法的更多信息，请参阅以下笔记本示例。

- 适用于 XGBoost 的托管 Spot 训练
- Amazon 的回归 SageMaker XGBoost（实木复合地板输入）

XGBoost 算法的输入/输出接口

梯度提升对表格数据进行操作，其中行表示观察、一个列表示目标变量或标签，其余列表示特征。

这些区域有： SageMaker 的 XGBoost 实现针对训练和推理支持 CSV 和 libsvm 格式：

- 对于训练 ContentType，有效输入是 text/libsvm（默认值）或 text/csv。
- 对于推理 ContentType，有效的输入是 text/libsvm（默认值）或 text/csv。

> **Note**
>
> 对于 CSV 训练，算法假定目标变量在第一列中，而 CSV 没有标头记录。
> 对于 CSV 推理，算法假定 CSV 输入没有标签列。
> 对于 libsvm 训练，算法假定标签在第一列中。后续的列包含特征的索引值对（从零开始）。因此，每一行的格式为：<label> <index0>:<value0> <index1>:<value1> ... 对于 libsvm 的推理请求可能没有 libsvm 格式的标签。

这与其他不同 SageMaker 算法，它使用 protobuf 训练输入格式来保持与标准 XGBoost 数据格式更高的一致性。

对于 CSV 训练输入模式，可用于算法的总内存 (实例计数 * `InstanceType` 中的可用内存) 必须能够容纳训练数据集。对于 libsvm 训练输入模式，它不是必需的，但我们建议使用它。

SageMaker XGBoost 使用 Python pickle 模块来序列化/反序列化模型，这可用于保存/加载模型。

使用训练过的模型 SageMaker 开源 xgBoost 中的 xgBoost

- 使用以下 Python 代码：

```
import pickle as pkl
import tarfile

t = tarfile.open('model.tar.gz', 'r:gz')
t.extractall()

model = pkl.load(open(model_file_path, 'rb'))

# prediction with test data
pred = model.predict(dtest)
```

使用实例权重支持区分标记数据点的重要性

- SageMaker XGBoost 允许客户通过向各个实例分配权重值来区分标记数据点的重要性。对于 text/libsvm 输入，客户可以通过在标签之后附加权重值，将权重值分配给数据实例。例如，`label:weight idx_0:val_0 idx_1:val_1...`。对于 text/csv 输入，客户需要在参数中打开 `csv_weights` 标志，并将权重值附加到标签之后的列中。例如：`label,weight,val_0,val_1,...`）。

XGBoost 算法的 EC2 实例推荐

SageMaker XGBoost 1.0-1 或更低版本目前仅使用 CPU 进行训练。它是一种内存限制型（而不是计算限制型）算法。因此，通用计算实例（例如 M5）是比计算优化的实例（例如 C4）更适合的选择。此外，我们建议您在选定的实例中有足够的总内存来保存训练数据。尽管它支持使用磁盘空间来处理无法放入主内存的数据（out-of-core 在 libsvm 输入模式下可用的功能），在磁盘上写入缓存文件会减慢算法处理时间。

SageMaker XGBoost 1.2 或更高版本支持单实例 GPU 训练。尽管每实例成本较高，但 GPU 训练的速度更快，因此更经济高效。要利用 GPU 训练，请将实例类型指定为 GPU 实例之一（例如 P3），然后将`tree_method`超参数`gpu_hist`在你现有的 xgBoost 脚本中。 SageMaker 目前不支持多 GPU 训练。

XGBoost 示例笔记本

下表概述了解决亚马逊不同使用案例的各种样本笔记本 SageMaker XGBoost 算法。

笔记本标题	描述
如何创建自定义 xgBoost 容器？	本笔记本向您展示了如何使用亚马逊构建自定义 xgBoost 容器 SageMaker Batch 转换。
使用镶木地板使用 xgBoost 进行回归	本笔记本向您展示了如何在实木复合地板中使用鲍鱼数据集来训练 xgBoost 模型。
如何训练和托管多类分类模型？	本笔记本展示了如何使用 MNIST 数据集训练和托管多类分类模型。
如何训练客户流失预测模型？	本笔记本向您展示了如何训练模型来预测移动客户离境，以确定不满意的客户。
亚马逊简介 SageMaker 用于 xgBoost 培训的托管竞价型基础	本笔记本向您展示了如何使用竞价型实例进行 xgBoost 容器的训练。

笔记本标题	描述
如何使用亚马逊 SageMaker 调试器来调试 xgBoost 培训作业？	本笔记本向您介绍如何使用 Amazon Amazon SageMaker 调试器用于监控训练作业以检测不一致之处。
如何使用亚马逊 SageMaker 调试器来实时调试 xgBoost 培训作业？	本笔记本向您展示了如何在训练作业运行时使用 MNIST 数据集和 Amazon SageMaker 调试器对 xgBoost 训练作业执行实时分析。

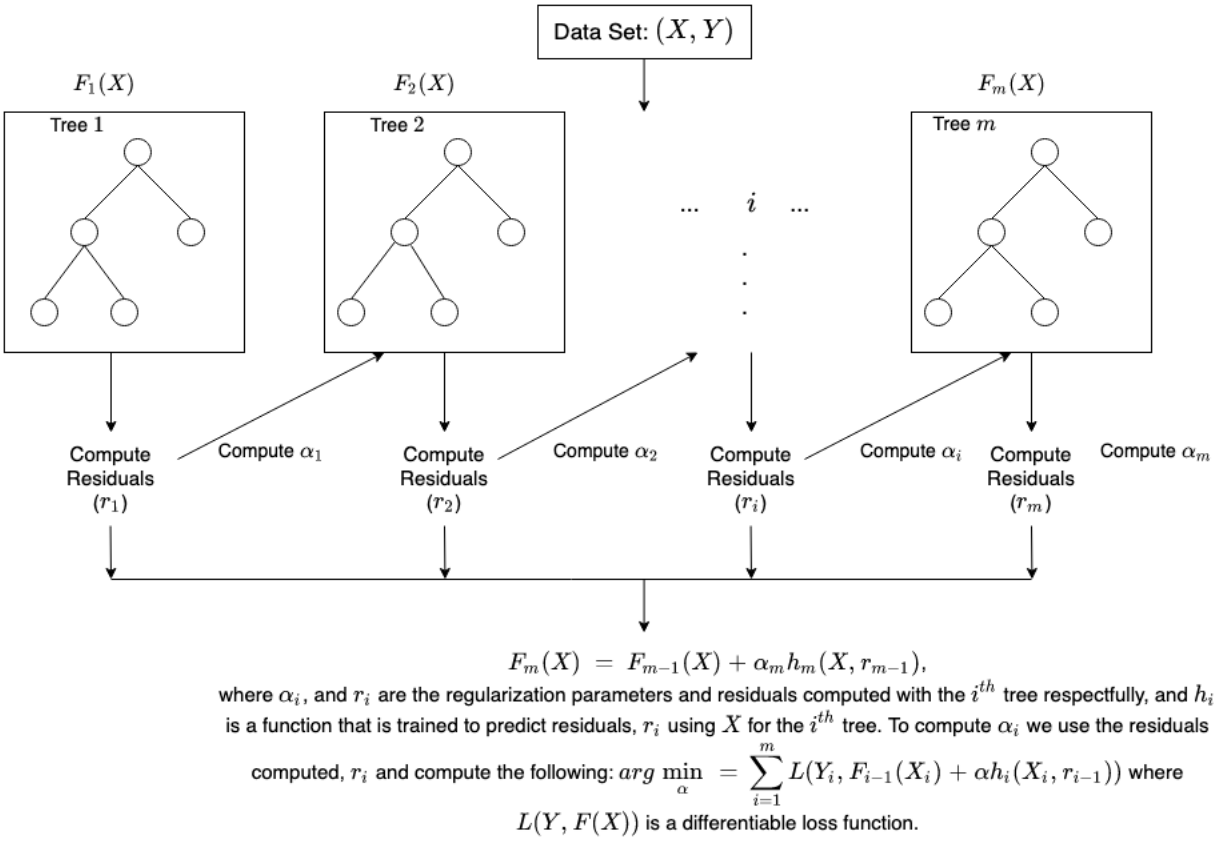
有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 创建笔记本实例并打开该实例后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例。使用线性学习算法的主题建模示例笔记本位于 Amazon 算法简介部分中。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

XGBoost 的工作原理

XGBoost 是梯度提升树算法的一种流行且高效的开源实施。梯度提升是一种指导式学习算法，它尝试将一组较简单、较弱的模型的估计值结合在一起，从而准确地预测目标变量。

当将梯度提升用于回归时，弱学习器是回归树，每个回归树将输入数据点映射到一个包含连续分数的叶中。XGBoost 最小化一个正则化 (L1 和 L2) 目标函数 (根据预测和目标输出的差异)，该函数将凸损失函数和模型复杂度的惩罚项 (换言之，回归树函数) 组合起来。训练以迭代的方式进行，从而添加新树来预测先前树的残差或错误，然后再与先前树结合，做出最后的预测。这称为梯度提升，因为它使用梯度下降算法来最小化添加新模型时的损失。

下面是关于梯度树增强工作原理的简要说明。



有关 XGBoost 的更多详细信息，请参阅：

- XGBoost：可扩展树提升系统
- 提升梯度树
- 提升树简介

XGBoost 超参数

下表包含 Amazon Amazon 必需或最常用的超参数子集。 SageMaker XGBoost 算法。这些是由用户设置的参数，以便于从数据中评估模型参数。首先，按字母顺序列出必须设置的所需超参数。接下来，也按字母顺序列出可以设置的可选超参数。SageMaker XGBoost 算法是开源 DMLC XGBoost 程序包的实施。目前 SageMaker 支持版本 1.2-2。有关可以为此版本的 XGBoost 配置的完整超参数集的详细信息，请参阅 XGBoost 参数。

参数名称	说明
num_class	类的数量。 必需如果objective设置为多:softmax要么多:softprob. 有效值：整数
num_round	运行训练的轮数。 必填 有效值：整数
alpha	权重上的 L1 正则化项。增加此值会使模型更加保守。 可选 有效值：浮点值 默认值：0
base_score	所有实例的初始预测分数，全局偏移。 可选 有效值：浮点值 默认值：0.5
booster	要使用的助推程序。gbtree 和 dart 值使用基于树的模型，而 gblinear 使用线性函数。 可选 有效值：字符串。gbtree、gblinear 或 dart。 默认值：gbtree
colsample_bylevel	每个级别中每个拆分的列的子样本比率。 可选 有效值：浮点。Range (范围)：[0,1]。

参数名称	说明
	默认值：1
colsample_bynode	每个节点中列的子样本比率。 可选 有效值：浮点。Range (范围)：(0,1]。 默认值：1
colsample_bytree	构造每个树时列的子样本比率。 可选 有效值：浮点。Range (范围)：[0,1]。 默认值：1
csv_weights	启用此标志时，对于 csv 输入，XGBoost 通过获取训练数据中的第二列（标签后的一列）作为实例权重，以此来区分实例的重要性。 可选 有效值：0 或 1 默认值：0
deterministic_histogram	启用此标志后，xgBoost 会确定地在 GPU 上构建直方图。仅在 tree_method 设置为 gpu_hist 时才会使用。 有关有效输入的完整列表，请参阅 XGBoost 参数。 可选 有效值：字符串。范围：true 或 false 默认值：true
early_stopping_rounds	模型会一直训练直到验证分数停止改善。验证错误至少需要每减少early_stopping_rounds继续训练。SageMaker 托管使用最佳模型进行推理。 可选 有效值：整数 默认值: -
eta	在更新中用于防止过度适合的步骤大小收缩。在每个提升步骤之后，您可以直接获得新特征的权重。eta 参数实际上缩小了特征权重，使提升过程更加保守。 可选 有效值：浮点。Range (范围)：[0,1]。 默认值：0.3

参数名称	说明
eval_metric	验证数据的评估指标。根据目标分配默认指标： • rmse：用于回归 • error：用于分类 • map：用于排名 有关有效输入的列表，请参阅xgBoost 学习任务参数. 可选 有效值：字符串 默认值：根据目标默认。
gamma	在树的叶节点上进行进一步分区所需的最小损失减少。该值越大，算法越保守。 可选 有效值：浮点。Range (范围)：[0, ∞)。 默认值：0
grow_policy	控制将新节点添加到树中的方式。目前，仅在 tree_method 设置为 hist 时受支持。 可选 有效值：字符串。depthwise 或 lossguide。 默认值：depthwise
interaction_constraints	指定允许进行交互的变量组。 可选 有效值：嵌套的整数列表。每个整数代表一个要素，每个嵌套列表都包含允许交互的功能，例如 [[1,2]、[3,4,5]]。 默认值：None（无）
lambda	权重上的 L2 正则化项。增加此值会使模型更加保守。 可选 有效值：浮点值 默认值：1
lambda_bias	偏移上的 L2 正则化项。 可选 有效值：浮点。Range (范围)：[0.0, 1.0]。 默认值：0

参数名称	说明
max_bin	离散分箱到存储桶连续特征的最大数量。仅在 tree_method 设置为 hist 时才会使用。 可选 有效值：整数 默认值：256
max_delta_step	每个树的权重估计允许的最大增量步骤。当使用正整数时，它有助于使更新更加保守。首选选项是在逻辑回归中使用它。将其设置为 1-10 可帮助控制更新。 可选 有效值：整数。Range (范围)：[0, ∞)。 默认值：0
max_depth	树的最大深度。增加此值会使模型更复杂，并可能会过度拟合。0 表示无限制。当 grow_policy=depth-wise 时，需要限制。 可选 有效值：整数。Range (范围)：[0, ∞) 默认值：6
max_leaves	要添加的最大节点数。仅当 grow_policy 设为 lossguide 时相关。 可选 有效值：整数 默认值：0
min_child_weight	子项中所需的最小实例权重总和 (hessian)。如果树分区步骤导致叶节点的实例权重总和小于 min_child_weight 的总和，则生成过程会放弃进一步的分区。在线性回归模型中，这只是对应于每个节点中所需的最小实例数。算法越大，就越保守。 可选 有效值：浮点。Range (范围)：[0, ∞)。 默认值：1

参数名称	说明
monotone_constraints	指定对任何要素的单调性约束。 可选 有效值：整数元组。有效整数 #1（递减约束）、0（无约束）、1（增加约束）。 例如，(0, 1)：对第一个预测变量没有约束，对第二个预测变量的约束越来越大。(-1, 1): 减少第一个预测变量的约束，对第二个预测变量的约束越来越大。 默认值：(0, 0)
normalize_type	标准化算法的类型。 可选 有效值：或者树要么森林. 默认值：tree
nthread	用于运行 xgboost 的并行线程数量。 可选 有效值：整数 默认值：最大线程数。
objective	指定学习任务和相应的学习目标。示例：reg:logistic、multi:softmax、reg:squarederror. 有关有效输入的完整列表，请参阅xgBoost 学习任务参数. 可选 有效值：字符串 默认值：reg:squarederror
one_drop	启用此标记后，在丢弃期间至少会丢弃一个树。 可选 有效值：0 或 1 默认值：0
process_type	要运行的提升过程的类型。 可选 有效值：字符串。default 或 update。 默认值：default

参数名称	说明
rate_drop	丢弃比率，指定在丢弃期间要丢弃的一小部分以前的树。 可选 有效值：浮点。Range (范围)：[0.0, 1.0]。 默认值：0.0
refresh_leaf	这是“刷新”更新者插件的参数。当设置为 true (1) 时，树叶和树节点统计数据会更新。当设置为 false (0) 时，只有树节点统计数据会更新。 可选 有效值：0/1 默认值：1
sample_type	采样算法的类型。 可选 有效值：uniform 或 weighted。 默认值：uniform
scale_pos_weight	控制正负权重的平衡。它对不平衡的类很有用。可考虑的典型值：sum(negative cases) / sum(positive cases)。 可选 有效值：浮点值 默认值：1
seed	随机数种子。 可选 有效值：整数 默认值：0
single_precision_histogram	启用此标志后，xgBoost 使用单精度来构建直方图而不是双精度。仅当时使用tree_method设置为hist要么gpu_hist. 有关有效输入的完整列表，请参阅 XGBoost 参数。 可选 有效值：字符串。范围：true 或 false 默认值：false

参数名称	说明
sketch_eps	仅用于近似贪婪算法。这会转换为 O(1 / sketch_eps) 分箱数。与直接选择分箱数相比，这附带了具有草图准确率的理论保证。 可选 有效值：Float (浮点范围) [0, 1]。 默认值：0.03
skip_drop	在提升迭代期间跳过丢弃过程的概率。 可选 有效值：浮点。Range (范围)：[0.0, 1.0]。 默认值：0.0
subsample	训练实例的子样本比率。将其设置为 0.5 意味着 XGBoost 随机收集一半数据实例来让树长大。这会防止过度适合 可选 有效值：浮点。Range (范围)：[0,1]。 默认值：1
tree_method	在 XGBoost 中使用的树构造算法。 可选 有效值：以下值之一：auto、exact、approx、hist 或 gpu_hist。 默认值：auto
tweedie_variance_power	控制 Tweedie 分布的方差的参数。 可选 有效值：浮点。Range (范围)：(1, 2)。 默认值：1.5
updater	一个逗号分隔的字符串，它定义要运行的树更新者序列。这提供了构造和修改树的模块化方法。 有关有效输入的完整列表，请参阅 XGBoost 参数。 可选 有效值：逗号分隔的字符串。 默认值：grow_colmaker、prune

参数名称	说明
verbosity	打印消息的详细程度。 有效值：0（静默）、1（警告）、2（信息）、3（调试）。 可选 默认值：1

优化 XGBoost 模型

自动调整模型也称为超参数优化，它通过运行很多在训练和验证数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择三种类型的超参数：

- 学习objective在模型训练期间优化的函数
- 一个eval_metric用于在验证期间评估模型性能
- 自动调整模型时要使用的一组超参数和一系列值，供每个参数使用

您可以从算法计算的一组评估指标中选择评估指标。自动模型优化将搜索所选超参数以找到导致优化评估指标的模型的值组合。

> **Note**
>
> XGBoost 0.90 的自动模型优化仅可从亚马逊获得 SageMaker SDK，而不是来自 SageMaker 控制台。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

由 XGBoost 算法计算的评估指标

XGBoost 算法计算以下指标以用于模型验证。在优化模型时，请从这些指标中选择一个指标来评估模型。有效的完整列表eval_metric值，请参阅xgBoost 学习任务参数

指标名称	说明	优化方向
validation:accuracy	分类速率，计算方式为正确用例数/所有用例数。	最大化
validation:auc	曲线下方的区域。	最大化
validation:error	二进制分类错误率，计算方式为错误用例数/所有用例数。	最小化
validation:f1	分类准确率指标，计算方式为查准率和查全率的调和平均值。	最大化
validation:logloss	负对数似然。	最小化
validation:mae	平均绝对误差。	最小化
validation:map	平均查准率。	最大化
validation:merror	多类别分类错误率，计算方式为错误用例数/所有用例数。	最小化
validation:mlogloss	多类别分类的负对数似然。	最小化

指标名称	说明	优化方向
validation:mse	均方差。	最小化
validation:ndcg	标准化折扣累积收益。	最大化
validation:rmse	均方根误差。	最小化

可优化 XGBoost 超参数

使用以下超参数优化 XGBoost 模型。对优化 XGBoost 评估指标影响最大的超参数包括：alpha、min_child_weight、subsample、eta, 和num_round.

参数名称	参数类型	建议的范围
alpha	ContinuousParameterRanges	MinValue：0，MaxValue：1000
colsample_bylevel	ContinuousParameterRanges	MinValue：0.1，MaxValue：1
colsample_bynode	ContinuousParameterRanges	MinValue：0.1，MaxValue：1
colsample_bytree	ContinuousParameterRanges	MinValue：0.5，MaxValue：1
eta	ContinuousParameterRanges	MinValue：0.1，MaxValue：0.5
gamma	ContinuousParameterRanges	MinValue：0，MaxValue：5
lambda	ContinuousParameterRanges	MinValue：0，MaxValue：1000
max_delta_step	IntegerParameterRanges	[0, 10]
max_depth	IntegerParameterRanges	[0, 10]
min_child_weight	ContinuousParameterRanges	MinValue：0，MaxValue：120
num_round	IntegerParameterRanges	[1, 4000]
subsample	ContinuousParameterRanges	MinValue：0.5，MaxValue：1

xgBoost 的弃用版本及其升级

本主题包含有关 Amazon SageMaker XGBoost 的早期版本的文档，这些文档仍然可用但已弃用。它还提供了有关如何在可能的情况下将 XGBoost 的弃用版本升级到更新的版本的说明。

主题

- 将 xgBoost 版本 0.90 升级到 1.2 版 (p. 1747)
- XGBoost 版本 0.72 (p. 1748)

将 xgBoost 版本 0.90 升级到 1.2 版

如果您使用的是 SageMaker Python SDK，要将现有 xgBoost 0.90 作业升级到 1.2 版，则必须安装 2.x 版本的 SDK 并更改 xgBoost`version`和`framework_version`参数为 1.2-2。如果你使用的是 Boto3，你需要更新 Docker 镜像以及一些超参数和学习目标。

主题

- 将 SageMaker Python SDK 1.x 版本升级到 2.x 版 (p. 1747)
- 将图片标签更改为 1.2-2 (p. 1747)
- 更改 Boto3 的 Docker 映像 (p. 1748)
- 更新超参数和学习目标 (p. 1748)

将 SageMaker Python SDK 1.x 版本升级到 2.x 版

如果您仍在使用 SageMaker Python SDK 的 1.x 版本，则必须升级 SageMaker Python SDK 的 2.x 版本。有关最新版本的 SageMaker Python SDK 的信息，请参阅使用版本的 SageMaker Python 开发工具包. 要安装最新版本，请运行：

```
python -m pip install --upgrade sagemaker
```

将图片标签更改为 1.2-2

如果您使用的是 SageMaker Python SDK 并使用 xgBoost 内置算法，请在中更改版本参数`image_uris.retrive`.

```
from sagemaker import image_uris
image_uris.retrieve(framework="xgboost", region="us-west-2", version="1.2-2")

estimator = sagemaker.estimator.Estimator(image_uri=xgboost_container,
                                          hyperparameters=hyperparameters,
                                          role=sagemaker.get_execution_role(),
                                          instance_count=1,
                                          instance_type='ml.m5.2xlarge',
                                          train_volume_size=5, # 5 GB
                                          output_path=output_path)
```

如果您使用 SageMaker Python SDK 并使用 XGBoost 作为框架来运行自定义训练脚本，请更改`framework_version`xgBoost API 中的参数。

```
estimator = XGBoost(entry_point = "your_xgboost_abalone_script.py",
                    framework_version='1.2-2',
                    hyperparameters=hyperparameters,
                    role=sagemaker.get_execution_role(),
                    instance_count=1,
                    instance_type='ml.m5.2xlarge',
                    output_path=output_path)
```

`sagemaker.session.s3_input`在 SageMaker Python SDK 版本 1.x 中已重命名为`sagemaker.inputs.TrainingInput`. 您必须使用`sagemaker.inputs.TrainingInput`如以下示例所示。

```
content_type = "libsvm"
train_input = TrainingInput("s3://{}/{}/{}/".format(bucket, prefix, 'train'),
 content_type=content_type)
validation_input = TrainingInput("s3://{}/{}/{}/".format(bucket, prefix, 'validation'),
 content_type=content_type)
```

有关 SageMaker Python SDK 版本 2.x 更改的完整列表，请参阅使用版本的 SageMaker Python 开发工具包.

更改 Boto3 的 Docker 映像

如果您使用 Boto3 训练或部署模型，请将 docker 图像标签（0.90-1 或 0.90-2）更改为 1.2-2。

```
{
    "AlgorithmSpecification":: {
        "TrainingImage": "746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-
xgboost:1.2-2"
    }
    ...
}
```

如果您使用 SageMaker Python SDK 检索注册表路径，请更改`version`中的参数`image_uris.retrieve`.

```
from sagemaker import image_uris
image_uris.retrieve(framework="xgboost", region="us-west-2", version="1.2-2")
```

更新超参数和学习目标

静默参数已弃用，在 XGBoost 1.2 及更高版本中不可用。请改用 `verbosity`。如果你正在使用`reg:linear`学习目标，它已被弃用也有利于 `reg:squarederror`. 请改用 `reg:squarederror`。

```
hyperparameters = {
    "verbosity": "2",
    "objective": "reg:squarederror",
    "num_round": "50",
    ...
}

estimator = sagemaker.estimator.Estimator(image_uri=xgboost_container,
                                          hyperparameters=hyperparameters,
                                          ...)
```

XGBoost 版本 0.72

Important

XgBoost 0.72 已被 Amazon SageMaker 弃用。您仍然可以通过拉取 XgBoost 的图像 URI 来使用这个旧版本的 xgBoost（作为内置算法），如下面的代码示例所示。对于 xgBoost，图像 URI 以`:1`适用于旧版本。

SageMaker Python SDK v1

```
import boto3
from sagemaker.amazon.amazon_estimator import get_image_uri

xgb_image_uri = get_image_uri(boto3.Session().region_name, "xgboost",
 repo_version="1")
```

SageMaker Python SDK v2

```
import boto3
from sagemaker import image_uris

xgb_image_uri = image_uris.retrieve("xgboost", boto3.Session().region_name, "1")
```

如果你想使用较新的版本，你必须明确指定图像 URI 标签（请参阅支持的版本 (p. 1731)）。

Amazon SageMaker XGBoost 算法的此早期版本基于 0.72 版本。XGBoost (eXtreme Gradient Boosting) 是梯度提升树算法的一种流行且高效的开源实施。梯度提升是一种指导式学习算法，它尝试将一组较简单、较弱的模型的估计值结合在一起，从而准确地预测目标变量。XGBoost 在机器学习竞赛中表现出色，因为它能够强健处理各种数据类型、关系、分布以及由于大量的超参数（可进行优化和调整来改进拟合）。这种灵活性使 XGBoost 成为回归、分类 (二进制和多类) 和排名问题的坚实选择。

客户应考虑使用 XGBoost 算法 (p. 1731) 的新版本。他们可以将其用作 SageMaker 内置算法或者用作框架在其本地环境中运行脚本，例如，他们通常使用 Tensorflow 深度学习框架执行操作。新实施占用内存更少，并且具有更好的日志记录、改进的超参数验证以及一组扩展的指标。如果客户需要推迟迁移到新版本，则以前的 XGBoost 实施仍可供客户使用。但是之前的此实施仍然与 XGBoost 的 0.72 版本保持关联。

XGBoost 算法的输入/输出接口

梯度提升对表格数据进行操作，其中行表示观察、一个列表示目标变量或标签，其余列表示特征。

SageMaker 的 XGBoost 实现针对训练和推理支持 CSV 和 libsvm 格式：

- 对于训练 ContentType，有效输入是 text/libsvm（默认值）或 text/csv。
- 对于推理 ContentType，有效输入是 text/libsvm 或（默认值）text/csv。

> **Note**
>
> 对于 CSV 训练，算法假定目标变量在第一列中，而 CSV 没有标头记录。对于 CSV 推理，算法假定 CSV 输入没有标签列。
> 对于 libsvm 训练，算法假定标签在第一列中。后续的列包含特征的索引值对（从零开始）。因此，每一行的格式为：<label> <index0>:<value0> <index1>:<value1> ... libsvm 的推理请求可能有也可能没有 libsvm 格式的标签。

这与其他 SageMaker 算法不同，其他算法使用 protobuf 训练输入格式来保持与标准 XGBoost 数据格式更高的一致性。

对于 CSV 训练输入模式，可用于算法的总内存 (实例计数 * `InstanceType` 中的可用内存) 必须能够容纳训练数据集。对于 libsvm 训练输入模式，它不是必需的，但我们建议使用它。

SageMaker XGBoost 使用 Python pickle 模块来序列化/反序列化模型，这可用于保存/加载模型。

在开源 XGBoost 中使用通过 SageMaker XGBoost 训练的模型

- 使用以下 Python 代码：

```
import pickle as pkl
import tarfile
import xgboost

t = tarfile.open('model.tar.gz', 'r:gz')
t.extractall()

model = pkl.load(open(model_file_path, 'rb'))

# prediction with test data
pred = model.predict(dtest)
```

使用实例权重支持区分标记数据点的重要性

- SageMaker XGBoost 允许客户通过向各个实例分配权重值来区分标记数据点的重要性。对于 text/libsvm 输入，客户可以通过在标签之后附加权重值，将权重值分配给数据实例。例如：`label:weight idx_0:val_0 idx_1:val_1...`。对于 text/csv 输入，客户需要在参数中打开 `csv_weights` 标志，并将权重值附加到标签之后的列中。例如：`label,weight,val_0,val_1,...`）。

XGBoost 版本 0.72 的 EC2 实例推荐

SageMaker XGBoost 目前仅使用 CPU 进行训练。它是一种内存限制型（而不是计算限制型）算法。因此，一般用途的计算实例 (例如 M4) 是比计算优化的实例 (例如 C4) 更适合的选择。此外，我们建议您在选定的实例中有足够的总内存来保存训练数据。尽管它支持使用磁盘空间来处理无法放入主内存的数据 (libsvm 输入模式下可用的内核外功能)，但在磁盘上写入缓存文件会减慢算法处理时间。

XGBoost 版本 0.72 示例笔记本

有关如何使用最新版本的 SageMaker XGBoost 作为内置算法来训练和托管回归模型的示例笔记本，请参阅。使用 Amazon SageMaker XGBoost 算法进行回归. 要使用 0.72 版本的 XGBoost，您需要将示例代码中的版本更改为 0.72。有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 在您创建笔记本实例并打开该实例之后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。使用 XGBoost 算法的主题建模示例笔记本位于 Amazon 算法简介部分中。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

XGBoost 版本 0.72 超参数

下表包含 XGBoost 算法的超参数。这些是由用户设置的参数，以便于从数据中评估模型参数。首先，按字母顺序列出必须设置的所需超参数。接下来，也按字母顺序列出可以设置的可选超参数。SageMaker XGBoost 算法是开源 XGBoost 程序包的实现。目前 SageMaker 支持版本 0.72。有关此 XGBoost 版本的超参数配置的详细信息，请参阅 XGBoost 参数。

参数名称	描述
num_class	类的数量。 必需如果objective设置为多:softmax要么多:softprob. 有效值：整数
num_round	运行训练的轮数。 必填 有效值：整数
alpha	权重上的 L1 正则化项。增加此值会使模型更加保守。 可选 有效值：浮点值 默认值：0
base_score	所有实例的初始预测分数，全局偏移。 可选 有效值：浮点值 默认值：0.5
booster	要使用的助推程序。gbtree 和 dart 值使用基于树的模型，而 gblinear 使用线性函数。 可选 有效值：字符串。gbtree、gblinear 或 dart。

参数名称	描述
	默认值：gbtree
colsample_bylevel	每个级别中每个拆分的列的子样本比率。 可选 有效值：浮点。Range (范围)：[0,1]。 默认值：1
colsample_bytree	构造每个树时列的子样本比率。 可选 有效值：浮点。Range (范围)：[0,1]。 默认值：1
csv_weights	启用此标志时，对于 csv 输入，XGBoost 通过获取训练数据中的第二列（标签后的一列）作为实例权重，以此来区分实例的重要性。 可选 有效值：0 或 1 默认值：0
early_stopping_rounds	模型会一直训练直到验证分数停止改善。至少需要每减少验证错误early_stopping_rounds继续训练。SageMaker 托管使用最佳模型进行推理。 可选 有效值：整数 默认值: -
eta	在更新中用于防止过度适合的步骤大小收缩。在每个提升步骤之后，您可以直接获得新特征的权重。eta 参数实际上缩小了特征权重，使提升过程更加保守。 可选 有效值：浮点。Range (范围)：[0,1]。 默认值：0.3

参数名称	描述
eval_metric	验证数据的评估指标。根据目标分配默认指标： • rmse：用于回归 • error：用于分类 • map：用于排名 有关有效输入的列表，请参阅 XGBoost 参数。 可选 有效值：字符串 默认值：默认根据目标进行默认。
gamma	在树的叶节点上进行进一步分区所需的最小损失减少。该值越大，算法越保守。 可选 有效值：浮点。Range (范围)：[0, ∞)。 默认值：0
grow_policy	控制将新节点添加到树中的方式。目前，仅在 tree_method 设置为 hist 时受支持。 可选 有效值：字符串。depthwise 或 lossguide。 默认值：depthwise
lambda	权重上的 L2 正则化项。增加此值会使模型更加保守。 可选 有效值：浮点值 默认值：1
lambda_bias	偏移上的 L2 正则化项。 可选 有效值：浮点。Range (范围)：[0.0, 1.0]。 默认值：0
max_bin	离散分箱到存储桶连续特征的最大数量。仅在 tree_method 设置为 hist 时才会使用。 可选 有效值：整数 默认值：256

参数名称	描述
max_delta_step	每个树的权重估计允许的最大增量步骤。当使用正整数时，它有助于使更新更加保守。首选选项是在逻辑回归中使用它。将其设置为 1-10 可帮助控制更新。 可选 有效值：整数。Range (范围)：[0, ∞)。 默认值：0
max_depth	树的最大深度。增加此值会使模型更复杂，并可能会过度拟合。当 grow_policy=depth-wise 时，需要限制。 可选 有效值：整数。Range (范围)：[0, ∞) 默认值：6
max_leaves	要添加的最大节点数。仅当 grow_policy 设为 lossguide 时相关。 可选 有效值：整数 默认值：0
min_child_weight	子项中所需的最小实例权重总和 (hessian)。如果树分区步骤导致叶节点的实例权重总和小于 min_child_weight 的总和，则生成过程会放弃进一步的分区。在线性回归模型中，这只是对应于每个节点中所需的最小实例数。算法越大，就越保守。 可选 有效值：浮点。Range (范围)：[0, ∞)。 默认值：1
normalize_type	标准化算法的类型。 可选 有效值：或者树要么森林. 默认值：tree
nthread	用于运行 xgboost 的并行线程数量。 可选 有效值：整数 默认值：最大线程数。

参数名称	描述
objective	指定学习任务和相应的学习目标。示例：reg:logistic、reg:softmax、multi:squarederror. 有关有效输入的完整列表，请参阅XGBoost 参数. 可选 有效值：字符串 默认值：reg:squarederror
one_drop	启用此标记后，在丢弃期间至少会丢弃一个树。 可选 有效值：0 或 1 默认值：0
process_type	要运行的提升过程的类型。 可选 有效值：字符串。default 或 update。 默认值：default
rate_drop	丢弃比率，指定在丢弃期间要丢弃的一小部分以前的树。 可选 有效值：浮点。Range (范围)：[0.0, 1.0]。 默认值：0.0
refresh_leaf	这是“刷新”更新者插件的参数。当设置为 true (1) 时，树叶和树节点统计数据会更新。当设置为 false (0) 时，只有树节点统计数据会更新。 可选 有效值：0/1 默认值：1
sample_type	采样算法的类型。 可选 有效值：uniform 或 weighted。 默认值：uniform

参数名称	描述
scale_pos_weight	控制正负权重的平衡。它对不平衡的类很有用。可考虑的典型值：sum(negative cases) / sum(positive cases)。 可选 有效值：浮点值 默认值：1
seed	随机数种子。 可选 有效值：整数 默认值：0
silent	0 表示打印运行消息，1 表示静默模式。 有效值：0 或 1 可选 默认值：0
sketch_eps	仅用于近似贪婪算法。这会转换为 O(1 / sketch_eps) 分箱数。与直接选择分箱数相比，这附带了具有草图准确率的理论保证。 可选 有效值：Float, 范围：[0, 1]。 默认值：0.03
skip_drop	在提升迭代期间跳过丢弃过程的概率。 可选 有效值：浮点。Range (范围)：[0.0, 1.0]。 默认值：0.0
subsample	训练实例的子样本比率。将其设置为 0.5 意味着 XGBoost 随机收集一半数据实例来让树长大。这会防止过度适合 可选 有效值：浮点。Range (范围)：[0,1]。 默认值：1
tree_method	在 XGBoost 中使用的树构造算法。 可选 有效值：以下值之一：auto、exact、approx 或 hist。 默认值：auto

参数名称	描述
tweedie_variance_power	控制 Tweedie 分布的方差的参数。 可选 有效值：浮点。Range (范围)：(1, 2)。 默认值：1.5
updater	一个逗号分隔的字符串，它定义要运行的树更新者序列。这提供了构造和修改树的模块化方法。 有关有效输入的完整列表，请参阅 XGBoost 参数。 可选 有效值：逗号分隔的字符串。 默认值：grow_colmaker、prune

调整 XGBoost 版本 0.72 模型

自动调整模型（也称作超参数优化）通过运行很多在训练和验证数据集上测试一系列超参数的作业来查找模型的最佳版本。您可以选择三种类型的超参数：

- 学习objective在模型训练期间优化的函数
- 一个eval_metric用于在验证过程中评估模型性能
- 自动调整模型时要使用的一组超参数和一系列值，供每个参数使用

您可以从算法计算的一组评估指标中选择评估指标。自动模型优化将搜索所选超参数以找到导致优化评估指标的模型的值组合。

有关模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

由 XGBoost 版本 0.72 算法计算的指标

基于版本 0.72 的 XGBoost 算法计算以下九个指标来用于模型验证。在优化模型时，选择这些指标之一来评估模型。有效的完整列表eval_metric值，请参阅xgBoost 学习任务参数

指标名称	描述	优化方向
validation:auc	曲线下方的区域。	最大化
validation:error	二进制分类错误率，计算方式为错误用例数/所有用例数。	最小化
validation:logloss	负对数似然。	最小化
validation:mae	平均绝对误差。	最小化
validation:map	平均查准率。	最大化
validation:merror	多类别分类错误率，计算方式为错误用例数/所有用例数。	最小化
validation:mlogloss	多类别分类的负对数似然。	最小化

指标名称	描述	优化方向
validation:ndcg	标准化折扣累积收益。	最大化
validation:rmse	均方根误差。	最小化

可优化 XGBoost 版本 0.72 超参数

使用以下超参数优化 XGBoost 模型。对优化 XGBoost 评估指标影响最大的超参数包括：alpha、min_child_weight、subsample、eta, 和num_round.

参数名称	参数类型	建议的范围
alpha	ContinuousParameterRanges	MinValue：0，MaxValue：1000
colsample_bylevel	ContinuousParameterRanges	MinValue：0.1，MaxValue：1
colsample_bytree	ContinuousParameterRanges	MinValue：0.5，MaxValue：1
eta	ContinuousParameterRanges	MinValue：0.1，MaxValue：0.5
gamma	ContinuousParameterRanges	MinValue：0，MaxValue：5
lambda	ContinuousParameterRanges	MinValue：0，MaxValue：1000
max_delta_step	IntegerParameterRanges	[0, 10]
max_depth	IntegerParameterRanges	[0, 10]
min_child_weight	ContinuousParameterRanges	MinValue：0，MaxValue：120
num_round	IntegerParameterRanges	[1, 4000]
subsample	ContinuousParameterRanges	MinValue：0.5，MaxValue：1

Amazon SageMaker 将强化学习结合使用

强化学习（RL）结合了计算机科学、神经科学和心理学等领域，以确定如何将情况与行动相映射，以最大限度地提高数字奖励信号。RL 中的这种奖励信号的概念源于神经科学研究，研究了人类大脑如何决定哪些行动最大限度地提高奖励和最大限度地减少惩罚。在大多数情况下，人类没有得到明确的指示，说明要采取哪些行动，而是必须了解哪些行动可以产生最直接的回报，以及这些行动如何影响未来的局势和后果。

RL 问题是使用源于动态系统理论的马尔可夫决策过程（MDP）正式化的。MDP 旨在获取学习代理在试图实现某个最终目标过程中遇到的一段时间内遇到的真实问题的高级详细信息。学习代理应能够确定其环境的当前状态，并确定影响学习座席当前状态的可能操作。此外，学习机构的目标应该与环境状况密切相关。以这种方式制定的问题的解决方案被称为强化学习方法。

强化、监督和无监督学习范式之间有什么区别？

机器学习可以分为三种不同的学习范式：监督、无监督和强化。

在监督学习中，外部主管提供了一组带标签的示例的培训。每个示例都包含有关某种情况的信息，属于某个类别，并有一个标签标识它所属的类别。监督学习的目标是进行概括化，以便在训练数据中没有的情况下正确预测。

相比之下，RL 处理的是交互式问题，因此无法收集代理人可能遇到的具有正确标签的所有可能情况示例。当代理人能够准确地从自己的经验中学习并进行相应调整时，这种类型的学习就是最有希望的。

在无监督学习中，座席通过发现未标记数据中的结构来学习。尽管 RL 代理商可能会从根据其经验发现结构中受益，但 RL 的唯一目的是最大限度地发挥奖励信号。

使用亚马逊管理 Machine Learning SageMaker 实验

亚马逊 SageMaker 实验是亚马逊的一项功能 SageMaker 您可以组织、跟踪、比较和评估机器学习实验。

机器学习是一个迭代过程。您需要试验数据、算法和参数的各种组合，同时观察增量变化对模型准确性的影响。随着时间的推移，这种迭代实验可能会导致成千上万的模型训练运行和产生不同的模型版本，从而难以跟踪性能最佳的模型及其输入配置。您也很难将现行的实验与过去的实验进行比较，以找出进一步改进的机会。

SageMaker 实验会以如下方式自动跟踪所有迭代的输入、参数、配置和结果：审判. 您可以将这些试验分配、分组和组织到实验. SageMaker 实验与亚马逊集成 SageMaker Studio 提供可视界面，以浏览您的现行实验和过去的实验，比较试验的关键性能指标，并确定性能最出色的实验。

SageMaker 实验附带自己的实验 Python SDK这使分析功能可供在亚马逊轻松使用 SageMaker 笔记本。由于 SageMaker 实验可以跟踪创建模型所需的所有步骤和构件，在对生产环境中的问题进行故障排除或审计模型以进行合规性验证时，可以快速重新访问模型的源。

主题

- SageMaker 实验功能 (p. 1758)
- 创建卓越亚马逊 SageMaker 试验 (p. 1759)
- 查看和比较亚马逊 SageMaker 查看实验、试验和试验组件 (p. 1763)
- 跟踪和比较教程 (p. 1766)
- 使用亚马逊搜索实验 SageMaker 工作室 (p. 1772)
- 清理 Amazon SageMaker 实验资源 (p. 1776)
- 使用亚马逊搜索 SageMaker 控制台和 API (p. 1778)

SageMaker 实验功能

以下部分简要概述了 SageMaker 实验提供的功能。

主题

- 组织实验 (p. 1759)
- 跟踪实验 (p. 1759)
- 比较和评估实验 (p. 1759)
- 亚马逊 SageMaker Autopilot (p. 1759)

组织实验

亚马逊 SageMaker 实验提供了一个结构化的组织方案，帮助用户对其机器学习迭代进行分组和组织。顶级实体（实验）是作为一个组进行观察、比较和评估的试验 集合。试验是一组称为试验组件 的步骤。每个试验组件都可以包含数据集、算法和参数等输入的组合，并生成模型、指标、数据集和检查点等特定输出。试验组件示例包括数据预处理作业、训练作业和批量转换作业。

实验目的是确定生成最佳模型的试验。我们将执行多个试验，每个试验都会隔离和衡量更改一个或多个输入而保持其余输入不变的影响。通过分析试验，您可以确定哪些功能对模型的影响最大。

跟踪实验

亚马逊 SageMaker 通过 实验可以跟踪实验。

自动跟踪

SageMaker 实验自动跟踪亚马逊 SageMaker Autopilot 作业作业作为实验，将其基础训练作业作业作为试验跟踪。 SageMaker 实验还会自动跟踪 SageMaker 无论作业已分配给试验还是未分配，还会将独立执行的训练、批量转换和处理作业作为试验组件处理。未分配的试验组件可以在以后与试验相关联。包括数据集、算法、超参数和模型指标在内的所有实验构件都会予以跟踪和记录。这些数据允许客户跟踪模型的完整谱系，这有助于进行模型管理、审计和合规性验证。

手动跟踪

SageMaker 实验提供跟踪 API用于记录和跟踪本地运行的机器学习工作流程 SageMaker 录音室笔记本，包括经典 SageMaker 笔记本。这些实验必须作为 SageMaker 训练、批量转换或处理作业。

比较和评估实验

亚马逊 SageMaker 实验与亚马逊集成 SageMaker Studio。当您使用时 SageMaker Studio、SageMaker 实验会自动跟踪您的实验和试验，并显示跟踪数据的可视化效果和用于搜索数据的界面。

SageMaker 实验使用试验排行榜的概念，根据所选指标自动对试验进行组织、排序和分类。 SageMaker Studio 生成实时数据可视化效果，如指标图表和图形，以快速比较和确定性能最佳的模型。这些内容将随着实验的推进而实时更新。

亚马逊 SageMaker Autopilot

亚马逊 SageMaker 实验与亚马逊集成 SageMaker Autopilot。当你执行自动驾驶仪作业时， SageMaker 实验会为该作业创建一个实验，并试用针对可用试验组件、参数和构件的每种不同组合。您可以使用直观地深入了解所有试验和组件。 SageMaker Studio。

创建卓越亚马逊 SageMaker 试验

创建卓越亚马逊 SageMaker 实验来跟踪你的 SageMaker 培训、处理和转换工作。

下面的过程演示如何创建 SageMaker 实验 SageMaker 培训、处理或转换工作。标记为 (Studio) 的步骤描述了如何在亚马逊中查看实验 SageMaker Studio。你不必在 Studio 中运行实验即可在 Studio 中查看实验。

有关在现有文件中显示此功能的教程 SageMaker 请参阅 Studio 笔记本跟踪和比较教程 (p. 1766).

1. 导入sys模块来安装 SDK。

```
import sys
```

2. （可选）亚马逊 SageMaker Python 开发工具包，已预装在 SageMaker Studio。如果你计划在 Studio 之外运行代码，请安装 SageMaker Python 开发工具包。

```
!{sys.executable} -m pip install sagemaker
```

3. 安装使用 Python 开发工具包.

```
!{sys.executable} -m pip install sagemaker-experiments
```

4. 导入模块。

```
import time
from time import strftime

import sagemaker

from smexperiments.experiment import Experiment
from smexperiments.trial import Trial
from smexperiments.trial_component import TrialComponent
from smexperiments.tracker import Tracker
```

5. 获取执行角色并创建 SageMaker 会话。

```
role = sagemaker.get_execution_role()
sm_sess = sagemaker.session.Session()
```

6. 创建 SageMaker 实验。您的账户中的实验名称必须是唯一的。

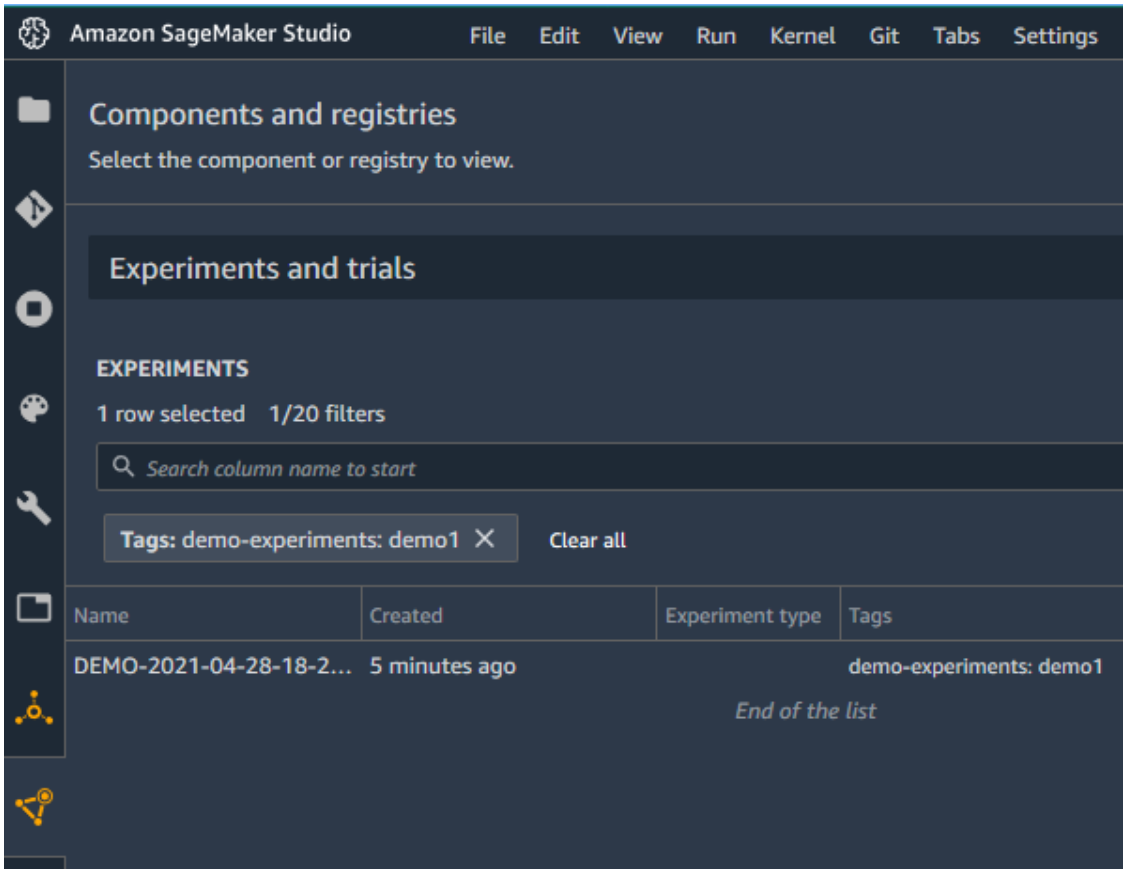
   Note

   tags 参数是可选的。您可以使用 Studio、SageMaker 控制台和 SDK 搜索标签。标签也可以应用于试验组件和试验组件。有关如何使用 Studio 搜索标签的信息，请参阅按标签搜索 (p. 1775).

```
create_date = strftime("%Y-%m-%d-%H-%M-%S")
demo_experiment = Experiment.create(experiment_name = "DEMO-{}".format(create_date),
                                    description = "Demo experiment",
                                    tags = [{'Key': 'demo-experiments', 'Value':
 'demo1'}])
```

7. （工作室）查看实验 SageMaker 在 Studio，在左侧边栏中，选择SageMaker 资源图标 ( )。在下拉菜单中，选择实验和试验以显示实验浏览器。

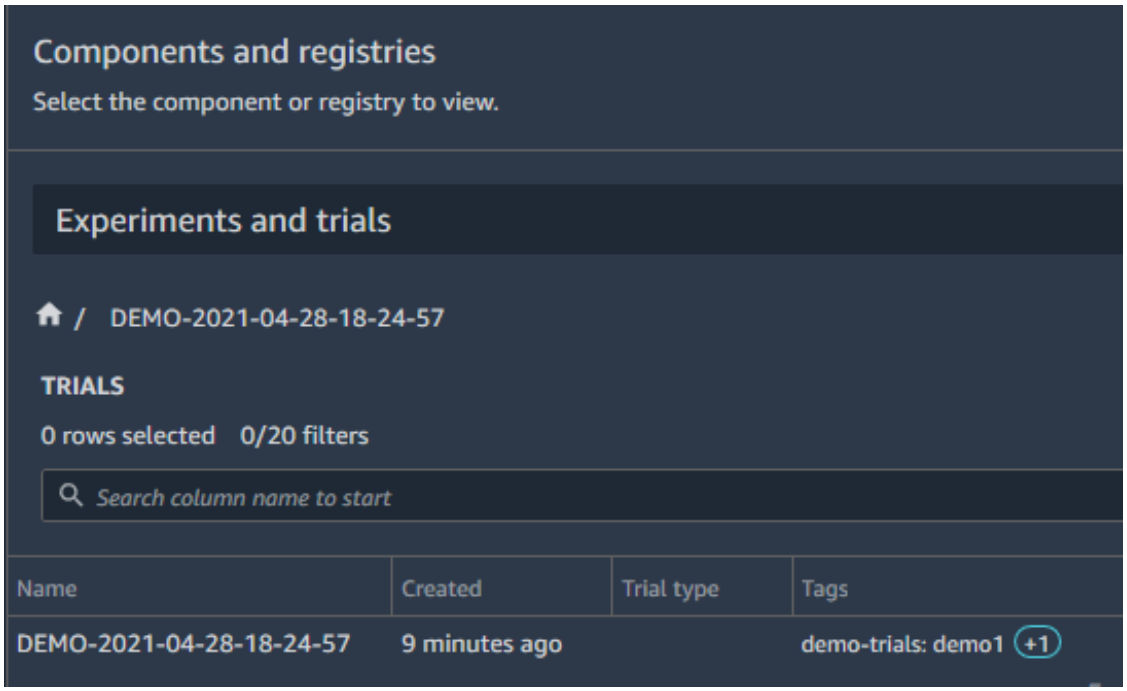
   在代码运行后，实验列表将包含新的实验。列表可能需要一点时间才能刷新并显示实验。还会显示实验标签上的过滤器。只显示具有匹配标签的实验。您的列表应如下所示：



8. 为实验创建试验。试验名称在您的账户中必须是唯一的。

```
demo_trial = Trial.create(trial_name = "DEMO-{}".format(create_date),
                          experiment_name = demo_experiment.experiment_name,
                          tags = [{'Key': 'demo-trials', 'Value': 'demo1'}])
```

9. (Studio) 在实验列表中双击实验以显示实验中的试验列表（此示例有一个试验）。您的列表应如下所示：



10. 创建试验组件，作为试验的一部分。试用组件是 SageMaker 任务。

    添加实验配置参数转换为适当的方法。这些区域有： SageMaker 支持下表中列出的作业。

任务	SageMaker Python SDK 方法	Boto3 方法
培训	估计器 .fit	CreateTrainingJob
处理	处理器 .run	CreateProcessingJob
转换	变形器。转换	CreateTransformJob

    下面的示例适用于训练作业。这些区域有：`Tags`参数向试用组件添加标签。`ExperimentName`未指定，因为试验在早期步骤中创建试验时与实验相关联。

    使用 SageMaker Python 开发工具包

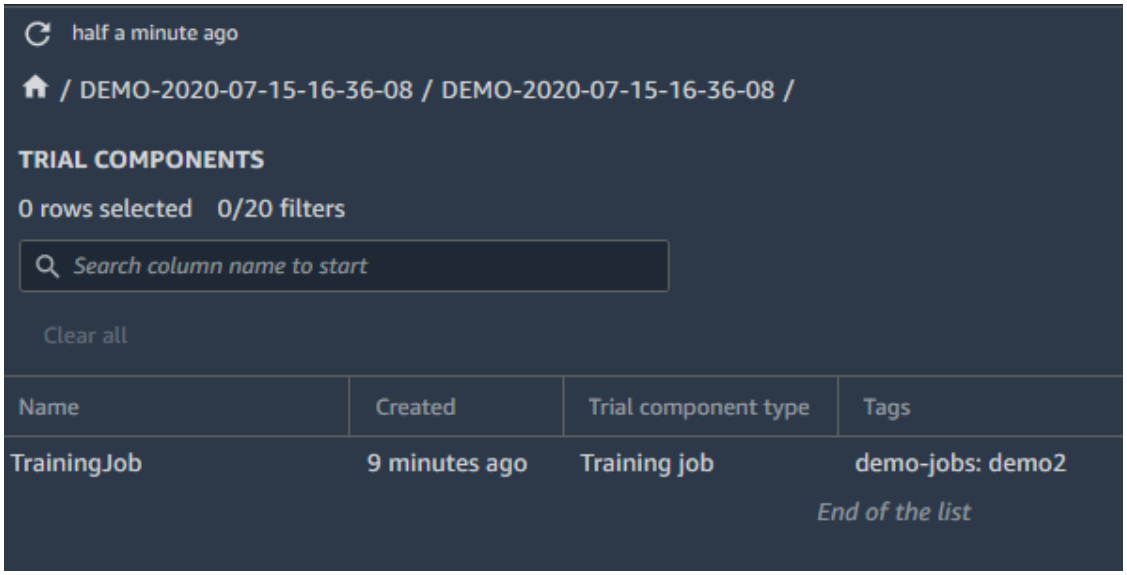
```
sagemaker.estimator.Estimator(
    ...,
    sagemaker_session = sm_sess,
    tags = [{'Key': 'demo-jobs', 'Value': 'demo2'}])

estimator.fit(
    ...,
    experiment_config = {
        # "ExperimentName"
        "TrialName" : demo_trial.trial_name,
        "TrialComponentDisplayName" : "TrainingJob",
    })
```

    使用 Boto3

```
create_training_job(
```

```
    ...,
    "ExperimentConfig": {
        # "ExperimentName"
        "TrialName" : demo_trial.trial_name,
        "TrialComponentDisplayName" : "TrainingJob",
    },
    "Tags": [{'Key': 'demo-jobs', 'Value': 'demo2'}])
```

11. (Studio) 在试用列表中，双击试用版以显示试用中组件的列表（此示例有一个试用版）。您的列表应如下所示：



12. (Studio) 要查看有关实验、试用和作业（试用组件）的信息，请参阅查看和比较亚马逊 SageMaker 查看实验、试验和试验组件 (p. 1763).

要清除您创建的资源，请参阅。清理 Amazon SageMaker 实验资源 (p. 1776).

查看和比较亚马逊 SageMaker 查看实验、试验和试验组件

Amazon SageMaker 实验包含多个审判有一个相关的目标。试验包括一个或多个试用组件，例如数据预处理作业和训练作业。

您可以使用实验浏览器显示这些实体的列表。您可以按实体名称、类型和标签来筛选列表。实体以分层视图（即实验 > 试验 > 试验组件）显示。双击一个实体可查看层次结构中位于较低级别的其他实体。使用列表上方的面包屑移动到层次结构中的更高级别。

对于使用 SageMaker 例如笔记本，请参阅跟踪和比较教程 (p. 1766). 有关 Studio 用户界面的概述，请参阅。亚马逊 SageMaker Studio UI 概述 (p. 77).

主题

- 查看实验、试验和试验组件 (p. 1763)
- 比较实验、试验和试验组件 (p. 1765)

查看实验、试验和试验组件

亚马逊 SageMaker Studio 提供了实验浏览器，您可以使用该浏览器查看实验、试验和试验组件列表。您可以选择其中一个实体来查看有关实体的详细信息，也可以选择多个实体进行比较。

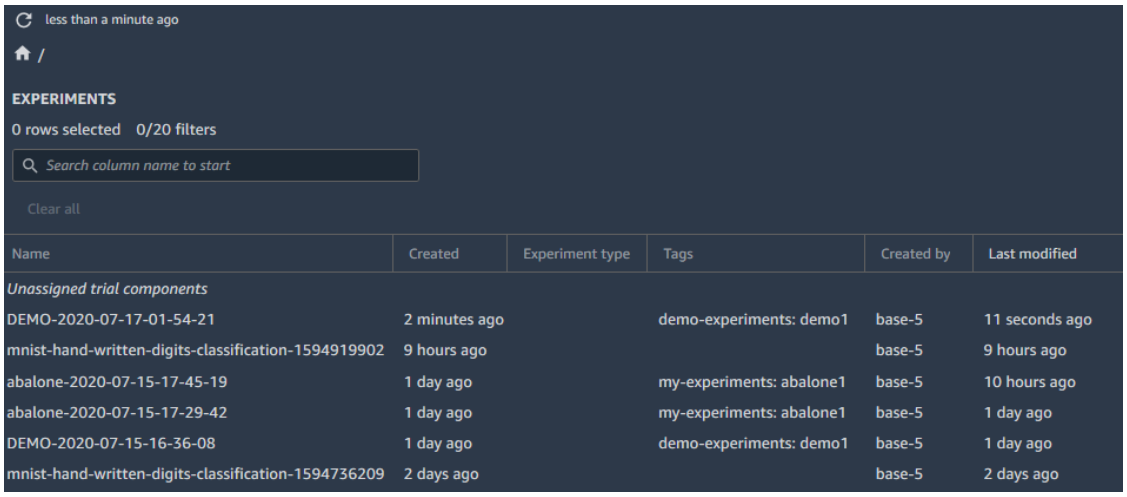
查看实验、试验和试验组件

1. 在 Studio 的左侧边栏中，选择SageMaker 资源图标 ( )。在下拉菜单中，选择实验和试验以显示实验浏览器。

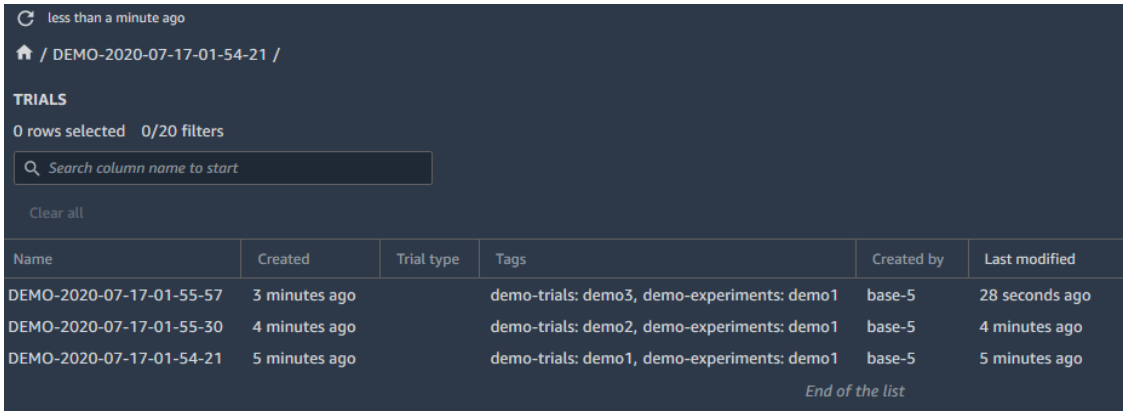
   此时将显示实验及其属性的列表。该列表包含所有 SageMaker 账户中的实验，包括在以外创建的实验 SageMaker Studio。

   Note

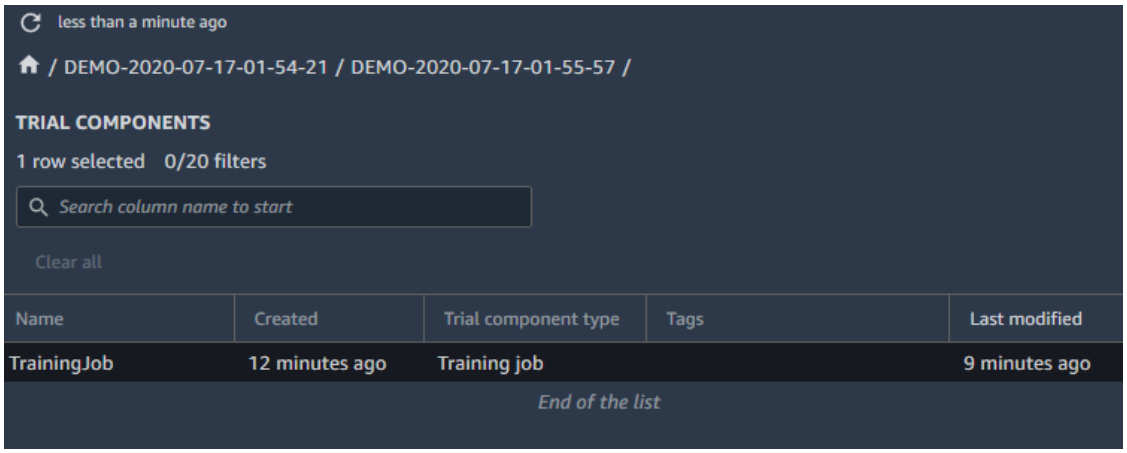
   要查看所有属性，您可能需要通过拖动右边框来扩大实验浏览器的宽度。



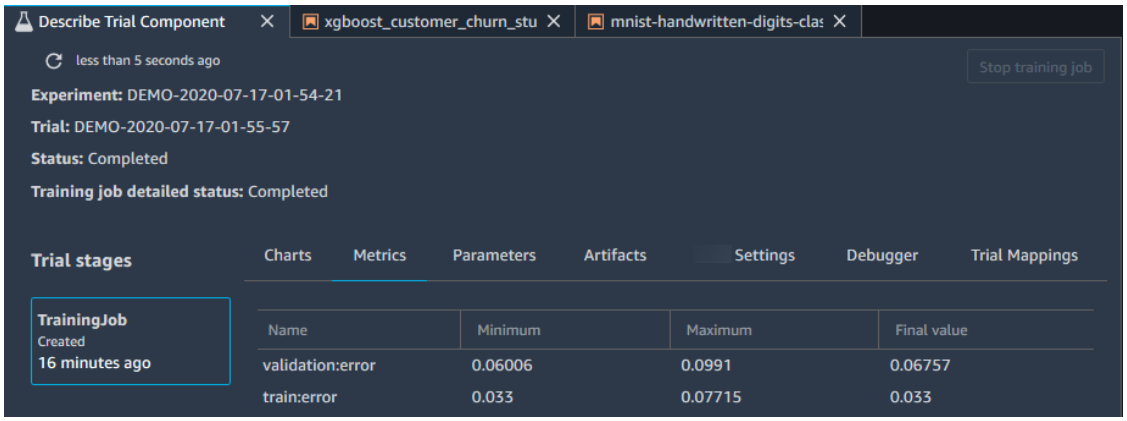
2. 在实验列表中，双击实验以显示实验中的试验列表。



3. 双击试验以显示试验中的组件的列表。



4. 双击其中一个组件以打开描述试验组件选项卡。



5. 在存储库的描述试验组件选项卡中，选择以下任一列标题以查看有关每个试验组件的可用信息：

   - 图表— 构建您自己的图表。
   - 指标— 记录的指标`Tracker`在试运行期间。
   - 参数— 超参数值和实例信息。
   - Artifacts— 输入数据集和输出模型的 Amazon S3 存储桶存储位置。
   - Amazon设置— Job 名称、ARN、状态、创建时间、训练时间、计费时间、实例信息等。
   - Debugger (调试程序)— 调试程序规则和发现的任何问题的列表。
   - Trial Mappings (试验映射)

有关比较实验实体的信息，请参阅查看和比较亚马逊 SageMaker 查看实验、试验和试验组件 (p. 1763).

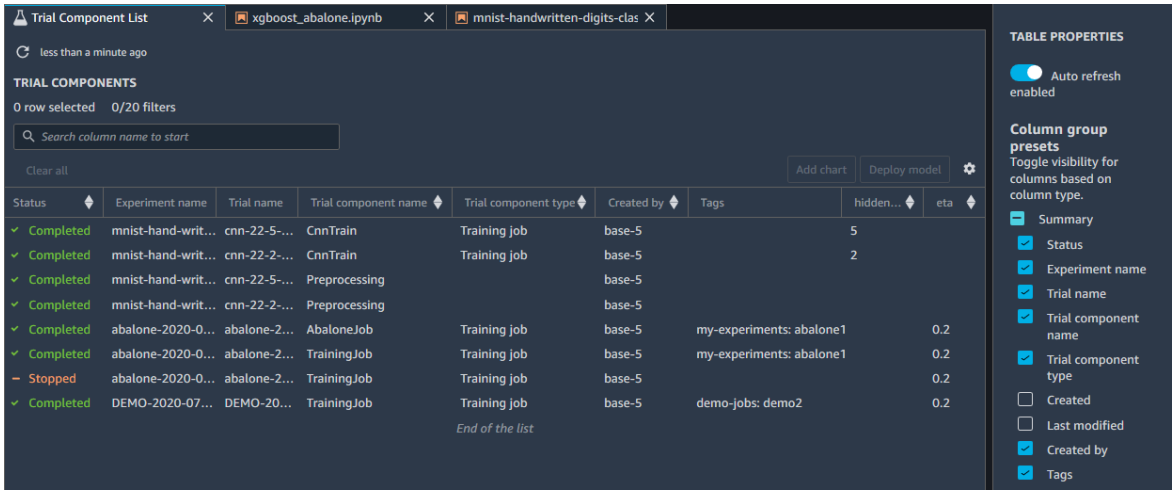
比较实验、试验和试验组件

您可以通过选择实体并在试验组件列表中将其打开来比较实验、试验和试验组件。试验组件列表称为 Studio 排行榜。在排行榜中，您可以执行以下操作：

- 查看有关实体的详细信息
- 比较实体
- 停止训练作业
- 部署模型

要比较实验、试验和试验组件

1. 在的左侧边栏 SageMaker 工作室，选择SageMaker 实验列表图标 ( )。
2. 在实验浏览器中，选择实验或试验列表。有关更多信息，请参阅 查看实验、试验和试验组件 (p. 1763)。
3. 选择要比较的实验或试验，右键单击所选内容，然后选择在试用组件列表中打开. 排行榜打开并列出相关的实验实体，如以下屏幕截图所示。



排行榜有表属性窗格在右侧。使用设置图标 ( ) 以打开并关闭窗格。您可以按类别或各个列隐藏或显示属性。显示图表时，窗格将更改为显示图表属性。

有关搜索实验实体的信息，请参阅使用亚马逊搜索实验 SageMaker 工作室 (p. 1772).

跟踪和比较教程

本教程演示了如何使用亚马逊直观地跟踪和比较模型训练实验中的试验。 SageMaker Studio。本教程的基础是MNIST 手写数字分类实验笔记本。

旨在使得能够在 MNIST 笔记本打开的情况下，在 Studio 的旁边查看本主题。在您遍历单元格时，本文档中的这一部分会突出显示相关代码并向您说明如何在 Studio 中观察结果。为简洁起见，已编辑一些代码段。

要清除笔记本创建的资源，请参阅清理 Amazon SageMaker 实验资源 (p. 1776).

有关展示 Studio 的其他功能的教程，请参阅 亚马逊 SageMaker Studio 导览 (p. 85)。

先决条件

- MNIST 示例笔记本和配套 mnist.py 文件的本地副本。两个文件都可以从`sagemaker_experiments/mnist-handwritten-digits-classification-experiment`中的文件夹aws/亚马逊ageemaker-示例存储库。要下载文件，请选择每个链接，右键单击 Raw (原始) 按钮，然后选择 Save as (另存为)。
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的IAM Identity Center或 IAM 账户要登录 SageMaker Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。

主题

- 在 Studio 中打开笔记本 (p. 1767)
- 安装实验开发工具包并导入模块 (p. 1767)

- 转换和跟踪输入数据 (p. 1767)
- 创建和跟踪实验 (p. 1768)
- 比较和分析试验 (p. 1770)

在 Studio 中打开笔记本

打开 笔记本

1. 登录到 Studio。
2. 在左侧边栏中，选择 File Browser (文件浏览器) 图标 ( )。
3. 在文件浏览器窗格的顶部，选择 Up arrow (上箭头键) 图标，此时将打开 File Upload (文件上传) 对话框。浏览找到并选择您的本地版本的 mnist-handwritten-digits-classification-experiment.ipynb 和 mnist.py 文件，然后选择 Open (打开)。
4. 这两个文件在文件浏览器中列出。双击上传的笔记本文件可在新选项卡中打开笔记本。
5. 在笔记本的右上角，确保内核是 Python 3 (Data Science) (Python 3 (数据科学))。如果不是，请选择当前内核名称以打开 Select Kernel (选择内核) 下拉列表。选择 Python 3 (Data Science) (Python 3 (数据科学))，然后选择 Select (选择)。

安装实验开发工具包并导入模块

这些区域有：亚马逊 SageMaker 实验 Python SDK是独立于亚马逊 SageMaker Python 开发工具包，它已预装在 SageMaker Studio。运行笔记本中的前几个单元格以安装实验并导入实验模块。笔记本单元格的相关部分如下所示。

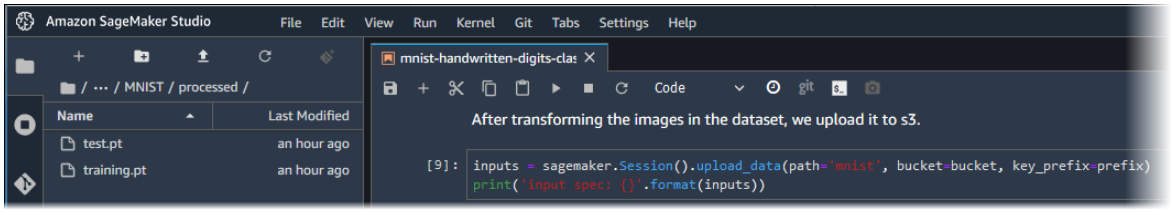
```
import sys
!{sys.executable} -m pip install sagemaker-experiments
```

```
import sagemaker
from sagemaker import get_execution_role
from sagemaker.session import Session
from sagemaker.analytics import ExperimentAnalytics

from smexperiments.experiment import Experiment
from smexperiments.trial import Trial
from smexperiments.trial_component import TrialComponent
from smexperiments.tracker import Tracker
```

转换和跟踪输入数据

接下来的几个单元格创建 Amazon S3 存储桶和存储桶中名为的文件夹。mnist. 在 Studio 中，文件浏览器显示 mnist 文件夹。输入数据下载到 mnist/MNIST/raw 文件夹，归一化，然后转换后的数据上传到 mnist/MNIST/processed 文件夹。您可以向下钻取到 mnist 文件夹以显示数据文件，但不打开数据文件。

您的屏幕应类似于以下内容：



Dataset (数据集) 部分的最后一个单元格为转换作业创建跟踪器。跟踪器会记录标准化参数和存储转换后数据集的 Amazon S3 存储桶的 URI。在后面的部分中，我们将说明如何在 Studio 中查找此信息。在下一个部分中，跟踪器用于跟踪实验和试验运行。

```
with Tracker.create(display_name="Preprocessing", sagemaker_boto_client=sm) as tracker:
    tracker.log_parameters({
        "normalization_mean": 0.1307,
        "normalization_std": 0.3081,
    })
    tracker.log_input(name="mnist-dataset", media_type="s3/uri", value=inputs)
```

创建和跟踪实验

以下过程创建并跟踪实验以确定模型的 `num_hidden_channel` 超参数的效果。作为实验的一部分，在一个循环内创建了五个试验，`num_hidden_channel` 超参数的每个值对应一个试验。稍后在笔记本中，您将比较这五个试验的结果。

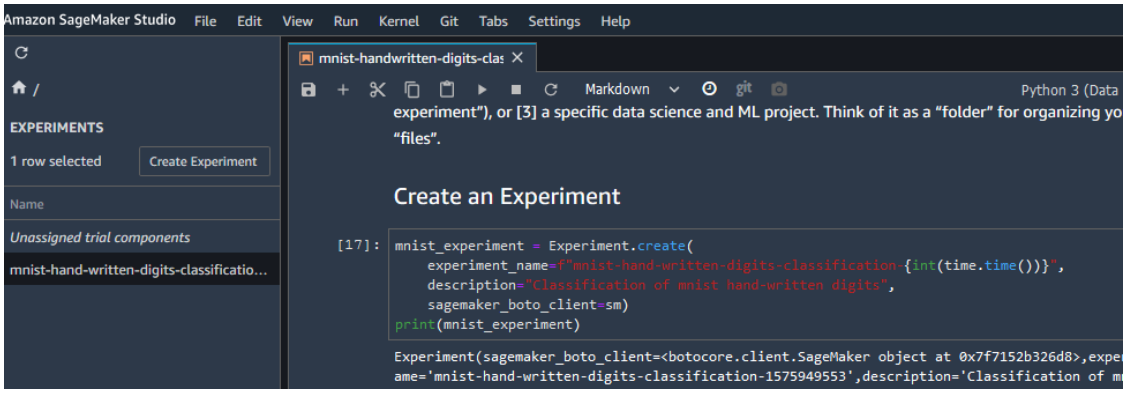
1. 在 Studio 的左侧边栏中，选择SageMaker 资源图标 ( )。在下拉菜单中，选择实验和试验在您的账户中显示实验列表。
2. 运行以下单元格。

```
mnist_experiment = Experiment.create(
    experiment_name=f"mnist-hand-written-digits-classification-{int(time.time())}",
    description="Classification of mnist hand-written digits",
    sagemaker_boto_client=sm)
print(mnist_experiment)
```

输出：

```
Experiment(sagemaker_boto_client=<botocore.client.SageMaker object at 0x7f7152b326d8>,
           experiment_name='mnist-hand-written-digits-classification-1575947870',
           description='Classification of mnist hand-written digits',
           experiment_arn='arn:aws:sagemaker:us-east-2:acct-id:experiment/mnist-hand-
written-digits-classification-1575947870')
```

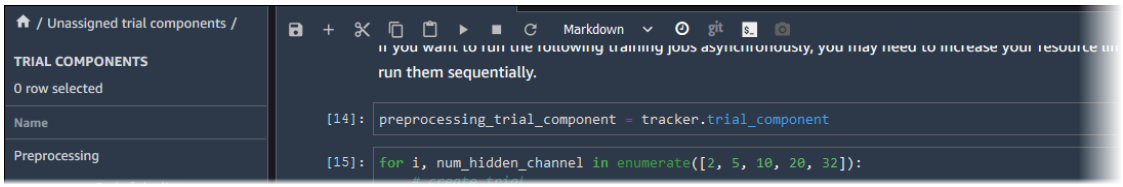
在代码运行后，该实验将在实验列表中包含一个对应的条目。可能需要一段时间才能显示，您可能必须刷新实验列表。您的屏幕应类似于以下内容：



3. 运行以下单元格。

```
preprocessing_trial_component = tracker.trial_component
```

在代码运行后，实验列表将包含一个标为 Unassigned trial components (未分配的试验组件) 的条目。试验组件条目是之前创建的数据预处理步骤。双击试验组件以进行验证。试验组件目前没有与实验相关联。您的屏幕应类似于以下内容：



4. 在实验浏览器顶部的导航位置提示中选择 Home 图标。在此处双击实验以显示实验中的试验列表。
5. 以下代码为实验创建试验。每个试验通过对 num_hidden_channel 超参数使用不同的数字来训练模型。将预处理试验组件添加到每个试验来进行完整跟踪（例如，用于审计目的）。该代码还指定了以下指标的定义：

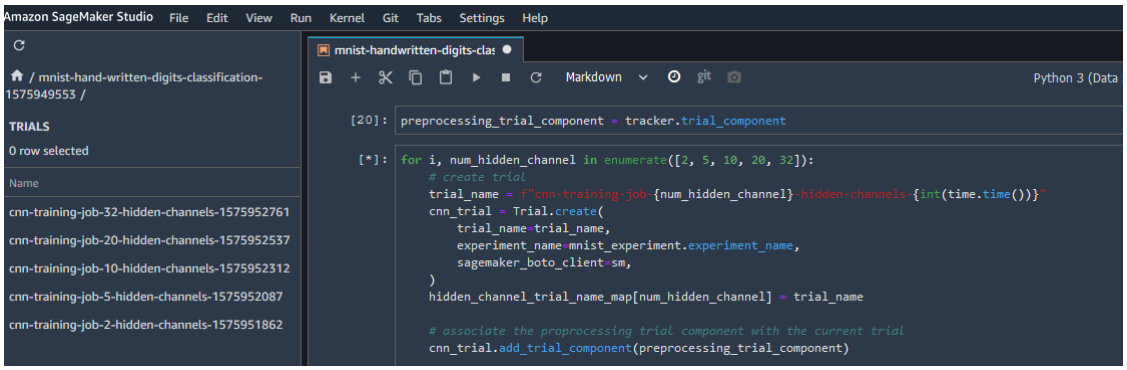
   - 训练损失
   - 测试损失
   - 测试精度

   定义告诉 SageMaker 从算法的日志输出中捕获这些指标。这些指标稍后用于评估和比较模型。

```
preprocessing_trial_component = tracker.trial_component

for i, num_hidden_channel in enumerate([2, 5, 10, 20, 32]):
    trial_name = f"cnn-training-job-{num_hidden_channel}-hidden-channels-
{int(time.time())}"
    cnn_trial = Trial.create(
        trial_name=trial_name,
        experiment_name=mnist_experiment.experiment_name,
        sagemaker_boto_client=sm,
    )
    hidden_channel_trial_name_map[num_hidden_channel] = trial_name

    cnn_trial.add_trial_component(preprocessing_trial_component)

    estimator = PyTorch(
        py_version='py3',
        framework_version='1.1.0',
        ...,
        hyperparameters={
            'hidden_channels': num_hidden_channel,
            ...
        },
        metric_definitions=[
            {'Name':'train:loss', 'Regex':'Train Loss: (.*?);'},
            {'Name':'test:loss', 'Regex':'Test Average loss: (.*?),'},
            {'Name':'test:accuracy', 'Regex':'Test Accuracy: (.*?)%;'}
        ],
        enable_sagemaker_metrics=True,
    )

    cnn_training_job_name = "cnn-training-job-{}".format(int(time.time()))

    estimator.fit(
        inputs={'training': inputs},
        job_name=cnn_training_job_name,
        experiment_config={
            "TrialName": cnn_trial.trial_name,
            "TrialComponentDisplayName": "Training",
        },
```

```
    )
```

试验列表会在每个训练作业运行时自动更新。每次显示试验需要花费几分钟时间。您的屏幕应类似于以下内容：



比较和分析试验

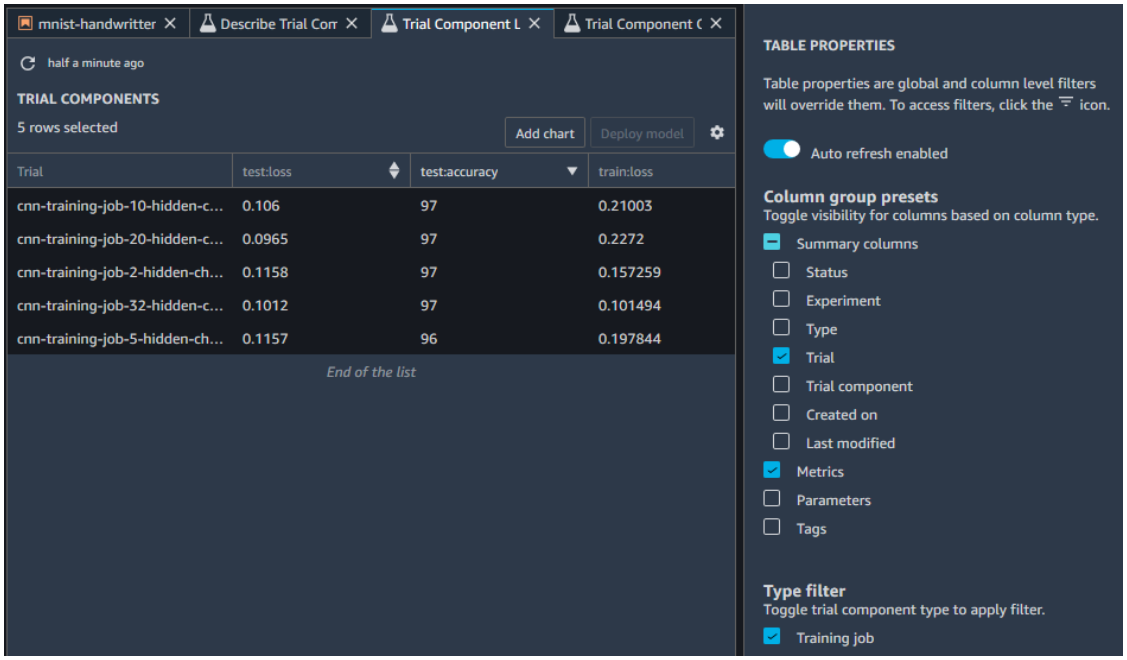
此部分不同于笔记本，向您说明如何使用 SageMaker Studio UI。

查看试验的详细信息

1. 双击其中一个试验以显示与该试验关联的试验组件的列表。每个试验均有一个预处理作业和训练作业。双击其中一个组件以打开一个显示有关每个组件的信息的新选项卡。
2. 在 Trial stages (试验阶段) 下，选择 Preprocessing (预处理)。在 Describe Trial Component (描述试验组件) 菜单上，选择 Parameters (参数) 以显示之前记录的标准化参数。下一步，选择Artifacts以显示存储已转换数据集的 Amazon S3 存储桶的 URI。
3. 在 Trial stages (试验阶段) 下，选择 Training (训练)。在 Describe Trial Component (描述试验组件) 菜单上，选择以下项目以显示有关训练作业试验组件的信息。
   - 指标–`test:loss`、`test:accuracy`, 和`train:loss`
   - 参数— 超参数值和实例信息
   - Artifacts— Amazon S3 存储用于输入数据集和输出模型
   - Amazon设置— 作业名称、ARN、状态、创建时间、训练时间、计费时间、实例信息等

查看按 `test:accuracy` 排序的试验列表

1. 选择 TRIAL COMPONENTS (试验组件) 上方的导航位置提示上的试验名称以显示试验列表。
2. 选择所有五个试验。按住 Ctrl/CMD 键并选择每个试验。右键单击所选内容，然后选择 Open in trial component list (在试验组件列表中打开)。这将打开一个新的选项卡，其中会显示每个试验和试验组件。
3. 如果 TABLE PROPERTIES (表属性) 窗格未打开，请选择右上角的 Settings (设置) 图标 ( ) 来将其打开。取消选择除 Trial (试用)、Metrics (指标) 和 Training job (训练作业) 之外的所有内容。选择 Settings (设置) 图标以关闭窗格。
4. 选择 test:accuracy 列标头以通过降低最大测试精度来对列表进行排序。您的屏幕应类似于以下内容：

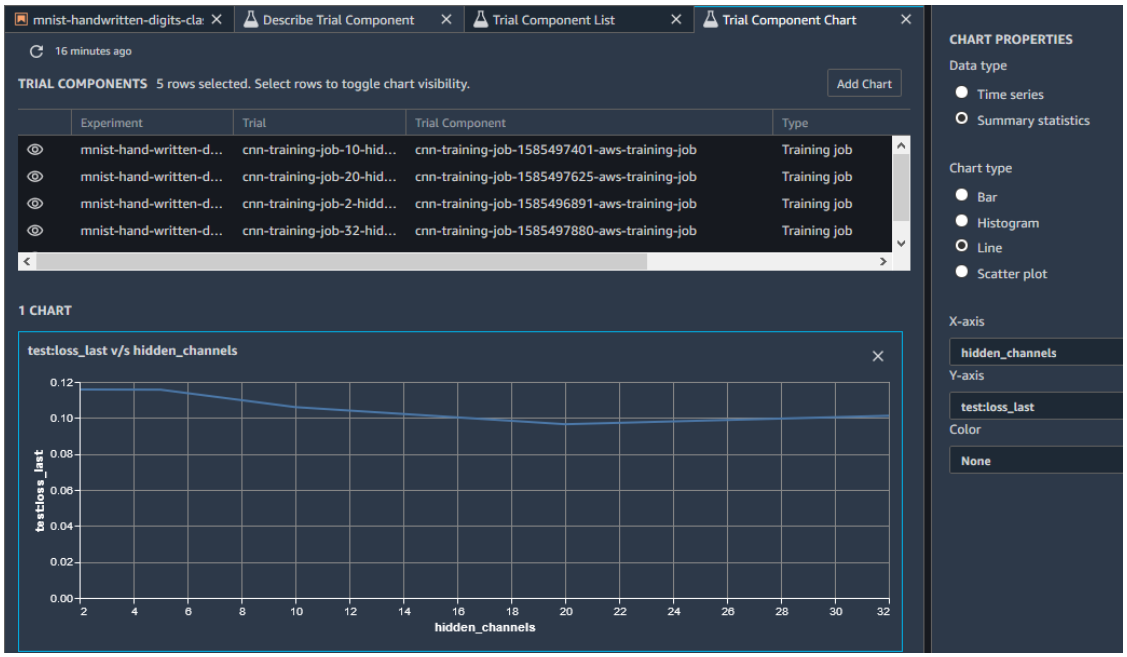


查看 `test:loss` 与 `num_hidden_channel` 的图表

1. 在 TRIAL COMPONENTS (试验组件) 窗格中，选择所有五个试验，然后选择 Add chart (添加图表)。在图表区域内进行选择，以打开 CHART PROPERTIES (图表属性) 的首选项窗格。
2. 在 CHART PROPERTIES (图表属性) 中，选择以下内容：

   - Data type (数据类型) - Summary statistics (摘要统计数据)
   - Chart type (图表类型) - Line (线条)
   - X-axis (X 轴) - hidden-channels (隐藏的通道)
   - Y 轴-测试:loss_last
   - Color (颜色) - None (无)

   您的屏幕应类似于以下内容：



使用亚马逊搜索实验 SageMaker 工作室

您可以使用实验浏览器搜索实验、试验和试验组件。选择实体后，您可以在亚马逊上搜索这些实体的属性 SageMaker 排行榜。

主题

- 搜索实验、试验和试验组件 (p. 1772)
- 搜索 SageMaker 排行榜 (p. 1773)
- 按标签搜索 (p. 1775)

搜索实验、试验和试验组件

您可以搜索并创建多个筛选器，以便在实验浏览器中显示实验、试验和试验组件的子集。搜索和筛选列表后，您可以在排行榜中打开实体并搜索其他属性。

要搜索实验、试验和试验组件，然后应用筛选器

有关以下步骤以及屏幕截图的更多信息，请参阅搜索 SageMaker 排行榜 (p. 1773).

1. 在实验浏览器中，导航到要搜索的实体类型：实验、试验或试验组件。
2. 将光标放在搜索框中以显示可搜索的列表。
3. 选择要搜索的列，将打开特定于您选择的列的以下对话框之一：

   - 字符串— 在打开的搜索对话框中输入文本字符串。输入组件属性值一部分的字符串，它至少具有三个字符。此过滤器将列表限制为具有包含文本字符串的属性值的组件。
   - 离散值— 从列表中选择一个或多个属性值。此筛选器将列表限制为属性值与至少一个选定值匹配的组件。
   - 日期— 在中输入日期起始日期和迄今为止，或者从日历中选择日期。此筛选器将列表限制为在指定数据范围内创建或上次修改的组件。

- 标记— 在搜索标签键，开始输入标签的密钥。标签键列表仅筛选为名称以您输入的文本字符串开头的键。从列表中选择要搜索的标签键。In搜索标签值，输入完整的标签值，然后选择Apply.

4. 要创建筛选器，请选择Apply.
5. 要应用其他筛选条件，请重复上述步骤。您最多可拥有 20 个筛选条件。实体必须与要显示的所有筛选器匹配。

搜索 SageMaker 排行榜

Amazon SageMaker Studio 试验组件列表称为排行榜。你可以使用排行榜来比较你的试验和实验。排行榜列出了试验的属性，例如试验状态、调试器状态、标签、指标、超参数以及输入和输出构件。

你可以搜索排行榜以查找特定的试用组件。您可以应用搜索结果来创建一个筛选器，该筛选器仅显示满足筛选器的那些组件。您最多可以应用 20 个筛选条件。仅当组件匹配所有筛选器时才会显示它。要删除筛选条件，请选择X显示在筛选条件右侧。

搜索排行榜并应用过滤器

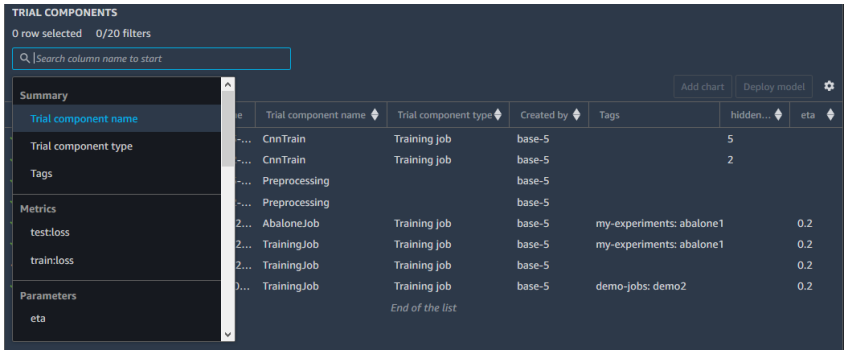
1. 如果表属性窗格未在中打开 SageMaker 工作室，选择设置图标 ( ) 以打开它。
2. 在设置窗格中，选中要查看的列对应的复选框，然后选择设置图标以关闭窗格。
3. 将光标放在搜索列名称框来显示以下列表表属性可搜索的列名称：

   摘要部分

   - 试用组件名称
   - 试用组件类型
   - 已创建
   - 上次修改时间
   - 创建者
   - 标签

   详情部分（所有栏目）

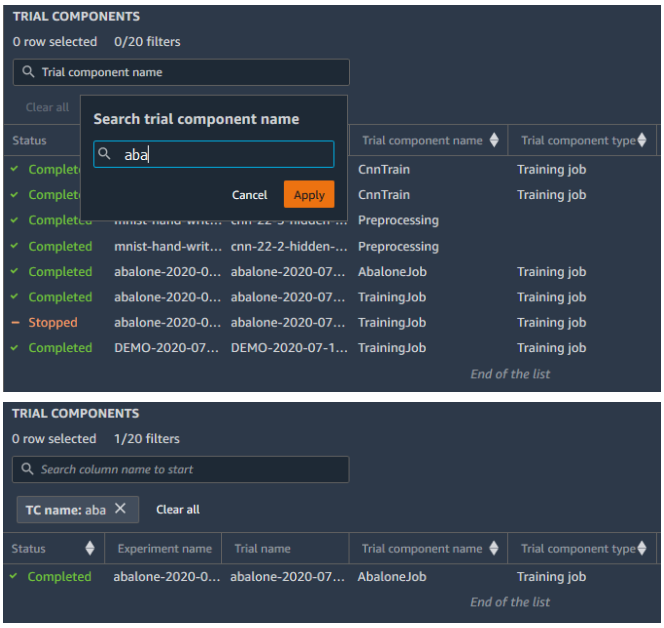
   - 指标
   - 参数
   - 输入构件
   - 输出构件



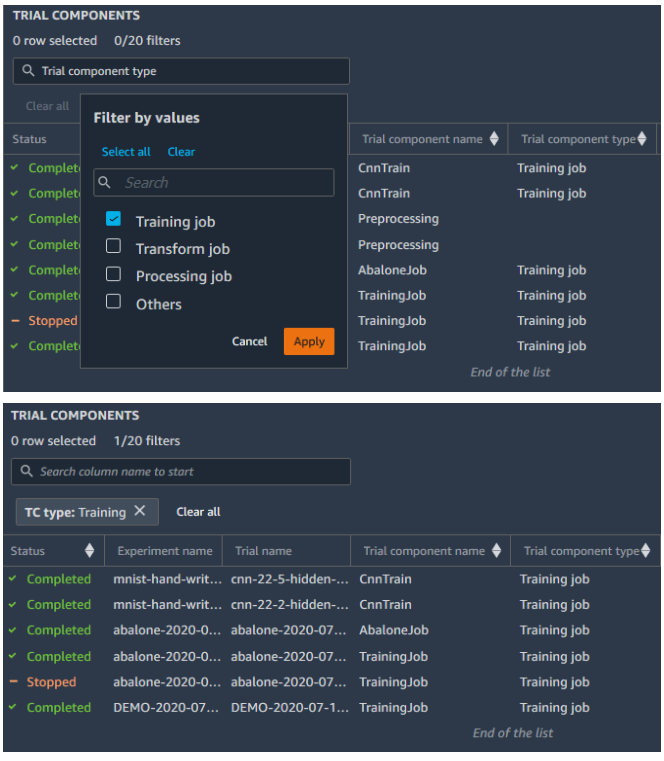
4. 要在搜索详细信息部分时限制列名，请开始键入列名称。该列表仅显示名称以您输入的文本字符串开头的列。

5. 选择要搜索的列。这将打开一个特定于该列中数据类型的对话框。
6. 在对话框中输入相应数据类型的搜索条件：

   - 字符串— 在打开的搜索对话框中输入文本字符串。输入组件属性值一部分的字符串，它至少具有三个字符。此过滤器将列表限制为具有包含文本字符串的属性值的组件。

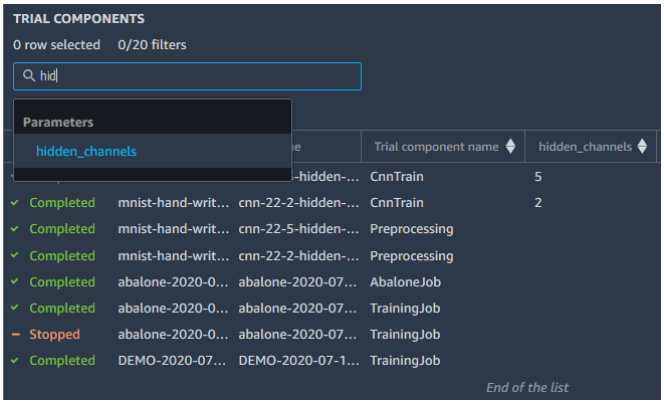


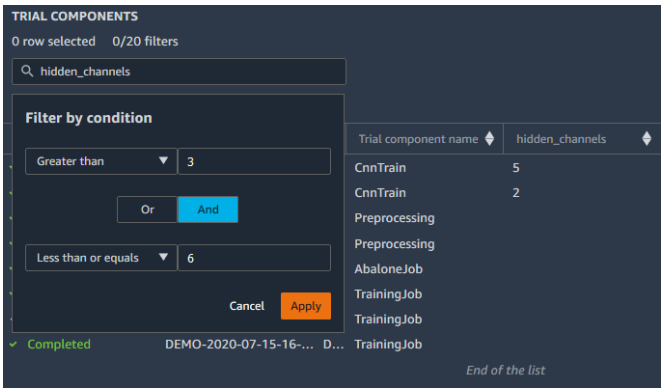
   - 离散值— 从列表中选择一个或多个属性值。此筛选器将列表限制为属性值与至少一个选定值匹配的组件。对于试用组件类型Column其他匹配列中没有值的组件。

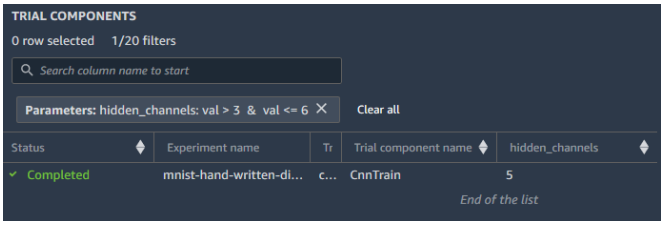


   - 日期— 在中输入日期起始日期和迄今为止或者从日历中选择日期。此筛选器将列表限制为在指定数据范围内创建或上次修改的组件。

- 创建者— Select我要么所有其他人. 此筛选器将列表限制为由当前登录用户或除当前登录用户之外的所有人创建的组件。
- 标记— 选择要搜索的标签。有关更多信息，请参阅 按标签搜索 (p. 1775)。
- 详细信息列— 指定组件的属性值必须满足的条件。







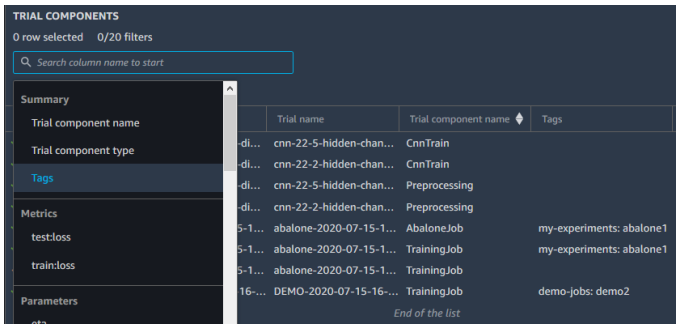
7. 要创建筛选器，请选择Apply.
8. 要应用其他筛选条件，请重复上述步骤。要显示组件，您必须与所有筛选条件匹配。您最多可拥有 20 个筛选条件。

按标签搜索

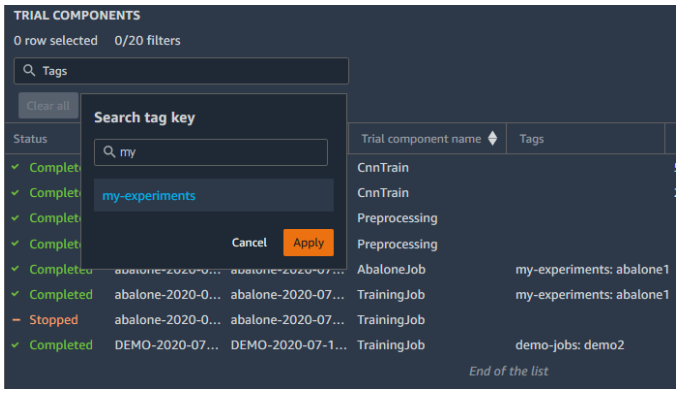
您可以在创建实体时或之后使用 AddTags API 将可搜索的标签添加到实验、试验和试验组件中。标签由区分大小写的唯一键和可选值组成。可以将多个标签添加到实体中。您可以将同一标签添加到多个实体。

按标签搜索并应用筛选条件

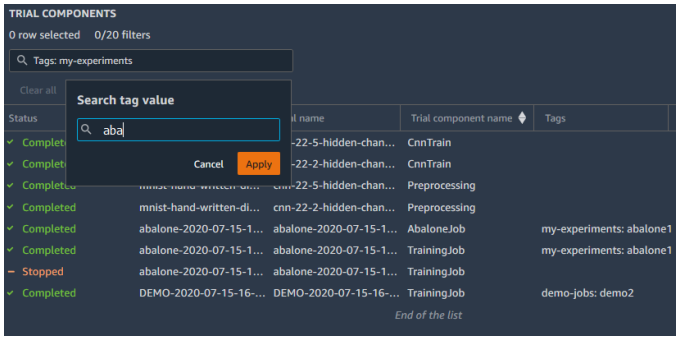
1. 如果表属性窗格未在中打开 SageMaker 工作室，选择设置图标 (⚙) 以打开它。
2. 在设置窗格中，选择标签如果未选择，然后选择设置图标以关闭窗格。
3. 将光标放在搜索列名称框然后选择标签.



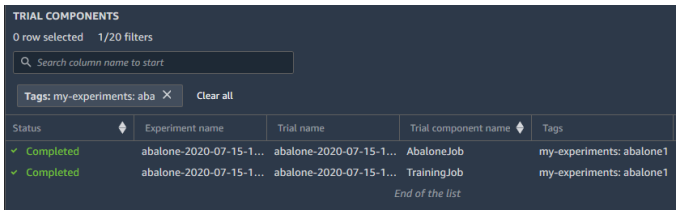
4. In搜索标签键，开始输入标签的密钥。标签键列表仅显示名称以您输入的文本字符串开头的键。



5. 从列表中选择要搜索的标签键。
6. In搜索标签值，输入完整的标签值，然后选择Apply.



7. 试验组件列表仅显示标签与您选择的键值对匹配的那些组件。



8. 要添加更多标签筛选条件，请重复前面的步骤。要显示组件的标签必须与所有筛选条件匹配。

清理 Amazon SageMaker 实验资源

为避免产生不必要的费用，请删除亚马逊 SageMaker 实验您不再需要的资源。无法通过 SageMaker 管理控制台或亚马逊删除实验资源 SageMaker Studio UI。此主题向您说明如何使用 Boto3 和实验开发工具包清理这些资源。有关实验开发工具包的更多信息，请参阅SageMaker 实验.

要删除实验，您必须删除实验中的所有试验。要删除试验，您必须从试验中删除所有试验组件。要删除试验组件，必须从所有试验中删除该组件。

> **Note**
>
> 试验组件可以独立于任何试验或实验而存在。您无需删除它们。如果你想重复使用它们，请注释掉`tc.delete()`在代码中。

主题

- 使用实验开发工具包进行清理 (p. 1777)
- 使用 Python 开发工具包进行清理（Boto3）(p. 1777)

使用实验开发工具包进行清理

使用实验 SDK 进行清理

```
import sys
!{sys.executable} -m pip install sagemaker-experiments
```

```
import time

from smexperiments.experiment import Experiment
from smexperiments.trial import Trial
from smexperiments.trial_component import TrialComponent
```

定义 clean up_sme_sdk

```
def cleanup_sme_sdk(experiment):
    for trial_summary in experiment.list_trials():
        trial = Trial.load(trial_name=trial_summary.trial_name)
        for trial_component_summary in trial.list_trial_components():
            tc = TrialComponent.load(
                trial_component_name=trial_component_summary.trial_component_name)
            trial.remove_trial_component(tc)
            try:
                # comment out to keep trial components
                tc.delete()
            except:
                # tc is associated with another trial
                continue
            # to prevent throttling
            time.sleep(.5)
        trial.delete()
        experiment_name = experiment.experiment_name
    experiment.delete()
    print(f"\nExperiment {experiment_name} deleted")
```

呼叫 clean _sme_sdk

```
experiment_to_cleanup = Experiment.load(
    # Use experiment name not display name
    experiment_name="experiment-name")

cleanup_sme_sdk(experiment_to_cleanup)
```

使用 Python 开发工具包进行清理（Boto3）

要使用 Boto 3 清除

```
import boto3
sm = boto3.Session().client('sagemaker')
```

定义 clean up_boto3

```
def cleanup_boto3(experiment_name):
    trials = sm.list_trials(ExperimentName=experiment_name)['TrialSummaries']
    print('TrialNames:')
    for trial in trials:
        trial_name = trial['TrialName']
        print(f"\n{trial_name}")

        components_in_trial = sm.list_trial_components(TrialName=trial_name)
        print('\tTrialComponentNames:')
        for component in components_in_trial['TrialComponentSummaries']:
            component_name = component['TrialComponentName']
            print(f"\t{component_name}")
            sm.disassociate_trial_component(TrialComponentName=component_name,
 TrialName=trial_name)
            try:
                # comment out to keep trial components
                sm.delete_trial_component(TrialComponentName=component_name)
            except:
                # component is associated with another trial
                continue
            # to prevent throttling
            time.sleep(.5)
        sm.delete_trial(TrialName=trial_name)
    sm.delete_experiment(ExperimentName=experiment_name)
    print(f"\nExperiment {experiment_name} deleted")
```

呼叫 cleanup_boto3

```
# Use experiment name not display name
experiment_name = "experiment-name"
cleanup_boto3(experiment_name)
```

使用亚马逊搜索 SageMaker 控制台和 API

开发机器学习模型通常需要广泛地试验不同数据集、算法和超参数值。要管理多达数千个机器学习模型实验，请使用 SageMaker 中的搜索功能。

您可以使用 SageMaker 搜索到：

- 使用属性、超参数、性能指标或任何元数据整理、查找和评估训练作业。
- 通过查看训练作业和模型指标（例如，训练损失或验证准确性）来找出性能最佳的模型。
- 跟踪模型的谱系，直到训练作业及其相关资源，例如训练数据集。

本主题介绍从 SageMaker 控制台和 SageMaker API。有关在亚马逊中搜索的信息 SageMaker 工作室，请参阅使用 Studio 搜索实验 (p. 1772).

主题

- 用于管理 ML 实验的示例笔记本 (p. 1779)
- 组织、查找、和评估训练任务(console) (p. 1779)
- 查找和评估训练作业 (API) (p. 1781)

- 验证您的训练作业使用的数据集 (p. 1782)
- 跟踪模型谱系 (p. 1782)

用于管理 ML 实验的示例笔记本

对于使用亚马逊的示例笔记本 SageMaker 管理 ML 实验的模型跟踪功能，请参阅使用亚马逊管理机器学习实验 SageMaker 模型跟踪功能.

有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147). 在您创建笔记本实例并将其打开后，选择SageMaker 示例选项卡以查看所有 SageMaker 示例。用于管理机器学习实验的笔记本位于高级功能部分。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。如果您有任何疑问，请将其发布到 Amazon Machine Learning 开发人员论坛。

组织、查找、和评估训练任务(console)

要组织训练作业，请为其分配一个或多个标签。

要查找特定训练作业、模型或资源，请使用模型跟踪功能以搜索分配给任意可搜索项的关键字。可搜索项目包括训练作业、模型、超参数、元数据、元数据、标签和网址. 要优化搜索结果，您可以使用多个条件进行搜索。

要选择最佳模型以进行部署，您需要针对一个或多个指标评估所有模型的性能。您可以使用模型跟踪结果来列出、排序和评估实验中模型的性能。

主题

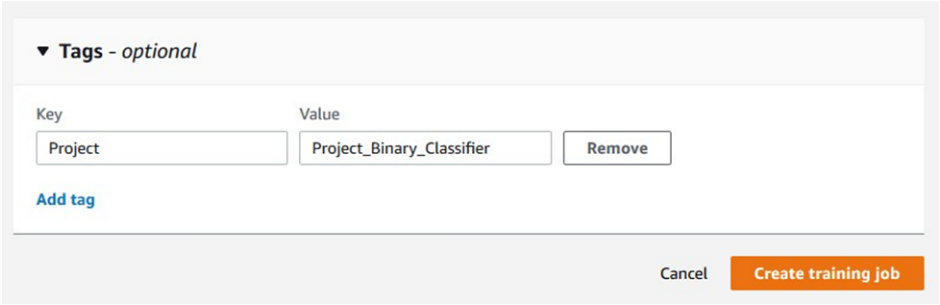
- 使用标签来跟踪训练作业（控制台） (p. 1779)
- 查找训练作业（控制台） (p. 1780)
- 评估模型(console) (p. 1780)

使用标签来跟踪训练作业（控制台）

要对训练作业进行分组，请创建标记以及描述性的密钥和值。例如，为以下各项创建标签键：项目、拥有者、客户和行业。

将标签添加至训练作业（控制台）

1. 打开亚马逊 SageMaker 控制台.
2. 在导航窗格中，选择 Training jobs (训练作业)，然后选择 Create training job (创建训练作业)。
3. 滚动到页面底部，并输入标签的键和值。



4. 要添加其他标签，请选择添加标签，然后添加另一个键值对。

查找训练作业（控制台）

您可以使用各种作业属性搜索训练作业。请注意，仅当您创建了具有该属性的训练作业时，某些搜索参数才会显示。例如，仅当您为训练作业添加了标记时，才会显示 Tags (标记)。

查找训练作业（控制台）

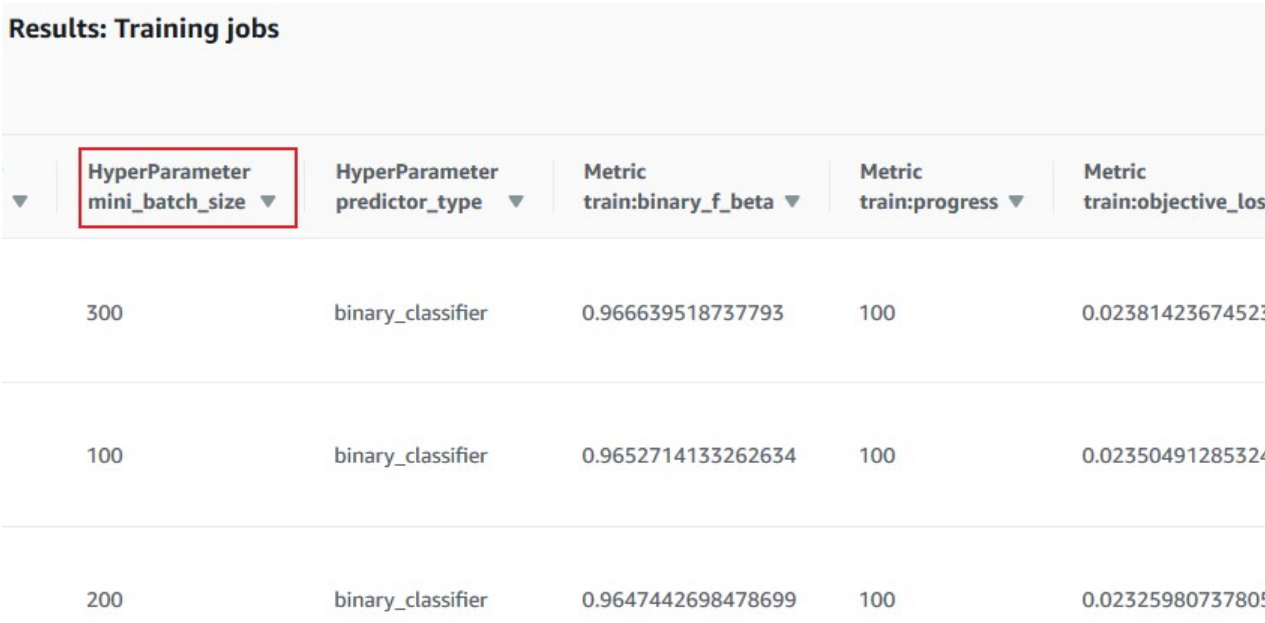
1. 打开亚马逊 SageMaker 控制台.
2. 在导航窗格中，选择 Search。
3. 添加 Parameters (参数)。
   a. 在搜索框中，输入参数并选择参数类型，例如 TrainingJobName。
   b. 选择条件操作。对于数字值，使用运算符，例如 is equals to (等于)、lesser than (小于) 或 greater than (大于)。对于基于文本的值，使用运算符，例如 equals to (等于) 或 contains (包含)。
   c. 为参数输入值。
4. （可选）要细化您的搜索，请添加额外的搜索条件。选择添加行并输入参数值。
5. 选择搜索。

评估模型(console)

要评估模型的性能，请查看其元数据、超参数和指标。要突出显示指标，调整视图以仅显示指标和重要的超参数。

评估模型(console)

1. 打开亚马逊 SageMaker 控制台.
2. 在导航窗格中，选择搜索和搜索用于训练作业 通过指定相关参数。结果将显示在表中。



3. 打开首选项窗口通过选择设置图标在搜索结果表中。
4. 要显示或隐藏超参数或指标，请通过以下方式将其打开或关闭：选择超参数要么指标.
5. 进行必要的更改，然后选择更新视图。
6. 在查看指标和重要的超参数后，您可以比较和对比结果。然后，您可以选择最佳模型来托管或调查性能较差的模型。

查找和评估训练作业 (API)

寻找和评估培训工作或获取在实验中使用的可搜索项的建议，您可以使用Search API。

主题

- 查找训练作业 (API) (p. 1781)
- 评估模型 (API) (p. 1781)
- 获取搜索建议 (API) (p. 1782)

查找训练作业 (API)

要查找训练作业，请使用 search_params 参数创建搜索参数。然后，您可以使用smclient 子进程中的AmazonSDK for Python (Boto3)。

以下示例说明了如何使用SearchAPI t找训练作业.

```
import boto3

search_params={
   "MaxResults": 10,
   "Resource": "TrainingJob",
   "SearchExpression": {
      "Filters": [{
            "Name": "Tags.Project",
            "Operator": "Equals",
            "Value": "Project_Binary_Classifier"
         }]},
  "SortBy": "Metrics.train:binary_classification_accuracy",
  "SortOrder": "Descending"
}

smclient = boto3.client(service_name='sagemaker')
results = smclient.search(**search_params)
```

评估模型 (API)

要评估模型，请按照中所述运行搜索查找训练作业 (API) (p. 1781)，然后查看模型指标，使用Amazon适用于 Python (Boto3) 的 SDK要创建一个表并绘制图形。

以下示例显示如何评估模型并在表中显示结果。

```
import pandas

headers=["Training Job Name", "Training Job Status", "Batch Size", "Binary Classification
 Accuracy"]
rows=[]
for result in results['Results']:
    trainingJob = result['TrainingJob']
    metrics = trainingJob['FinalMetricDataList']
    rows.append([trainingJob['TrainingJobName'],
     trainingJob['TrainingJobStatus'],
     trainingJob['HyperParameters']['mini_batch_size'],
     metrics[[x['MetricName'] for x in
     metrics].index('train:binary_classification_accuracy')]['Value']
    ])

df = pandas.DataFrame(data=rows,columns=headers)
```

```
from IPython.display import display, HTMLdisplay(HTML(df.to_html()))
```

获取搜索建议 (API)

要获取搜索建议，使用GetSearchSuggestionsAPI。

以下示例为了Amazon适用于 Python (Boto3) 的 SDK是get_search_suggestions请求物品包含了linear.

```
search_suggestion_params={
   "Resource": "TrainingJob",
   "SuggestionQuery": {
      "PropertyNameQuery": {
         "PropertyNameHint": "linear"
      }
   }
}
```

以下是 get_search_suggestions 请求的示例响应。

```
{
        'PropertyNameSuggestions': [{'PropertyName': 'hyperparameters.linear_init_method'},
                    {'PropertyName': 'hyperparameters.linear_init_value'},
                    {'PropertyName': 'hyperparameters.linear_init_sigma'},
                    {'PropertyName': 'hyperparameters.linear_lr'},
                    {'PropertyName': 'hyperparameters.linear_wd'}]
}
```

获得搜索建议后，您可以在搜索中使用其中一个属性名称。

验证您的训练作业使用的数据集

您可以使用模型跟踪功能来验证训练中使用了哪些数据集，holdout 数据集用于哪些地方，以及其他有关训练作业的详细信息。例如，使用模型跟踪功能验证在训练作业中使用了某个特定数据集进行审计或验证合规性。

要检查训练作业中是否使用了特定的数据集，请在 Amazon Simple Storage Service (Amazon S3) 中搜索指向其位置的 URL。模型跟踪功能将返回使用您指定的数据集的训练作业。如果搜索未返回数据集（结果为空），则数据集未用于训练作业中。例如，空结果可确认未使用保留数据集。

跟踪模型谱系

您可以使用模型跟踪功能来获取有关训练作业的谱系以及用于它们的模型资源的信息，包括数据集、算法、超参数和指标。例如，如果您发现托管模型的性能已下降，您可以查看其训练作业和所用资源来确定导致此问题的原因。

主题

- 跟踪模型谱系（控制台）(p. 1782)
- 跟踪模型谱系 (API) (p. 1783)

跟踪模型谱系（控制台）

跟踪模型的谱系（控制台）

1. 打开亚马逊 SageMaker 控制台.

2. 在导航窗格中，选择终端节点，然后选择相关的终端节点。
3. 滚动到 Endpoint configuration settings (终端节点配置设置) 部分。本节列出了在终端节点上部署的所有模型版本，以及指向创建每个模型版本的训练作业的超链接。

跟踪模型谱系 (API)

要追踪模型的谱系，获取模特的名字，然后用它来搜索用于训练作业.

以下示例显示如何使用跟踪模型的谱系：API.

```
# Get the name of model deployed at endpoint
endpoint_config = smclient.describe_endpoint_config(EndpointConfigName=endpointName)
model_name = endpoint_config['ProductionVariants'][0]['ModelName']

# Get the model's name
model = smclient.describe_model(ModelName=model_name)

# Search the training job by the location of model artifacts in Amazon S3
search_params={
   "MaxResults": 1,
   "Resource": "TrainingJob",
   "SearchExpression": {
      "Filters": [
         {
            "Name": "ModelArtifacts.S3ModelArtifacts",
            "Operator": "Equals",
            "Value": model['PrimaryContainer']['ModelDataUrl']
         }]},
}
results = smclient.search(**search_params)
```

找到培训工作之后，您可以查看用于训练模型的资源。

亚马逊SageMakerDebugger (调试程序)

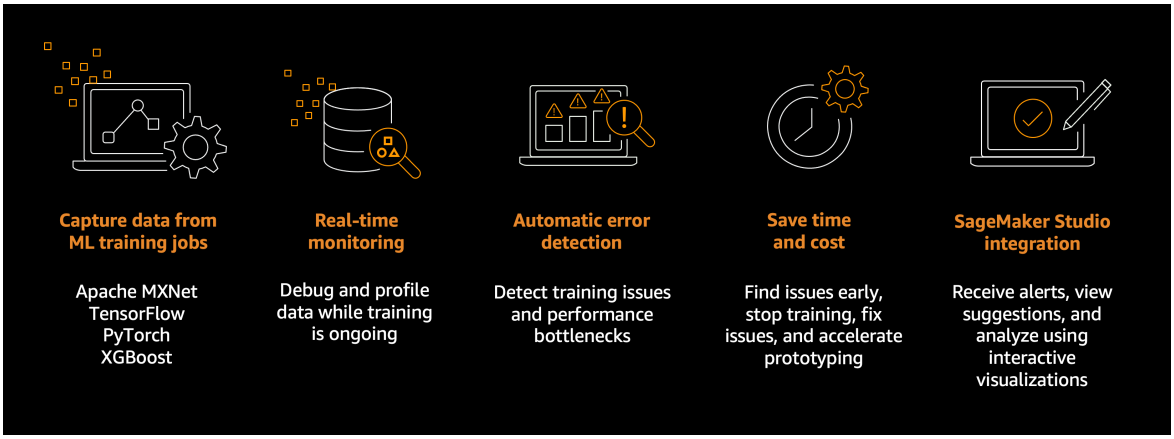
使用 Amazon 实时调试、监控和分析培训作业、检测非融合条件、通过消除瓶颈优化资源利用率、缩短培训时间并降低机器学习模型的成本SageMaker调试程序。

亚马逊SageMaker调试程序功能

机器学习 (ML) 培训作业可能存在诸如系统瓶颈、过度拟合、饱和激活函数和渐变消失等问题，这些问题可能会影响模型性能。

SageMaker调试器配置和调试训练作业，以帮助解决此类问题并提高机器学习模型的计算资源利用率和性能。调试器提供了一些工具，用于在发现训练异常情况时发送警报，对问题采取措施，并通过可视化收集的指标和张量来确定问题的根本原因。

SageMaker调试器支持 Apache MxNet，TensorFlow、PyTorch和 xgBoost。有关可用框架和版本的更多信息，请参阅支持的框架和算法 (p. 1785).



高级调试程序工作流程序如下：

1. 配置SageMaker使用调试器进行培训。
   - 使用配置SageMaker `EstimatorAPI（适用于 Python SDK）`.
   - 使用配置SageMaker `CreateTrainingJob请求（适用于 Boto3 或 CLI）`.
   - 配置自定义培训容器 (p. 1868)使用 Debugger。
2. 开始培训工作并实时监控培训问题。
   - SageMakerStudio 实验和试验中的 Studio 调试器仪表板 (p. 1881).
   - 调试程序内置规则列表 (p. 1826).
3. 获取警报并对培训问题迅速采取行动。
   - 在发现培训问题时使用接收短信和电子邮件并停止培训工作调试器内置的规则操作 (p. 1872).
   - 使用设置自己的操作亚马逊CloudWatch事件和Amazon Lambda (p. 1876).
4. 接收培训报告、解决问题的建议以及对培训工作的见解。
   - 用于深度学习框架的 Studio 调试器见解仪表板
   - 深度学习框架分析报告
   - SageMakerXGBoost 训练报告
5. 探索对培训问题和瓶颈的深入分析。
   - 有关分析训练作业，请参阅使用 smDebug 客户端库分析数据 (p. 1920).
   - 有关调试模型参数，请参阅在 TensorBoard 中可视化调试器输出张量 (p.).
6. 考虑调试器提供的建议修复问题，然后重复步骤 1-5，直到优化模型并达到目标准度。

这些区域有：SageMaker调试器开发人员指南将引导您完成以下主题。

主题

- 支持的框架和算法 (p. 1785)
- 亚马逊SageMaker调试程序架构 (p. 1786)
- 开始使用调试器教程 (p. 1788)
- 使用亚马逊配置调试器SageMakerPython 开发工具包 (p. 1796)
- 使用 Amazon SageMaker API 配置调试器 (p. 1816)
- 调试程序内置规则列表 (p. 1826)
- 为训练 Job 分析创建调试器自定义规则 (p. 1866)
- 将调试器用于自定义训练容器 (p. 1868)
- 对 Amazon SageMaker 调试器规则的操作 (p. 1872)
- Amazon SageMaker Studio 中的 Amazon SageMaker 调试器 (p. 1881)

- SageMaker 调试器交互式报告 (p. 1896)
- 使用 smDebug 客户端库分析数据 (p. 1920)
- 在 TensorBoard 中可视化 Amazon SageMaker 调试器输出张量 (p. 1927)
- Amazon SageMaker 调试器的最佳实践 (p. 1930)
- Amazon SageMaker 调试器高级主题和参考文档 (p. 1932)

支持的框架和算法

下表显示了SageMaker调试器支持的机器学习框架和算法。

SageMaker-supported frameworks and algorithms	Monitoring system bottlenecks	Profiling deep learning framework operations	Debugging output tensors
TensorFlow	全部Amazon深度学习容器	Amazon TensorFlow深度学习容器2.3.1 或更高版本	Amazon TensorFlow深度学习容器1.15.4 或更高版本
PyTorch		Amazon PyTorch深度学习容器1.6.0 或更高版本	Amazon PyTorch深度学习容器1.5.0 或更高版本
MXNet		-	AmazonMxNet 深度学习容器1.6.0 或更高版本
XGBoost	1.0-1、1.2-1、1.3-1	-	1.0-1、1.2-1、1.3-1
SageMaker通用估算	SageMaker使用图像 URI 的内置算法 自定义培训容器 (p. 1868)（使用Amazon深度学习容器映像、公共 Docker 映像或您自己的 Docker 映像）	-	自定义培训容器 (p. 1868)(适用于TensorFlow、PyTorch、MxNet 和 xgBoost（带手动挂钩注册）

- 监控系统瓶颈— 监控 CPU、GPU、内存、网络和数据 I/O 指标等资源的系统利用率。这是一个与框架和模型无关的功能，适用于SageMaker.
- 分析深度学习框架操作— 概要介绍的深度学习操作TensorFlow和PyTorch框架，例如步骤持续时间、数据加载器、向前和向后操作、Python 分析指标和框架特定的指标。
- 调试输出张量— 跟踪和调试模型参数，例如训练作业的权重、渐变、偏差和标量值。可用的深度学习框架是 Apache MxNet，TensorFlow、PyTorch和 xgBoost。

  Important

  对于TensorFlow与 Keras 的框架SageMaker调试器弃用零代码更改支持调试使用`tf.keras`的模块TensorFlow2.6 和更高版本。这是由于在TensorFlow版本 2.6.0. SageMaker调试器继续支持本机的零代码更改体验TensorFlow（不包括`tf.keras`模块)。

如果表中没有列出要训练和调试的框架或算法，请转到Amazon开发论坛然后留下反馈SageMaker调试程序。

将调试器用于自定义训练容器

把你的训练容器带到SageMaker并使用调试器深入了解你的训练工作。通过使用监控和调试功能在 Amazon EC2 实例上优化模型，最大限度地提高工作效率。

有关如何使用SMDebug客户端库，将其推送到 Amazon Elastic Container Registry (Amazon ECR)，然后监控和调试，请参阅将调试器用于自定义训练容器 (p. 1868).

开源调试程序GitHub存储库

调试器 API 是通过SageMakerPython SDK，旨在构建调试器挂钩和规则配置SageMaker CreateTraining作业和 DescribeTraining作业API 操作。这些区域有：SMDebug客户端库提供注册工具钩子并通过其访问培训数据审判功能，所有这些都通过其灵活而强大的 API 操作。它支持机器学习框架 TensorFlow、PyTorchPython 3.6 和更高版本上的 XGBoost。

有关调试器的直接资源和SMDebug请参阅以下链接：

- AmazonSageMakerPython SDK 文档
- AmazonSageMaker调试程序 API
- SmDebugger 开源客户端库
- smdebug PyPI

如果你使用适用 SDK for Java 来执行SageMaker训练作业并想要配置调试器 API，请参阅以下参考资料：

- Amazon SageMaker 调试器 API 操作 (p. 1932)
- 使用 Amazon SageMaker API 配置调试器 (p. 1816)

亚马逊SageMaker调试程序架构

本主题将指导您高度概括介绍 AmazonSageMaker调试程序工作流。

调试器支持分析功能性能优化识别计算问题，例如系统瓶颈和利用不足，并帮助大规模优化硬件资源利用率。

调试器的调试功能模型优化是关于分析可能出现的非融合训练问题，这些问题同时使用优化算法最大限度地减少损失函数，例如梯度下降及其变化。

下图演示了的架构SageMaker调试程序。带粗边界线的方块是调试器用于分析训练作业的方法。

调试器将训练作业中的以下数据存储在安全的 Amazon S3 存储桶中：

- 系统指标— 硬件资源利用率数据，例如 CPU、GPU、CPU 和 GPU 内存、网络以及数据输入和输出 (I/O) 指标。
- 框架指标— 用于跟踪每次调用或采样的每个框架操作的指标，例如向前传递中的卷积层操作、向后传递中的批量标准化操作、步骤之间的数据加载器进程以及用于计算和更新损失函数的梯度下降算法操作。
- 输出张量— 在训练机器学习模型的同时，在向前和向后传递期间不断更新的标量和模型参数的集合。输出张量包括标量值（精度和损耗）和矩阵（权重、渐变、输入图层和输出图层）。

  Note

  默认情况下，调试器会监视和调试SageMaker在中配置了没有任何调试器特定参数的训练作业SageMaker估算法。调试器每 500 毫秒收集系统指标，每 500 个步骤收集基本输出张量（标量输出，如损耗和准确度）。它还运行`ProfilerReport`规则来分析系统指标并聚合 Studio Debug 见解仪表板和性能分析报告。调试器将输出数据保存到您的安全 Amazon S3 存储桶。

调试器内置规则在处理容器上运行，这些规则旨在通过处理 S3 存储桶中收集的训练数据来评估机器学习模型（请参阅处理数据和评估模型)。调试程序完全管理内置规则。您也可以创建自己的模型自定义规则来监控任何要监控的问题。

请参阅以下主题，了解使用以下方法改进模型性能的最佳实践：SageMaker调试程序。

开始使用调试器教程

以下主题将向您介绍使用调试器监控、分析和调试 SageMaker 培训作业的基础知识到高级用例的教程。探索调试器功能，了解如何使用调试器有效地调试和改进机器学习模型。

主题

- 调试程序教程视频 (p. 1788)
- 笔记本调试器示例 (p. 1789)
- 调试器高级演示和可视化 (p. 1790)

调试程序教程视频

以下视频说明了 Amazon SageMaker 调试程序使用 SageMaker Studio 和 SageMaker 笔记本实例的功能。

主题

- 使用 Amazon SageMaker 调试器分析、检测训练运行问题并获得警报 (p. 1788)
- 在 Studio 中使用 Amazon SageMaker 调试程序调试模型 (p. 1788)
- 深入了解 Amazon SageMaker 调试程序和 SageMaker 模型监控器 (p. 1789)

使用 Amazon SageMaker 调试器分析、检测训练运行问题并获得警报

艾米莉·韦伯，AmazonMachine Learning 专家 | 长度：13 分 54 秒

本教程视频将为您介绍 Amazon SageMaker 调试器，以便使用 MxNet 捕获、调试和可视化训练模型中的模型输出数据。了解 Amazon SageMaker 调试器如何通过自动捕获指标、分析训练运行情况和检测问题来使培训过程透明。

使用 Amazon SageMaker 调试器分析、检测训练运行问题并获得警报

您可以在该视频中找到该视频中的示例笔记本可视化 MXNet 训练的调试张量中的Amazon SageMaker 示例GitHub 存储库。

在 Studio 中使用 Amazon SageMaker 调试程序调试模型

朱利安·西蒙，Amazon技术宣传官 | 长度：14 分 17 秒

该教程视频说明了如何使用 Amazon SageMaker 调试程序从训练模型中捕获和检查调试信息。该视频中使用的示例训练模型是一个简单卷积神经网络 (CNN)，它基于具有 TensorFlow 后端的 Keras。使用 TensorFlow 框架中的 SageMaker 和调试程序，您可以直接使用训练脚本构建评估程序并调试训练作业。

使用 Amazon SageMaker 调试程序调试模型（第 1 部分）

您可以在作者提供的该 Studio 演示存储库中找到该视频中的示例笔记本。你需要克隆`debugger.ipynb`笔记本文件和`mnist_keras_tf.py`将脚本训练到您的 SageMaker Studio 或 SageMaker 笔记本实例。在克隆这两个文件后，请指定 `debugger.ipynb` 笔记本中的 `mnist_keras_tf.py` 文件的 `keras_script_path` 路径。例如，如果将这两个文件克隆到同一目录中，请将其设置为`keras_script_path = "mnist_keras_tf.py"`.

深入了解 Amazon SageMaker 调试程序和 SageMaker 模型监控器

朱利安·西蒙，Amazon技术宣传官 | 长度：44 分 34 秒

该视频会话介绍了调试程序和 SageMaker 模型监控器的高级功能，这些功能有助于提高生产率和模型质量。首先，该视频说明了如何使用调试程序检测和修复训练问题，对张量进行可视化以及改进模型。接下来，在 22 分 41 秒，该视频说明了如何使用 SageMaker 模型监控器在生产环境中监控模型并找出预测问题，例如缺少功能或数据偏差。最后，它提供了成本优化技巧，以帮助您充分利用机器学习预算。

使用调试程序调试模型（第 2 部分）

您可以在中找到该视频中的示例笔记本这个Amazon2020 年开发日存储库由提交人提供。

笔记本调试器示例

SageMaker 调试器示例笔记本在中提供aws/amazon-sagemaker-examples存储库。调试器示例笔记本将引导您完成调试和分析培训作业的基本到高级使用案例。

我们建议您在 SageMaker Studio 或 SageMaker 笔记本实例上运行示例笔记本，因为大多数示例都是为了在 SageMaker 生态系统中培训作业而设计，包括 Amazon EC2、Amazon S3 和 Amazon SageMaker Python SDK。

要将示例存储库克隆到 SageMaker Studio，请按照Amazon SageMaker 工作室之旅.

要在 SageMaker 笔记本实例中查找示例，请按照示 SageMaker 记本示例笔记本.

> **Important**
>
> 要使用新的调试器功能，你需要升级 SageMaker Python SDK 和`SMDebug`客户端库。在你的 iPython 内核、Jupyter 笔记本或 JupyterLab 环境中，运行以下代码以安装最新版本的库并重新启动内核。

```
import sys
import IPython
!{sys.executable} -m pip install -U sagemaker smdebug
IPython.Application.instance().kernel.do_shutdown(True)
```

用于分析训练作业的调试器示例笔记

以下列表显示了调试器示例笔记本，介绍了调试器在监控和分析各种机器学习模型、数据集和框架的训练作业方面的适应性。

笔记本标题	框架	模型	数据集	描述
Amazon SageMaker 调试器分析数据分析	TensorFlow	克拉斯 ResNet50	Cifar-10	本笔记本介绍了对 SageMaker 调试器捕获的配置文件数据的交互式分析。探索的全部功能`SMDebug`交互式分析工具。
使用 Amazon SageMaker 调试器配置机器学习培训	TensorFlow	1-D 卷积神经网络	IMDB 数据集	分析 TensorFlow 1-D CNN，以便对 IMDB 数据进行情绪分析，该数据包括标记为具有正面或负面情绪的电影评论。浏览 Studio 调试器见解和调试器性能分析报告。

笔记本标题	框架	模型	数据集	描述
使用各种分布式训练设置分析 TensorFlow ResNet 模型训练	TensorFlow	ResNet50	Cifar-10	使用各种分布式训练设置运行 TensorFlow 训练作业，使用调试器监控系统资源利用率和分析模型性能。
使用各种分布式训练设置分析 PyTorch ResNet 模型训练	PyTorch	ResNet50	Cifar-10	使用各种分布式训练设置运行 PyTorch 训练作业，使用调试器监控系统资源利用率和分析模型性能。

调试器示例笔记本用于分析模型参数

以下列表显示了调试器示例笔记本，介绍了调试器调试各种机器学习模型、数据集和框架的训练作业的适应性。

笔记本标题	框架	模型	数据集	描述
使用内置规则：使用 Amazon SageMaker 调试器	TensorFlow	卷积神经网络	MNIST	使用 Amazon SageMaker 调试器内置规则调试 TensorFlow 模型。
Amazon SageMaker 调试器-张量流 2.1	TensorFlow	ResNet50	Cifar-10	使用 Amazon SageMaker 调试器挂接配置和内置规则使用 Tensorflow 2.1 框架调试模型。
可视化 MXNet 训练的调试张量	MXNet	Gluon 卷积神经网络	MNIST 时尚	运行训练作业并将 SageMaker 调试器配置为存储此作业中的所有张量，然后在笔记本电脑中可视化这些张量。
使用 Amazon SageMaker 调试器启用竞价培训	MXNet	Gluon 卷积神经网络	MNIST 时尚	了解调试器如何从竞价型实例上的训练作业中收集张量数据，以及如何将调试器内置规则与托管竞价训练结合使用。
解释使用 Amazon SageMaker 调试器预测个人收入的 xgBoost 模型	XGBoost	XGBoost 回归	成人普查数据集	了解如何使用调试器钩子和内置规则来收集和可视化 XgBoost 回归模型中的张量数据，例如损耗值、功能和 SHAP 值。

要查找模型参数和使用案例的高级可视化效果，请参阅下一主题：调试器高级演示和可视化 (p. 1790).

调试器高级演示和可视化

以下演示将引导您使用调试器完成高级用例和可视化脚本。

主题

- 使用训练和优化 Amazon SageMaker 实验和调试器模型 (p. 1791)

- 使用 SageMaker 调试器监视卷积自动编码器模型训练 (p. 1794)
- 使用 SageMaker 调试器监视 BERT 模型训练中的注意力 (p. 1794)
- 使用 SageMaker 调试器在卷积神经网络 (CNN) 中可视化类激活图 (p. 1796)

使用训练和优化 Amazon SageMaker 实验和调试器模型

Nathalie Rauschmayr 博士Amazon应用科学家 | 长度：49 分 26 秒

通过 SageMaker 实验和调试器训练和修剪模型

了解 Amazon SageMaker 实验和调试器如何简化培训作业的管理。Amazon SageMaker 调试器提供对于训练作业的透明可见性，并将训练指标保存到 Amazon S3 存储桶中。SageMaker 实验使您能够将培训信息称为审判通过 SageMaker Studio 并支持培训作业的可视化。这有助于保持模型质量，同时根据重要性排名来减少不太重要的参数。

本视频演示模型修剪 技术，该技术使预训练的 ResNet50 和 AlexNet 模型更轻、更经济实惠，同时保持高标准的模型精度。

SageMaker Estemator 将 PyTorch 模型动物园提供的算法训练在Amazon通 Deep Learning Containers PyTorch 框架和调试器从训练过程中提取训练指标。

该视频还演示如何设置调试器自定义规则来观察修剪模型的准确性，以便触发 Amazon CloudWatch 事件和Amazon Lambda在精度达到阈值时函数，并自动停止修剪过程以避免冗余迭代。

学习目标如下：

- 了解如何使用 SageMaker 加速 ML 模型训练并提高模型质量。
- 了解如何通过自动捕获输入参数、配置和结果来管理的训练迭代。
- 了解调试器如何通过自动从卷积神经网络的权重、梯度和激活输出等指标中捕获实时张量数据，使训练过程变得透明。
- 当调试器发现问题时，使用 CloudWatch 触发 Lambda。
- 使用 SageMaker 实验和调试器掌握 SageMaker 训练过程。

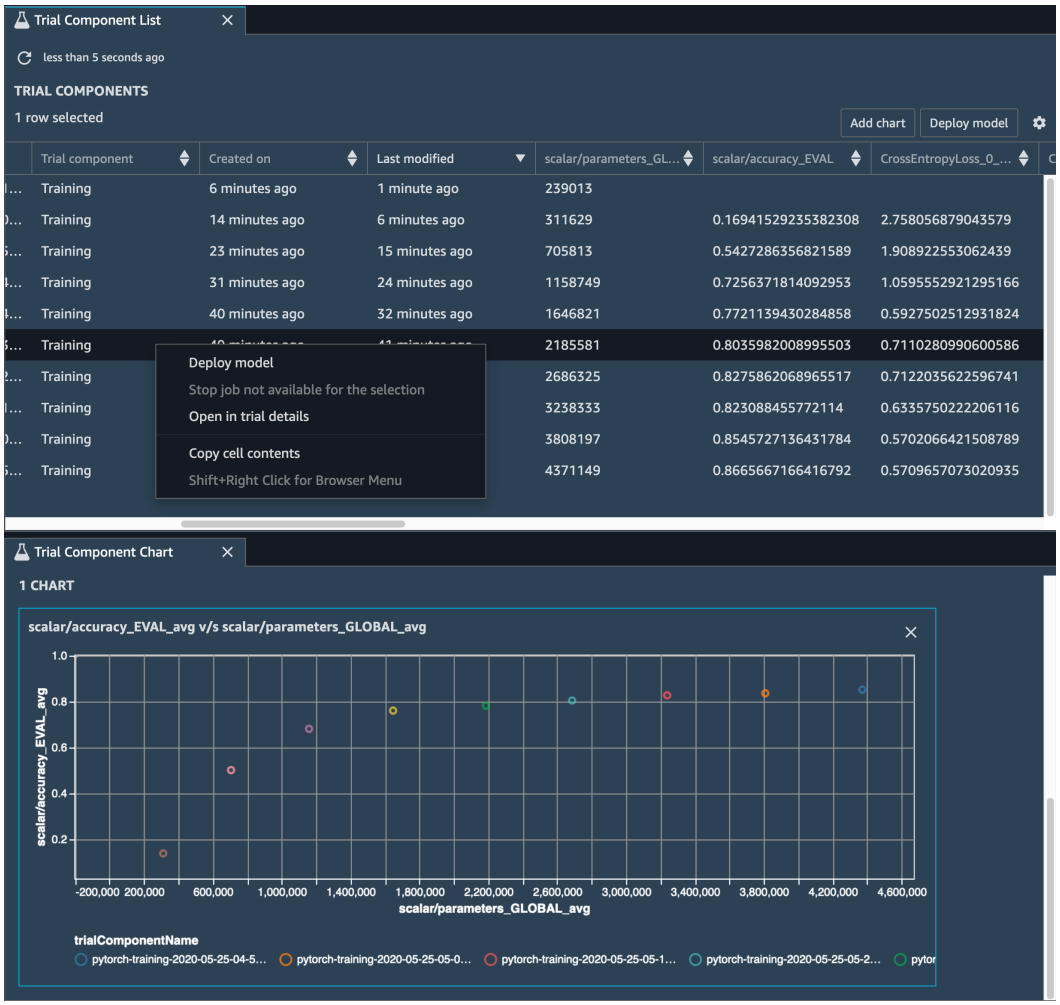
您可以从以下位置找到本视频中使用的笔记本和训练脚本。SageMaker 调试器 pyTorch 迭代模型修剪.

下图显示迭代模型修剪过程如何根据由激活输出和梯度评估的重要性等级来去掉 100 个最不显著的筛选条件，从而减小 AlexNet 的大小。

修剪过程将最初的 5000 万个参数减少到 1800 万个。它还将估计的模型大小从 201 MB 减少到 73 MB。

```
Pruning iteration: 0
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 58, 55, 55]          21,112
              ReLU-2           [-1, 58, 55, 55]               0
         MaxPool2d-3           [-1, 58, 27, 27]               0
            Conv2d-4          [-1, 166, 27, 27]         240,866
              ReLU-5          [-1, 166, 27, 27]               0
         MaxPool2d-6          [-1, 166, 13, 13]               0
            Conv2d-7          [-1, 305, 13, 13]         455,975
              ReLU-8          [-1, 305, 13, 13]               0
            Conv2d-9          [-1, 206, 13, 13]         565,676
             ReLU-10          [-1, 206, 13, 13]               0
           Conv2d-11          [-1, 217, 13, 13]         402,535
             ReLU-12          [-1, 217, 13, 13]               0
        MaxPool2d-13            [-1, 217, 6, 6]               0
AdaptiveAvgPool2d-14            [-1, 217, 6, 6]               0
          Dropout-15                 [-1, 7812]               0
           Linear-16                 [-1, 4096]      32,002,048
             ReLU-17                 [-1, 4096]               0
          Dropout-18                 [-1, 4096]               0
           Linear-19                 [-1, 4096]      16,781,312
             ReLU-20                 [-1, 4096]               0
           Linear-21                  [-1, 101]         413,797
================================================================
Total params: 50,883,321
Trainable params: 50,883,321
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 7.27
Params size (MB): 194.10
Estimated Total Size (MB): 201.95
----------------------------------------------------------------
```

您还需要跟踪模型精度，下图显示如何绘制模型修剪过程，以便根据 SageMaker Studio 中的参数数量来对模型精度的变化进行可视化。



在 SageMaker 工作室中，选择实验选项卡中，从修剪过程中选择调试器保存的张量列表，然后编写列出试验组件面板。选择所有十个迭代，然后选择添加图要创建试验组件图. 确定要部署的模型后，选择试验组件并选择菜单以执行操作，或选择选择。部署模型.

Note

要使用以下笔记本示例通过 SageMaker Studio 部署模型，请在`train`在中的函数`train.py`脚本。

```
# In the train.py script, look for the train function in line 58.
def train(epochs, batch_size, learning_rate):
    ...
        print('acc:{:.4f}'.format(correct/total))
        hook.save_scalar("accuracy", correct/total, sm_metric=True)

    # Add the following code to line 128 of the train.py script to save the pruned
 models
    # under the current SageMaker Studio model directory
    torch.save(model.state_dict(), os.environ['SM_MODEL_DIR'] + '/model.pt')
```

使用 SageMaker 调试器监视卷积自动编码器模型训练

本笔记本演示 SageMaker Debug 如何在手写数字的 MNIST 图像数据集上可视化来自不受监督（或自监督）学习过程的张量。

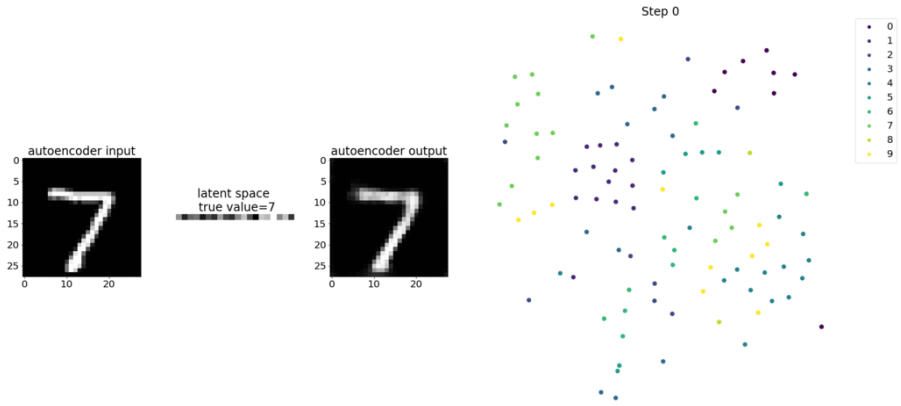
本笔记本中的训练模型是一个带有 MXNet 框架的卷积自动编码器。卷积自动编码器具有瓶颈形状的卷积神经网络，由编码器部分和解码器部分组成。

此示例中的编码器具有两个卷积层，以生成输入图像的压缩表示（潜在变量）。在这种情况下，编码器从大小（28，28）的原始输入图像中生成大小（1，20）的潜在变量，并显著减少训练数据的大小达 40 倍。

解码器有两个解卷积层，并通过重建输出图像来确保潜在变量保留关键信息。

卷积编码器为聚类算法提供更小的输入数据大小以及聚类算法（如 k 均值、k-nN 和 T 分布随机邻居嵌入 (t-SNE)）的性能。

此笔记本演示如何使用可视化调试器潜在变量，如下面的动画所示。它还演示 t-SNE 算法如何将潜在变量分为十个聚类并将它们投影到二维空间中。图像右侧的散点图颜色方案反映了真实值，以显示 BERT 模型和 t-SNE 算法将潜在变量组织到聚类中的程度。



使用 SageMaker 调试器监视 BERT 模型训练中的注意力

来自转换器的双向编码表示 (BERT) 是一种语言表示模型。正如模型名称所反映的那样，BERT 模型建立在用于自然语言处理 (NLP) 的迁移学习 和转换器模型 之上。

BERT 模型接受了不受监督任务的预训练，例如，预测句子中缺失的单词或预测自然接在前一个句子之后的下一个句子。该训练数据包含 33 亿个字词（标记）的英文文本，来自维基百科和电子书籍等来源。对于一个简单的例子，BERT 模型可以高度注意 来自主题标记的适当动词标记或代词标记。

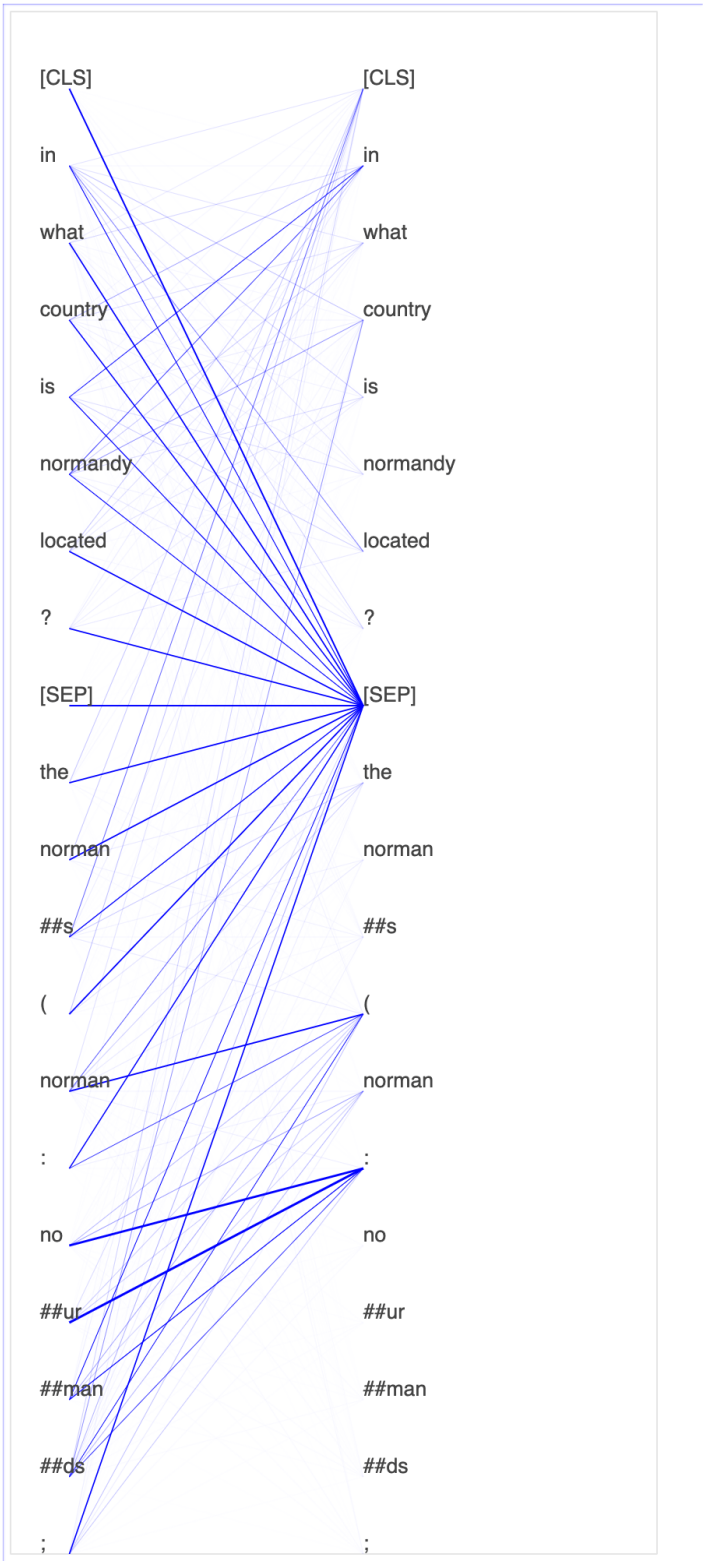
预先训练的 BERT 模型可以通过额外的输出层进行微调，以便在 NLP 任务中实现最先进的模型训练，例如自动响应问题、文本分类等。

调试器从微调过程中收集张量。在 NLP 上下文中，神经元的权重被称为。注意.

此笔记本演示如何使用来自 GluonNLP 模型动物园的预先训练的 BERT 模型了解斯坦福大学问答数据集以及如何设置 SageMaker Debug 来监视训练作业。

在查询和关键向量中绘制注意力分数 和各个神经元有助于识别模型预测不正确的原因。使用 SageMaker 调试器，您可以检索张量并绘制注意头视图在训练进行过程中实时了解模型正在学习的内容。

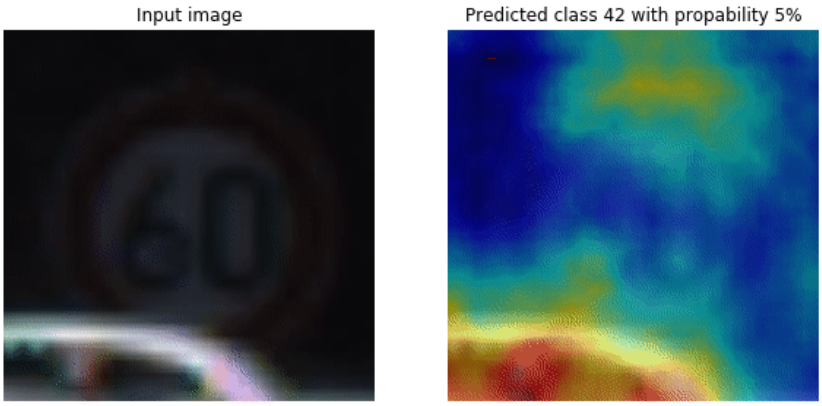
下面的动画显示在笔记本示例中提供的训练作业中十次迭代的前 20 个输入标记的注意力分数。


[CLS]
in
what
country
is
normandy
located
?
[SEP]
the
norman
##s
(
norman
:
no
##ur
##man
##ds
;
[CLS]
in
what
country
is
normandy
located
?
[SEP]
the
norman
##s
(
norman
:
no
##ur
##man
##ds
;

使用 SageMaker 调试器在卷积神经网络 (CNN) 中可视化类激活图

本笔记本演示如何使用 SageMaker Debug 绘制类激活图，以便在卷积神经网络 (CNN) 中进行图像检测和分类。在深度学习中，卷积神经网络（CNN 或 ConvNet）是一类深度神经网络，最常用于分析视觉图像。采用类激活图的应用之一是自动驾驶汽车，这需要对图像进行即时检测和分类，如交通标志、道路和障碍物。

在该笔记本中，PyTorch ResNet 模型接受德国交通标志数据集的训练，该数据集包含 40 多个类别的交通相关物体和总共 50,000 多张图像。



在训练过程中，SageMaker Debug 收集张量以实时绘制类激活图。如动画图像所示，类激活图（也称为显著图）以红色突出显示具有高激活率的区域。

使用调试器捕获的张量，您可以对激活图在模型训练期间的演变方式进行可视化。在训练作业开始时，模型首先检测左下角的边缘。随着训练进行，焦点转移到中心并检测限速标志，模型成功地将输入图像预测为类别 3，这是限速 60km/h 标志的类别，置信度为 97%。

使用亚马逊配置调试器SageMakerPython 开发工具包

要配置调试器，请使用亚马逊SageMakerPython 开发工具包并在构造时指定调试器特定的参数SageMaker估算程序。您需要配置三个参数：`profiler_config`、`debugger_hook_config`, 和`rules`.

> **Note**
>
> 默认情况下，调试器会监视和调试SageMaker在中配置了没有任何调试器特定参数的训练作业SageMaker估算程序。调试器每 500 毫秒收集系统指标，每 500 个步骤收集基本输出张量（标量输出，如损耗和准确度）。它还运行`ProfilerReport`规则来分析系统指标并聚合 Studio Debug 见解仪表板和性能分析报告。调试器将输出数据保存在安全的 S3 存储桶中。

> **Important**
>
> 要使用新的调试器功能，你需要升级SageMakerPython 开发工具包和`SMDebug`客户端库。在你的 iPython 内核中，Jupyter 笔记本，或者JupyterLab环境中，运行以下代码以安装库的最新版本并重新启动内核。

```
import sys
import IPython
!{sys.executable} -m pip install -U sagemaker smdebug
IPython.Application.instance().kernel.do_shutdown(True)
```

构造SageMaker带调试程序的估算程序

以下示例代码说明了如何构建SageMaker估计器具有调试者特定参数，具体取决于你选择的框架。在以下主题的整个文档中，您可以找到有关如何配置可以根据需要混合和匹配的调试器特定参数的更多信息。

> **Note**
>
> 以下示例代码不能直接执行。你需要继续下一节并配置调试器特定的参数。

TensorFlow

访问深度分析功能TensorFlow，目前您需要指定最新的Amazon使用 CUDA 11 进行深度学习容器映像。例如，您必须指定具体映像 URI，如以下示例代码所示：

```
# An example of constructing a SageMaker TensorFlow estimator
import boto3
import sagemaker
from sagemaker.tensorflow import TensorFlow
from sagemaker.debugger import ProfilerConfig, DebuggerHookConfig, Rule, ProfilerRule,
 rule_configs

session=boto3.session.Session()
region=session.region_name

profiler_config=ProfilerConfig(...)
debugger_hook_config=DebuggerHookConfig(...)
rules=[
    Rule.sagemaker(rule_configs.built_in_rule()),
    ProfilerRule.sagemaker(rule_configs.BuiltInRule())
]

estimator=TensorFlow(
    entry_point="directory/to/your_training_script.py",
    role=sagemaker.get_execution_role(),
    base_job_name="debugger-demo",
    instance_count=1,
    instance_type="ml.p3.2xlarge",
    image_uri=f"763104351884.dkr.ecr.{region}.amazonaws.com/tensorflow-training:2.3.1-
gpu-py37-cu110-ubuntu18.04"

    # Debugger-specific parameters
    profiler_config=profiler_config,
    debugger_hook_config=debugger_hook_config,
    rules=rules
)

estimator.fit(wait=False)
```

PyTorch

访问深度分析功能PyTorch，目前您需要指定最新的Amazon使用 CUDA 11 进行深度学习容器映像。例如，您必须指定具体映像 URI，如以下示例代码所示：

```
# An example of constructing a SageMaker PyTorch estimator
import boto3
import sagemaker
from sagemaker.pytorch import PyTorch
from sagemaker.debugger import ProfilerConfig, DebuggerHookConfig, Rule, ProfilerRule,
 rule_configs

session=boto3.session.Session()
region=session.region_name
```

```
profiler_config=ProfilerConfig(...)
debugger_hook_config=DebuggerHookConfig(...)
rules=[
    Rule.sagemaker(rule_configs.built_in_rule()),
    ProfilerRule.sagemaker(rule_configs.BuiltInRule())
]

estimator=PyTorch(
    entry_point="directory/to/your_training_script.py",
    role=sagemaker.get_execution_role(),
    base_job_name="debugger-demo",
    instance_count=1,
    instance_type="ml.p3.2xlarge",
    image_uri=f"763104351884.dkr.ecr.{region}.amazonaws.com/pytorch-training:1.6.0-gpu-
py36-cu110-ubuntu18.04",

    # Debugger-specific parameters
    profiler_config=profiler_config,
    debugger_hook_config=debugger_hook_config,
    rules=rules
)

estimator.fit(wait=False)
```

MXNet

```
# An example of constructing a SageMaker MXNet estimator
import sagemaker
from sagemaker.mxnet import MXNet
from sagemaker.debugger import ProfilerConfig, DebuggerHookConfig, Rule, ProfilerRule,
 rule_configs

profiler_config=ProfilerConfig(...)
debugger_hook_config=DebuggerHookConfig(...)
rules=[
    Rule.sagemaker(rule_configs.built_in_rule()),
    ProfilerRule.sagemaker(rule_configs.BuiltInRule())
]

estimator=MXNet(
    entry_point="directory/to/your_training_script.py",
    role=sagemaker.get_execution_role(),
    base_job_name="debugger-demo",
    instance_count=1,
    instance_type="ml.p3.2xlarge",
    framework_version="1.7.0",
    py_version="py37",

    # Debugger-specific parameters
    profiler_config=profiler_config,
    debugger_hook_config=debugger_hook_config,
    rules=rules
)

estimator.fit(wait=False)
```

> **Note**
>
> 对于 MXNet，在配置`profiler_config`参数，您只能配置系统监控。MxNet 不支持分析框架指标。

XGBoost

```
# An example of constructing a SageMaker XGBoost estimator
```

```
import sagemaker
from sagemaker.xgboost.estimator import XGBoost
from sagemaker.debugger import ProfilerConfig, DebuggerHookConfig, Rule, ProfilerRule,
 rule_configs

profiler_config=ProfilerConfig(...)
debugger_hook_config=DebuggerHookConfig(...)
rules=[
    Rule.sagemaker(rule_configs.built_in_rule()),
    ProfilerRule.sagemaker(rule_configs.BuiltInRule())
]

estimator=XGBoost(
    entry_point="directory/to/your_training_script.py",
    role=sagemaker.get_execution_role(),
    base_job_name="debugger-demo",
    instance_count=1,
    instance_type="ml.p3.2xlarge",
    framework_version="1.2-1",

    # Debugger-specific parameters
    profiler_config=profiler_config,
    debugger_hook_config=debugger_hook_config,
    rules=rules
)

estimator.fit(wait=False)
```

Note

对于 xgBoost，在配置`profiler_config`参数，您只能配置系统监控。xgBoost 不支持分析框架指标。

Generic estimator

```
# An example of constructing a SageMaker generic estimator using the XGBoost algorithm
 base image
import boto3
import sagemaker
from sagemaker.estimator import Estimator
from sagemaker import image_uris
from sagemaker.debugger import ProfilerConfig, DebuggerHookConfig, Rule, ProfilerRule,
 rule_configs

profiler_config=ProfilerConfig(...)
debugger_hook_config=DebuggerHookConfig(...)
rules=[
    Rule.sagemaker(rule_configs.built_in_rule()),
    ProfilerRule.sagemaker(rule_configs.BuiltInRule())
]

region=boto3.Session().region_name
xgboost_container=sagemaker.image_uris.retrieve("xgboost", region, "1.2-1")

estimator=Estimator(
    role=sagemaker.get_execution_role()
    image_uri=xgboost_container,
    base_job_name="debugger-demo",
    instance_count=1,
    instance_type="ml.m5.2xlarge",

    # Debugger-specific parameters
    profiler_config=profiler_config,
    debugger_hook_config=debugger_hook_config,
    rules=rules
```

```
)

estimator.fit(wait=False)
```

您可以在其中配置以下参数：

- `profiler_config`参数 — 配置调试器以从训练作业中收集系统指标和框架指标，然后保存到安全的 S3 存储桶 URI 或本地计算机中。要了解如何配置`profiler_config`参数，请参阅配置调试器监视硬件系统资源利用率 (p. 1800)和配置调试器框架分析 (p. 1801).
- `debugger_hook_config`参数 — 配置调试器以从训练作业中收集输出张量并保存到安全的 S3 存储桶 URI 或本地计算机中。要了解如何配置`debugger_hook_config`参数，请参阅配置调试程序挂钩以保存张量 (p. 1804).
- `rules`参数 — 配置此参数以启用要并行运行的调试器内置规则。这些规则会自动分析你的训练工作并找到培训问题。这些区域有：ProfilerReport规则将调试器性能分析报告保存在安全的 S3 存储桶 URI 中。要了解如何配置`rules`参数，请参阅配置调试程序内置规则 (p. 1810).

> **Note**
>
> 调试器将输出数据安全地保存在默认 S3 存储桶的子文件夹中。例如，默认 S3 存储桶 URI 的格式为`s3://sagemaker-<region>-<12digit_account_id>/<base-job-name>/<debugger-subfolders>/`. 调试器创建了三个子文件夹：`debug-output`、`profiler-output`, 和`rule-output`. 您还可以使用检索默认的 S3 存储桶 URI。SageMaker估算程序类方法 (p. 1815).

请参阅以下主题，了解如何详细配置调试器特定参数。

主题

- 配置调试器监视硬件系统资源利用率 (p. 1800)
- 配置调试器框架分析 (p. 1801)
- 在训练 Job 运行时更新调试器系统监控和框架分析配置 (p. 1804)
- 配置调试程序挂钩以保存张量 (p. 1804)
- 配置调试程序内置规则 (p. 1810)
- 关闭调试程序 (p. 1815)
- 有用SageMaker用于调试器的估计器类方法 (p. 1815)

配置调试器监视硬件系统资源利用率

要调整调试器系统监控时间间隔，请使用`ProfilerConfig`API 操作用于在构建SageMaker框架或通用估计器取决于你的偏好。

> **Note**
>
> 默认情况下，对于所有SageMaker训练作业时，调试器每 500 毫秒从 Amazon EC2 实例收集硬件系统利用率数据以进行系统监控，而不在中指定任何调试器特定参数SageMaker估算程序。
> 调试器将系统指标保存在默认 S3 存储桶中。默认 S3 存储桶 URI 的格式为`s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/profiler-output/`.

以下示例代码说明了如何设置`profiler_config`系统监视时间间隔为 1000 毫秒的参数。

```
from sagemaker.debugger import ProfilerConfig

profiler_config=ProfilerConfig(
    system_monitor_interval_millis=1000
)
```

- system_monitor_interval_millis(int) — 指定记录系统指标的监控间隔（以毫秒为单位）。可用值为 100、200、500、1000（1 秒）、5000（5 秒）和 60000（1 分钟）毫秒。默认值是 500 毫秒。

要查看系统监控的进度，请参阅打开 Amazon SageMaker 调试程序见解控制面板 (p. 1881).

配置调试器框架分析

要启用调试器框架分析，请配置framework_profile_params构建估计器时的参数。调试器框架分析收集框架指标，例如初始化阶段的数据、数据加载器进程、深度学习框架和训练脚本的 Python 运算符、步骤内部和步骤之间的详细分析，以及使用 cProfile 或 Py仪器选项。使用FrameworkProfile类，你可以配置自定义框架分析选项。

Note

在开始使用调试器框架分析之前，请验证用于构建模型的框架是否受到调试器的支持以进行框架分析。有关更多信息，请参阅 支持的框架和算法 (p. 1785)。
调试器将框架指标保存在默认 S3 存储桶中。默认 S3 存储桶 URI 的格式为s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/profiler-output/.

使用默认系统监控和框架分析开始培训 Job

以下示例代码是最简单的profiler_config参数设置以启动默认系统监控和默认框架分析。这些区域有：FrameworkProfile以下示例代码中的类在训练作业开始时启动默认框架分析。调试器框架分析包括以下选项：详细的分析、数据加载器分析和 Python 分析。

```
from sagemaker.debugger import ProfilerConfig, FrameworkProfile

profiler_config=ProfilerConfig(
    framework_profile_params=FrameworkProfile()
)
```

有了这个profiler_config参数配置时，调试器调用监控和分析的默认设置。调试器每 500 毫秒监视系统指标；使用详细分析选项分析第五步；使用数据加载器分析选项分析选项分析第七步；使用 Python 分析选项执行第九、第十和第十一步。

要查找可用的性能分析配置选项、默认参数设置以及如何配置这些选项的示例，请参阅使用默认系统监控和使用不同的分析选项的自定义框架分析开始训练 Job (p. 1802)和SageMaker调试程序 API — FrameworkProfile中的亚马逊SageMakerPython 开发工具包.

如果要更改系统监视时间间隔并启用默认框架分析，则可以指定system_monitor_interval_millis显式使用参数framework_profile_params参数。例如，要每 1000 毫秒监控一次并启用默认框架分析，请使用以下示例代码。

```
from sagemaker.debugger import ProfilerConfig, FrameworkProfile

profiler_config=ProfilerConfig(
    system_monitor_interval_millis=1000,
    framework_profile_params=FrameworkProfile()
)
```

有关的更多信息FrameworkProfile上课，请参阅SageMaker调试程序 API —FrameworkProfile中的亚马逊SageMakerPython 开发工具包.

使用目标步骤或目标时间范围的默认系统监控和自定义框架分析开始训练 Job

如果要指定目标步骤或目标时间间隔来分析训练作业，则需要为FrameworkProfile类。以下代码示例说明了如何指定用于分析的目标范围以及系统监控。

- 对于目标步长范围

使用以下示例配置，调试器每 500 毫秒监视整个训练作业（默认监控），并分析从步骤 5 到步骤 15（10 个步骤）的目标步骤范围。

```
from sagemaker.debugger import ProfilerConfig, FrameworkProfile

profiler_config=ProfilerConfig(
    framework_profile_params=FrameworkProfile(start_step=5, num_steps=10)
)
```

使用以下示例配置，调试器每 1000 毫秒监控整个训练作业，并分析从步骤 5 到步骤 15（10 个步骤）的目标步骤范围。

```
from sagemaker.debugger import ProfilerConfig, FrameworkProfile

profiler_config=ProfilerConfig(
    system_monitor_interval_millis=1000,
    framework_profile_params=FrameworkProfile(start_step=5, num_steps=10)
)
```

- 对于目标时间范围

使用以下示例配置，调试器每 500 毫秒监视整个训练作业（默认监控），并分析目标时间范围与当前 Unix 时间 600 秒相比。

```
import time
from sagemaker.debugger import ProfilerConfig, FrameworkProfile

profiler_config=ProfilerConfig(
    framework_profile_params=FrameworkProfile(start_unix_time=int(time.time()),
 duration=600)
)
```

使用以下示例配置，调试器每 1000 毫秒监视整个训练作业，并描述目标时间范围与当前 Unix 时间相比 600 秒。

```
import time
from sagemaker.debugger import ProfilerConfig, FrameworkProfile

profiler_config=ProfilerConfig(
    system_monitor_interval_millis=1000,
    framework_profile_params=FrameworkProfile(start_unix_time=int(time.time()),
 duration=600)
)
```

在目标步骤或时间范围内对所有分析选项执行框架分析。

要查找有关可用分析选项的更多信息，请参阅SageMaker调试程序 API —FrameworkProfile中的亚马逊 SageMakerPython 开发工具包.

下一节将为您展示如何为可用分析选项编写脚本。

使用默认系统监控和使用不同的分析选项的自定义框架分析开始训练 Job

您可以使用以下分析配置类来管理框架分析选项：

- DetailedProfilingConfig— 使用本机框架分析器指定目标步长或时间范围来分析框架操作 (TensorFlow和分析程序PyTorch分析器)。例如，如果使用TensorFlow，调试器挂钩启用TensorFlow要收集的分析器

TensorFlow特定的框架指标。通过详细的分析，您可以在培训作业的前一步（在第一步之前）、步骤内以及在培训作业的步骤之间对所有框架运营商进行分析。

Note

详细的分析可能会显著增加 GPU 内存消耗。我建议不要启用详细的分析功能超过几个步骤。

- DataloaderProfilingConfig— 指定目标步长或时间范围以分析深度学习框架数据加载器流程。调试器收集框架的每个数据加载器事件。

  Note

  数据加载器分析可能会降低训练性能，同时从数据加载器收集信息。我们建议不要在多个步骤中启用数据加载器分析。
  调试器已预配置为仅针对Amazon深度学习容器。调试器无法从任何其他自定义或外部培训容器中分析数据加载器流程。

- PythonProfilingConfig— 指定目标步长或时间范围来分析 Python 函数。您还可以在两个 Python 分析器之间进行选择：cProfile 和 Py仪器。
  - C 个人资料— 标准 Python 分析器。cProfile 会为训练期间调用的每个 Python 运算符收集信息。借助 cProfile，调试器可以节省每个函数调用的累积时间和注释，从而提供了 Python 函数的完整详细信息。例如，在深度学习中，最常被调用的函数可能是卷积过滤器和向后传递运算符，以及 cProfile 对其中的每一个函数进行分析。对于 cProfile 选项，您可以进一步选择计时器选项：总时间、CPU 时间和 CPU 外时间。虽然您可以分析 CPU 时间内在处理器（包括 CPU 和 GPU）上执行的每个函数调用，但也可以使用 “非 CPU 时间” 选项识别 I/O 或网络瓶颈。默认值为总时间，调试器会分析 CPU 和 CPU 外时间。借助 cProfile，您可以在分析配置文件数据时向下钻取到每个功能。
  - py仪器/仪器— Py仪器是基于采样工作的低开销 Python 分析器。使用 Py仪器选项，调试器每毫秒对性能分析事件进行一次采样。由于 Py仪器测量的是挂钟时间而不是 CPU 时间，因此 Py仪器选项是减少性能分析噪声（过滤累计速度不相关的函数调用）和捕获实际计算密集型运算符（训练模型的过程中，累计缓慢）。有了 Py仪器，你可以看到一棵函数调用树，并更好地了解缓慢的结构和根本原因。

    Note

    启用 Python 分析可能会减慢整体训练时间。cProfile 会在每次调用时分析最常被调用的 Python 运算符，因此性能分析的处理时间随调用次数而增加。对于 Py仪器而言，由于其取样机制，累计分析时间相对于时间而增加。

以下示例配置显示了当您使用具有指定值的不同分析选项时的完整结构。

```
import time
from sagemaker.debugger import (ProfilerConfig,
                                FrameworkProfile,
                                DetailedProfilingConfig,
                                DataloaderProfilingConfig,
                                PythonProfilingConfig,
                                PythonProfiler, cProfileTimer)

profiler_config=ProfilerConfig(
    system_monitor_interval_millis=500,
    framework_profile_params=FrameworkProfile(
        detailed_profiling_config=DetailedProfilingConfig(
            start_step=5,
            num_steps=1
        ),
        dataloader_profiling_config=DataloaderProfilingConfig(
            start_step=7,
            num_steps=1
        ),
        python_profiling_config=PythonProfilingConfig(
            start_step=9,
            num_steps=1,
            python_profiler=PythonProfiler.CPROFILE,
```

```
            cprofile_timer=cProfileTimer.TOTAL_TIME
        )
    )
)
```

有关可用分析选项的更多信息，请参阅DetailedProfilingConfig、DataloaderProfilingConfig, 和PythonProfilingConfig中的亚马逊SageMakerPython 开发工具包.

在训练 Job 运行时更新调试器系统监控和框架分析配置

如果要为当前正在运行的训练作业启用或更新调试器监视和分析配置，请使用以下命令SageMaker估计器扩展方法：

- 要为正在运行的训练作业启用调试器系统监控并接收调试器性能分析报告，请使用以下命令：

```
estimator.enable_default_profiling()
```

当您使用enable_default_profiling方法，调试器启动默认的系统监控，ProfileReport内置规则，该规则在培训作业结束时生成全面的概况分析报告。只有当前训练作业在没有调试器监控和分析的情况下运行时，才能调用此方法。

有关更多信息，请参阅 。估计器 .enable_default_ 分析中的亚马逊SageMakerPython 开发工具包.

- 要使用自定义配置参数启用调试器内置规则、系统监控和框架分析，请使用以下命令：

```
estimator.update_profiler(
    rules=[ProfilerRule.sagemaker(rule_configs.BuiltInRule())],
    system_monitor_interval_millis=500,
    framework_profile_params=FrameworkProfile()
)
```

有关更多信息，请参阅 。估计器 .update_profiler中的亚马逊SageMakerPython 开发工具包.

配置调试程序挂钩以保存张量

张量是每次训练迭代的向后和向前通过的更新参数的数据集合，调试器收集输出张量来分析训练作业的状态。AmazonSageMakerDebugger (调试程序)CollectionConfig和DebuggerHookConfigAPI 操作提供了将张量分组成的方法集合然后将其保存到目标 S3 存储桶。

> **Note**
>
> 默认情况下，对于所有SageMaker训练作业，调试器每 500 个步骤从训练作业中收集损失和准确度输出标量，而不在中指定任何调试器特定参数SageMaker估算程序。调试器将输出数据保存在默认 S3 存储桶中。默认 S3 存储桶 URI 的格式为s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/debug-output/.

构造SageMaker估计器，通过指定挂接配置参数启用调试器，debugger_hook_config. 以下步骤包含有关如何设置debugger_hook_config使用CollectionConfig和DebuggerHookConfigAPI 操作可将张量从训练作业中抽出并保存它们。如果您使用调试程序支持Amazon用于零脚本更改的容器，你可以简单地运行训练作业而不更改训练脚本。你也可以使用调试器来训练在任何其他任何地方运行的作业调试程序支持Amazon具有脚本模式的容器，对训练脚本进行最小的更改。

使用CollectionConfigAPI 操作

使用CollectionConfig用于配置张量集合的 API 操作。如果使用 DebuGger 支持的深度学习框架和机器学习算法，调试器提供预构建的张量集合，这些集合涵盖了各种参数的正则表达式（正则表达式）。如以下示例代码所示，添加要调试的内置张量集合。

```
from sagemaker.debugger import CollectionConfig

collection_configs=[
    CollectionConfig(name="weights"),
    CollectionConfig(name="gradients")
]
```

前面的集合设置了调试器钩子，基于默认值，每 500 个步骤保存张量"save_interval"值。

有关可用调试程序内置集合的完整列表，请参阅调试程序内置集合.

如果你想自定义内置集合，例如更改保存间隔和张量正则表达式，请使用以下命令CollectionConfig模板来调整参数。

```
from sagemaker.debugger import CollectionConfig

collection_configs=[
    CollectionConfig(
        name="tensor_collection",
        parameters={
            "key_1": "value_1",
            "key_2": "value_2",
            ...
            "key_n": "value_n"
        }
    )
]
```

有关可用参数键的更多信息，请参阅CollectionConfig中的亚马逊SageMakerPython 开发工具包. 例如，以下代码示例展示了如何在训练的不同阶段调整 “亏损” 张量收集的保存间隔：在训练阶段每 100 个步骤保存损失，在验证阶段每 10 个步骤保存一次验证损失。

```
from sagemaker.debugger import CollectionConfig

collection_configs=[
    CollectionConfig(
        name="losses",
        parameters={
            "train.save_interval": "100",
            "eval.save_interval": "10"
        }
    )
]
```

Tip

这个 tensor 集合配置对象可以用于两者DebuggerHookConfig和RuleAPI 操作。

配置调试程序挂钩以保存张量

使用DebuggerHookConfig类来创建debugger_hook_config使用对象collection_configs您在上一步中创建的对象。

```
from sagemaker.debugger import DebuggerHookConfig

debugger_hook_config=DebuggerHookConfig(
    collection_configs=collection_configs
)
```

调试程序将模型训练输出张量保存到默认 S3 存储桶中。默认 S3 存储桶 URI 的格式为s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/debug-output/.

如果要指定确切的 S3 存储桶 URI，请使用以下代码示例：

```
from sagemaker.debugger import DebuggerHookConfig

debugger_hook_config=DebuggerHookConfig(
    s3_output_path="specify-your-s3-bucket-uri"
    collection_configs=collection_configs
)
```

有关更多信息，请参阅 。DebuggerHookConfig中的亚马逊SageMakerPython 开发工具包.

用于配置调试器挂钩的示例笔记本和代码示例

以下各节提供了如何使用调试器钩子保存、访问和可视化输出张量的笔记本和代码示例。

主题

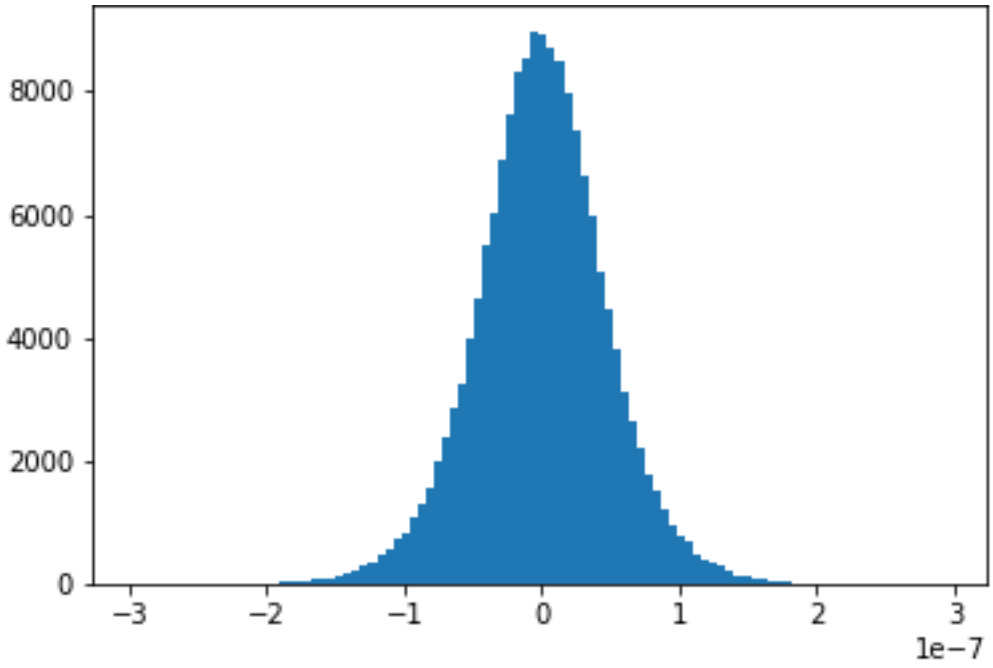
- 张量可视化示例笔记本 (p. 1806)
- 使用调试程序内置集合保存张量 (p. 1807)
- 使用调试程序修改的内置集合保存张量 (p. 1808)
- 使用调试程序自定义集合保存张量 (p. 1809)

张量可视化示例笔记本

以下两个笔记本示例显示了亚马逊的高级使用SageMaker用于可视化张量的调试器。调试器为训练深度学习模型提供了透明的视图。

- 中的交互式张量分析SageMakerMXNet 工作室笔记本

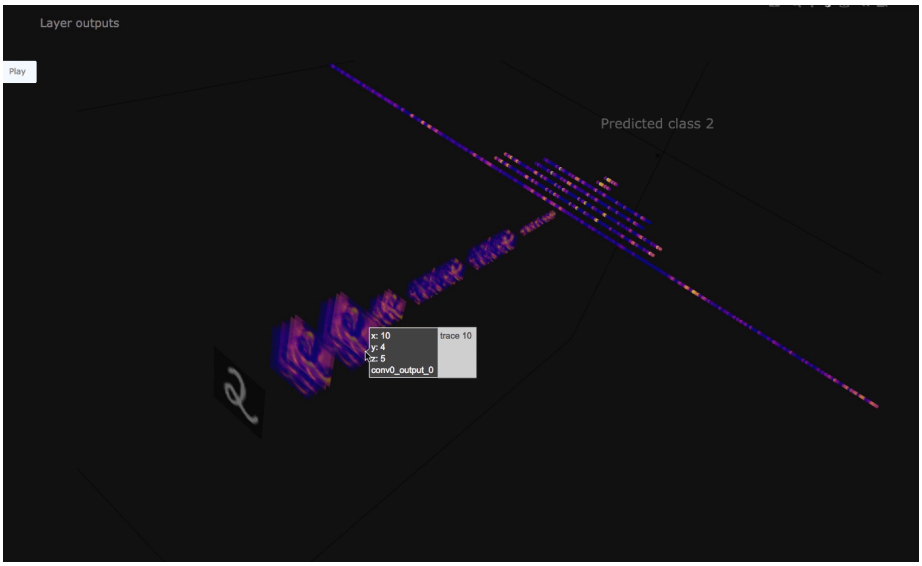
  该笔记本示例说明了如何使用 Amazon 可视化保存的张量。SageMaker调试程序。通过可视化张量，您可以在训练深度学习算法的同时，看到张量值如何变化。该笔记本包含一个具有未正确配置的神经网络的训练作业，并使用SageMaker调试程序用于聚合和分析张量，包括梯度、激活输出和权重。例如，下图显示了出现梯度消失问题的卷积层的梯度分布情况。



该笔记本还说明了正确的初始超参数设置如何生成相同的张量分布图，从而改进了训练过程。

- 从 MXNet 模型训练中可视化和调试张量

  该笔记本示例说明了如何使用亚马逊保存 MXNet Gluon 模型训练作业中的张量并进行可视化。SageMaker调试程序。它说明了调试程序设置为将所有张量保存到 Amazon S3 存储桶中并检索ReLu可视化的激活输出。下图显示了一个可视化的三维可视化。ReLu激活输出。颜色方案设置为蓝色以表示接近于 0 的值，设置为黄色以表示接近于 1 的值。



  在这本笔记本中，`TensorPlot`从导入类`tensor_plot.py`旨在绘制采用二维图像作为输入的卷积神经网络 (CNN)。这些区域有：`tensor_plot.py`该笔记本附带的脚本使用调试程序检索张量并对 CNN 进行可视化。你可以在上运行这个笔记本SageMakerStudio 可重现张量可视化并实施您自己的卷积神经网络模型。

- 中的实时张量分析SageMaker带 MXNet 的笔记本

  此示例将指导您完成在亚马逊中安装发出张量所需的组件SageMaker训练作业并使用调试器 API 操作在训练运行期间访问这些张量。gluon CNN 模型是根据时尚 MNIST 数据集进行训练的。在作业运行期间，您将看到调试器如何从 100 个批次中的每个批次中检索第一个卷积层的激活输出并对其进行可视化。此外，这将向您展示如何在作业完成后可视化权重。

使用调试程序内置集合保存张量

您可以使用使用内置张量集合。`CollectionConfig`API 然后使用`DebuggerHookConfig`API。以下示例说明了如何使用调试程序挂钩配置的默认设置来构建SageMaker TensorFlow估算程序。你也可以将它用于 MxNet，PyTorch，以及 xgBoost 估计器。

> **Note**
>
> 在以下示例代码中，`s3_output_path`对于的参数`DebuggerHookConfig`为可选项。如果你没有指定它，调试器将张量保存在`s3://<output_path>/debug-output/`，其中，`<output_path>`是的默认输出路径SageMaker训练作业。例如：
>
> ```
> "s3://sagemaker-us-east-1-111122223333/sagemaker-debugger-training-YYYY-MM-DD-HH-MM-SS-123/debug-output"
> ```

```
import sagemaker
from sagemaker.tensorflow import TensorFlow
```

```
from sagemaker.debugger import DebuggerHookConfig, CollectionConfig

# use Debugger CollectionConfig to call built-in collections
collection_configs=[
        CollectionConfig(name="weights"),
        CollectionConfig(name="gradients"),
        CollectionConfig(name="losses"),
        CollectionConfig(name="biases")
    ]

# configure Debugger hook
# set a target S3 bucket as you want
sagemaker_session=sagemaker.Session()
BUCKET_NAME=sagemaker_session.default_bucket()
LOCATION_IN_BUCKET='debugger-built-in-collections-hook'

hook_config=DebuggerHookConfig(
    s3_output_path='s3://{BUCKET_NAME}/{LOCATION_IN_BUCKET}'.
                    format(BUCKET_NAME=BUCKET_NAME,
                           LOCATION_IN_BUCKET=LOCATION_IN_BUCKET),
    collection_configs=collection_configs
)

# construct a SageMaker TensorFlow estimator
sagemaker_estimator=TensorFlow(
    entry_point='directory/to/your_training_script.py',
    role=sm.get_execution_role(),
    base_job_name='debugger-demo-job',
    instance_count=1,
    instance_type="ml.m4.xlarge",
    framework_version="2.3.0",
    py_version="py37",

    # debugger-specific hook argument below
    debugger_hook_config=hook_config
)

sagemaker_estimator.fit()
```

要查看调试程序内置集合列表，请参阅调试程序内置集合.

使用调试程序修改的内置集合保存张量

您可以使用CollectionConfigAPI 操作。以下示例说明了如何调整内置文件。losses收集并构造 SageMaker TensorFlow估算程序。您也可以将其用于 MXNet，PyTorch，以及 xgBoost 估计器。

```
import sagemaker
from sagemaker.tensorflow import TensorFlow
from sagemaker.debugger import DebuggerHookConfig, CollectionConfig

# use Debugger CollectionConfig to call and modify built-in collections
collection_configs=[
    CollectionConfig(
                name="losses",
                parameters={"save_interval": "50"})]

# configure Debugger hook
# set a target S3 bucket as you want
sagemaker_session=sagemaker.Session()
BUCKET_NAME=sagemaker_session.default_bucket()
LOCATION_IN_BUCKET='debugger-modified-collections-hook'

hook_config=DebuggerHookConfig(
    s3_output_path='s3://{BUCKET_NAME}/{LOCATION_IN_BUCKET}'.
```

```
                    format(BUCKET_NAME=BUCKET_NAME,
                           LOCATION_IN_BUCKET=LOCATION_IN_BUCKET),
    collection_configs=collection_configs
)

# construct a SageMaker TensorFlow estimator
sagemaker_estimator=TensorFlow(
    entry_point='directory/to/your_training_script.py',
    role=sm.get_execution_role(),
    base_job_name='debugger-demo-job',
    instance_count=1,
    instance_type="ml.m4.xlarge",
    framework_version="2.3.0",
    py_version="py37",

    # debugger-specific hook argument below
    debugger_hook_config=hook_config
)

sagemaker_estimator.fit()
```

有关完整列表`CollectionConfig`参数，请参阅Debugger (调试程序)CollectionConfigAPI.

使用调试程序自定义集合保存张量

您还可以选择保存减少的张量数而不是全部张量（例如，如果要减少 Amazon S3 存储桶中保存的数据量）。以下示例说明了如何自定义调试程序挂钩配置以指定要保存的目标张量。您可以将此用于 TensorFlow，MXNet，PyTorch，以及 xgBoost 估计器。

```
import sagemaker
from sagemaker.tensorflow import TensorFlow
from sagemaker.debugger import DebuggerHookConfig, CollectionConfig

# use Debugger CollectionConfig to create a custom collection
collection_configs=[
        CollectionConfig(
            name="custom_activations_collection",
            parameters={
                "include_regex": "relu|tanh", # Required
                "reductions": "mean,variance,max,abs_mean,abs_variance,abs_max"
            })
    ]

# configure Debugger hook
# set a target S3 bucket as you want
sagemaker_session=sagemaker.Session()
BUCKET_NAME=sagemaker_session.default_bucket()
LOCATION_IN_BUCKET='debugger-custom-collections-hook'

hook_config=DebuggerHookConfig(
    s3_output_path='s3://{BUCKET_NAME}/{LOCATION_IN_BUCKET}'.
                    format(BUCKET_NAME=BUCKET_NAME,
                           LOCATION_IN_BUCKET=LOCATION_IN_BUCKET),
    collection_configs=collection_configs
)

# construct a SageMaker TensorFlow estimator
sagemaker_estimator=TensorFlow(
    entry_point='directory/to/your_training_script.py',
    role=sm.get_execution_role(),
    base_job_name='debugger-demo-job',
    instance_count=1,
    instance_type="ml.m4.xlarge",
    framework_version="2.3.0",
```

```
    py_version="py37",

    # debugger-specific hook argument below
    debugger_hook_config=hook_config
)

sagemaker_estimator.fit()
```

有关完整列表CollectionConfig参数，请参阅Debugger (调试程序)CollectionConfig.

配置调试程序内置规则

Amazon SageMaker Debugger 规则将分析模型训练期间发出的张量。调试程序提供RuleAPI 操作，用于监控训练作业进度和错误，以便成功训练模型。例如，规则可以检测梯度是否变得太大或太小，检测模型是否过度拟合还是过度训练，以及训练任务是否不降低损失功能和改善。要查看可用内置规则的完整列表，请参阅调试程序内置规则列表 (p. 1826).

Note

内置规则是在 Amazon SageMaker 处理容器中准备的，并由 SageMaker 调试器完全管理。默认情况下，调试器启动剖析器报告 (p. 1828)规则适用于所有 SageMaker 训练作业，而不为 SageMaker 估计器指定任何调试者特定的规则参数。ProfilerReport 规则调用以下所有内置规则来监视系统瓶颈和分析框架指标：

- BatchSize (p. 1829)
- CPUBottleneck (p. 1831)
- GPUMemoryIncrease (p. 1832)
- IOBottleneck (p. 1833)
- LoadBalancing (p. 1834)
- LowGPUUtilization (p. 1834)
- OverallSystemUsage (p. 1835)
- MaxInitializationTime (p. 1836)
- OverallFrameworkMetrics (p. 1836)
- StepOutlier (p. 1837)

调试器将性能分析报告保存在默认 S3 存储桶中。默认 S3 存储桶 URI 的格式为s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/rule-output/. 有关如何下载性能分析报告的更多信息，请参阅SageMaker 调试器性能分析报告 (p. 1897). SageMaker 调试器完全管理内置规则并行分析你的训练作业。有关计费的更多信息，请参阅Amazon SageMaker Studio 不收取额外费用的 部分Amazon SageMaker 定价页.

在以下主题中，了解如何使用调试器内置规则。

主题

- 将调试器内置规则与默认参数设置一起使用 (p. 1810)
- 使用带有自定义参数值的调试器内置规则 (p. 1811)
- 用于配置调试器规则的示例笔记本和代码示例 (p. 1812)

将调试器内置规则与默认参数设置一起使用

要在估计器中指定调试器内置规则，你需要配置rules列出对象。以下示例代码显示列出 Debugger 内置规则的基本结构：

```
from sagemaker.debugger import Rule, ProfilerRule, rule_configs

rules=[
    ProfilerRule.sagemaker(rule_configs.BuiltInProfilerRuleName_1()),
    ProfilerRule.sagemaker(rule_configs.BuiltInProfilerRuleName_2()),
    ...
    ProfilerRule.sagemaker(rule_configs.BuiltInProfilerRuleName_n()),
    Rule.sagemaker(rule_configs.built_in_rule_name_1()),
    Rule.sagemaker(rule_configs.built_in_rule_name_2()),
    ...
    Rule.sagemaker(rule_configs.built_in_rule_name_n())
]
```

有关内置规则的默认参数值和说明的更多信息，请参阅调试程序内置规则列表 (p. 1826).

例如，要检查模型的整体训练性能和进度，请使用以下内置规则配置构建一个 SageMaker 估计器。

```
from sagemaker.debugger import Rule, rule_configs

rules=[
    ProfilerRule.sagemaker(rule_configs.ProfilerReport()),
    Rule.sagemaker(rule_configs.loss_not_decreasing()),
    Rule.sagemaker(rule_configs.overfit()),
    Rule.sagemaker(rule_configs.overtraining()),
    Rule.sagemaker(rule_configs.stalled_training_rule())
]
```

当你开始训练作业时，调试器每 500 毫秒收集系统资源利用率数据，默认情况下每 500 个步骤收集损失和准确度值。调试器分析资源利用率，以确定模型是否存在瓶颈问题。这些区域有：loss_not_decreasing、overfit、overtraining, 和stalled_training_rule监控模型是否在没有这些训练问题的情况下优化损失功能。如果规则检测到训练异常，则规则评估状态将更改为IssueFound. 您可以设置自动操作，例如使用 Amazon CloudWatch Events 通知培训问题和停止培训作业Amazon Lambda. 有关更多信息，请参阅 对 Amazon SageMaker 调试器规则的操作 (p. 1872)。

使用带有自定义参数值的调试器内置规则

如果你想调整内置的规则参数值并自定义张量集合正则表达式，请配置base_config和rule_parameters的参数ProfilerRule.sagemaker和Rule.sagemaker类方法。如果是Rule.sagemaker类方法，你也可以通过collections_to_save参数。关于如何使用CollectionConfig课程提供在 使用CollectionConfigAPI 操作 (p. 1804).

使用以下配置模板进行内置规则来自定义参数值。通过根据需要更改规则参数，您可以调整要触发的规则的敏感度。

- 这些区域有：base_config参数是你调用内置规则方法的地方。
- 这些区域有：rule_parameters参数是调整中列出的内置规则的默认键值调试程序内置规则列表 (p. 1826).
- 这些区域有：collections_to_save参数通过一个张量配置CollectionConfigAPI，它需要name和parameters参数。
  - 查找可用的张量集合name，请参阅内置 Tensor 集合.
  - 有关可调整的完整列表parameters，请参阅调试程序 CollectionConfig API.

有关调试程序规则类、方法和参数的更多信息，请参阅SageMaker 调试器规则类中的Amazon SageMaker Python 开发工具包.

```
from sagemaker.debugger import Rule, ProfilerRule, rule_configs, CollectionConfig
```

```
rules=[
    ProfilerRule.sagemaker(
        base_config=rule_configs.BuiltInProfilerRuleName(),
        rule_parameters={
                "key": "value"
        }
    )
    Rule.sagemaker(
        base_config=rule_configs.built_in_rule_name(),
        rule_parameters={
                "key": "value"
        }
        collections_to_save=[
            CollectionConfig(
                name="tensor_collection_name",
                parameters={
                    "key": "value"
                }
            )
        ]
    )
]
```

每个规则的参数描述和值自定义示例都在调试程序内置规则列表 (p. 1826).

用于配置调试器规则的示例笔记本和代码示例

在以下各节中，提供了有关如何使用调试器规则监控 SageMaker 培训作业的笔记本和代码示例。

主题

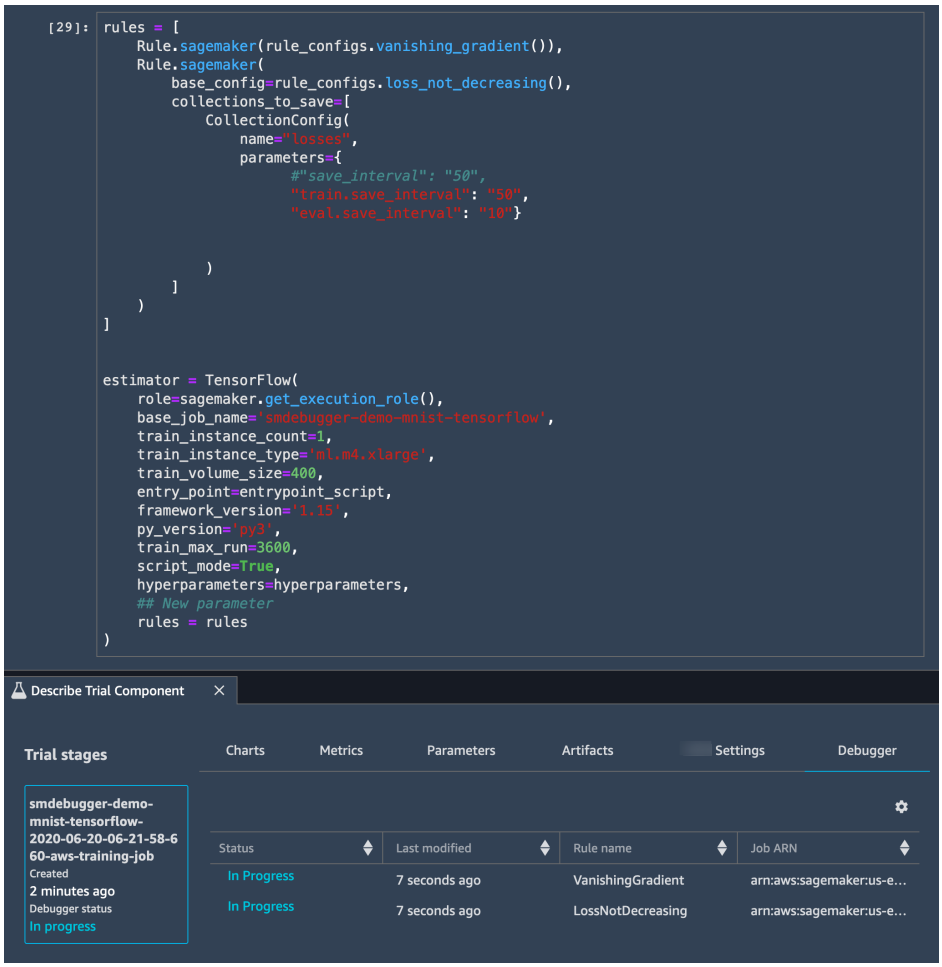
- 笔记本调试程序内置规则示例 (p. 1812)
- 调试程序内置规则示例代码 (p. 1813)
- 将调试程序内置规则与参数修改一起 (p. 1814)

笔记本调试程序内置规则示例

以下示例笔记本展示了在 Amazon SageMaker 运行训练作业时如何使用调试器内置规则：

- 将 SageMaker 调试器内置规则与 TensorFlow 一起使用
- 将 SageMaker 调试器内置规则与托管竞价训练和 MXNet 一起使用
- 将 SageMaker 调试器内置规则与 XGBoost 一起使用
- 使用 SageMaker 调试器内置规则和参数修改，以便使用 xgBoost 进行实时训练作业分析

在 SageMaker Studio 中运行示例笔记本时，您可以找到在Studio 实验列表选项卡。例如，如以下屏幕截图所示，您可以找到并打开描述试验组件你当前训练工作的窗口。在调试器选项卡上，你可以检查调试器规则是否`vanishing_gradient()`和`loss_not_decreasing()`, 正在同时监测培训课程. 有关如何在 Studio UI 中找到训练工作试用组件的完整说明，请参阅SageMaker Studio-查看实验、试验和试验组件.



在 SageMaker 环境中使用调试器内置规则有两种方法：在准备时部署内置规则或根据需要调整其参数。以下主题向您演示如何将内置规则与示例代码一起使用。

调试程序内置规则示例代码

以下代码示例说明了如何使用`Rule.sagemaker`方法。要指定要运行的内置规则，请使用`rules_configs`调用内置规则的 API 操作。要查找调试程序内置规则和默认参数值的完整列表，请参阅调试程序内置规则列表 (p. 1826).

```
import sagemaker
from sagemaker.tensorflow import TensorFlow
from sagemaker.debugger import Rule, CollectionConfig, rule_configs

# call built-in rules that you want to use.
built_in_rules=[
            Rule.sagemaker(rule_configs.vanishing_gradient())
            Rule.sagemaker(rule_configs.loss_not_decreasing())
]

# construct a SageMaker estimator with the Debugger built-in rules
sagemaker_estimator=TensorFlow(
    entry_point='directory/to/your_training_script.py',
    role=sm.get_execution_role(),
    base_job_name='debugger-built-in-rules-demo',
    instance_count=1,
    instance_type="ml.m4.xlarge",
```

```
    framework_version="2.1.0",
    py_version="py3",

    # debugger-specific arguments below
    rules=built_in_rules
)
sagemaker_estimator.fit()
```

Note

调试器内置规则与您的训练作业并行运行。培训作业的内置规则容器的最大数量为 20 个。

有关调试程序规则类、方法和参数的更多信息，请参阅SageMaker 调试器规则类中的Amazon SageMaker Python 开发工具包.

要查找如何调整调试器规则参数的示例，请参阅以下内容将调试程序内置规则与参数修改一起 (p. 1814)部分。

将调试程序内置规则与参数修改一起

以下代码示例说明了调整参数的内置规则的结构。在此示例中，stalled_training_rule收集losses每 50 个步骤从训练工作中收集张量，每 10 个步骤进行评估阶段。如果训练过程开始停滞并且在 120 秒内没有收集张量输出，那么stalled_training_rule停止训练任务。

```
import sagemaker
from sagemaker.tensorflow import TensorFlow
from sagemaker.debugger import Rule, CollectionConfig, rule_configs

# call the built-in rules and modify the CollectionConfig parameters

base_job_name_prefix= 'smdebug-stalled-demo-' + str(int(time.time()))

built_in_rules_modified=[
    Rule.sagemaker(
        base_config=rule_configs.stalled_training_rule(),
        rule_parameters={
                'threshold': '120',
                'training_job_name_prefix': base_job_name_prefix,
                'stop_training_on_fire' : 'True'
        }
        collections_to_save=[
            CollectionConfig(
                name="losses",
                parameters={
                      "train.save_interval": "50"
                      "eval.save_interval": "10"
                }
            )
        ]
    )
]

# construct a SageMaker estimator with the modified Debugger built-in rule
sagemaker_estimator=TensorFlow(
    entry_point='directory/to/your_training_script.py',
    role=sm.get_execution_role(),
    base_job_name=base_job_name_prefix,
    instance_count=1,
    instance_type="ml.m4.xlarge",
    framework_version="2.1.0",
    py_version="py3",

    # debugger-specific arguments below
```

```
    rules=built_in_rules_modified
)
sagemaker_estimator.fit()
```

要获得调试器内置规则的高级配置，请使用CreateTrainingJobAPI，请参阅使用 Amazon SageMaker API 配置调试器 (p. 1816).

关闭调试程序

如果要完全关闭调试程序，请执行以下操作之一：

- 开始训练作业之前，请执行以下操作：

  要禁用监控和分析功能，请包括disable_profiler参数到你的估计器并将其设置为True.

  > Warning
  >
  > 如果你禁用它，你将无法查看综合的 Studio 调试器见解仪表板和自动生成的性能分析报告。

  要禁用调试，请将debugger_hook_config要参数False.

  > Warning
  >
  > 如果禁用该选项，您将无法收集输出张量，并且无法调试模型参数。

  ```
  estimator=Estimator(
      ...
      disable_profiler=True
      debugger_hook_config=False
  )
  ```

  有关调试程序特定参数的更多信息，请参阅SageMaker估算程序中的亚马逊SageMakerPython 开发工具包.

- 在训练作业运行时，请执行以下操作：

  要在训练作业运行期间禁用监视和分析，请使用以下估ator class 方法：

  ```
  estimator.disable_profiling()
  ```

  要仅禁用框架分析并保持系统监控，请使用update_profilier方法：

  ```
  estimator.update_profiler(disable_framework_metrics=true)
  ```

  有关估计程序扩展方法的更多信息，请参阅估计器 .禁用_分析和估计器 .update_profiler中的类方法亚马逊SageMakerPython 开发工具包文档中）。

有用SageMaker用于调试器的估计器类方法

以下估算器类方法对于访问您的SageMaker训练工作信息和检索调试器收集的训练数据的输出路径。以下方法可以在您使用estimator.fit()方法。

- 检查一个的基本 S3 存储桶 URISageMaker训练作业：

  ```
  estimator.output_path
  ```

- 要检查一个的基本作业名称SageMaker训练作业：

```
estimator.latest_training_job.job_name
```

- 要查看完整`CreateTrainingJob`的 API 操作配置SageMaker训练作业：

```
estimator.latest_training_job.describe()
```

- 要检查调试程序规则的完整列表，请在SageMaker培训作业正在运行：

```
estimator.latest_training_job.rule_job_summary()
```

- 要检查保存模型参数数据（输出张量）的 S3 存储桶 URI，请执行以下操作：

```
estimator.latest_job_debugger_artifacts_path()
```

- 要检查 S3 存储桶 URI 保存模型性能数据（系统和框架指标）的位置，请执行以下操作：

```
estimator.latest_job_profiler_artifacts_path()
```

- 要检查调试器规则配置以调试输出张量，请执行以下操作：

```
estimator.debugger_rule_configs
```

- 查看调试器规则列表以便在SageMaker培训作业正在运行：

```
estimator.debugger_rules
```

- 要检查用于监视和分析系统和框架指标的调试器规则配置：

```
estimator.profiler_rule_configs
```

- 查看用于监视和分析的调试器规则列表SageMaker培训作业正在运行：

```
estimator.profiler_rules
```

有关的更多信息SageMaker估计器类及其方法，请参见估算程序 API中的亚马逊SageMakerPython 开发工具包.

使用 Amazon SageMaker API 配置调试器

上述主题重点介绍了通过 Amazon SageMaker Python SDK 使用调试器，这是一个包装器Amazon SDK for Python (Boto3)以及 SageMaker API 操作。这提供了访问 Amazon SageMaker API 操作的高级体验。如果您需要使用手动配置 SageMaker API 操作AmazonBoto3 或Amazon Command Line Interface(CLI) 对于其他 SDK，例如 Java、Go 和 C++，本指南介绍了如何配置CreateTrainingJob、更新培训工作、调试器配置 API 以及使用调试器内置和自定义规则的参数。

主题

- JSON (Amazon CLI) (p. 1817)
- AmazonBoto3 (p. 1821)

JSON (Amazon CLI)

可以使用为训练作业配置 Amazon SageMaker 调试器内置规则debugHookconfig、调试规则配置、ProfilerConfig, 和ProfilerRules 配置通过 SageMaker 的对象CreateTrainingJobAPI 操作。你需要在`RuleEvaluatorImage`参数，以下示例向您介绍如何设置要请求的 JSON 字符串。CreateTrainingJob.

下面的代码显示了一个完整的 JSON 模板，用于运行具有所需设置和调试器配置的训练作业。将模板保存为 JSON 文件到工作目录中，然后使用AmazonCLI。例如，将以下代码保存为`debugger-training-job-cli.json`.

> **Note**
>
> 确保使用正确的 Docker 容器镜像。要查找Amazon深度学习容器映像，请参阅可用的 Deep Learning Containers 映像. 要查找用于使用调试器规则的 Docker 映像的完整列表，请参阅将调试程序 Docker 映像用于内置规则或自定义规则 (p. 1933).

```
{
   "TrainingJobName": "debugger-aws-cli-test",
   "RoleArn": "arn:aws:iam::111122223333:role/service-role/AmazonSageMaker-ExecutionRole-
YYYYMMDDT123456",
   "AlgorithmSpecification": {
      // Specify a training Docker container image URI (Deep Learning Container or your own
 training container) to TrainingImage.
      "TrainingImage": "763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-
training:2.4.1-gpu-py37-cu110-ubuntu18.04",
      "TrainingInputMode": "File",
      "EnableSageMakerMetricsTimeSeries": false
   },
   "HyperParameters": {
      "sagemaker_program": "entry_point/tf-hvd-train.py",
      "sagemaker_submit_directory": "s3://sagemaker-us-west-2-111122223333/debugger-boto3-
profiling-test/source.tar.gz"
   },
   "OutputDataConfig": {
      "S3OutputPath": "s3://sagemaker-us-west-2-111122223333/debugger-aws-cli-test/output"
   },
   "DebugHookConfig": {
      "S3OutputPath": "s3://sagemaker-us-west-2-111122223333/debugger-aws-cli-test/debug-
output",
      "CollectionConfigurations": [
         {
            "CollectionName": "losses",
            "CollectionParameters" : {
                "train.save_interval": "50"
            }
         }
      ]
   },
   "DebugRuleConfigurations": [
      {
         "RuleConfigurationName": "LossNotDecreasing",
         "RuleEvaluatorImage": "895741380848.dkr.ecr.us-west-2.amazonaws.com/sagemaker-
debugger-rules:latest",
         "RuleParameters": {"rule_to_invoke": "LossNotDecreasing"}
      }
   ],
   "ProfilerConfig": {
      "S3OutputPath": "s3://sagemaker-us-west-2-111122223333/debugger-aws-cli-test/
profiler-output",
      "ProfilingIntervalInMilliseconds": 500,
      "ProfilingParameters": {
          "DataloaderProfilingConfig": "{\"StartStep\": 5, \"NumSteps\": 3, \"MetricsRegex
\": \".*\", }",
          "DetailedProfilingConfig": "{\"StartStep\": 5, \"NumSteps\": 3, }",
```

```
          "PythonProfilingConfig": "{\"StartStep\": 5, \"NumSteps\": 3, \"ProfilerName\":
 \"cprofile\", \"cProfileTimer\": \"total_time\"}",
          "LocalPath": "/opt/ml/output/profiler/"
       }
   },
   "ProfilerRuleConfigurations": [
      {
         "RuleConfigurationName": "ProfilerReport",
         "RuleEvaluatorImage": "895741380848.dkr.ecr.us-west-2.amazonaws.com/sagemaker-
debugger-rules:latest",
         "RuleParameters": {"rule_to_invoke": "ProfilerReport"}
      }
   ],
   "ResourceConfig": {
      "InstanceType": "ml.p3.8xlarge",
      "InstanceCount": 1,
      "VolumeSizeInGB": 30
   },

   "StoppingCondition": {
      "MaxRuntimeInSeconds": 86400
   }
}
```

保存 JSON 文件后，在您的终端中运行以下命令。（使用！如果你使用 Jupyter 笔记本，则在行的开头。）

```
aws sagemaker create-training-job --cli-input-json file://debugger-training-job-cli.json
```

配置调试器规则以调试模型参数

以下代码示例演示了如何配置内置VanishingGradient使用此 SageMaker API 的规则。

启用调试器收集输出张量

按如下方式指定调试器挂接配置：

```
"DebugHookConfig": {
    "S3OutputPath": "s3://<default-bucket>/<training-job-name>/debug-output",
    "CollectionConfigurations": [
        {
            "CollectionName": "gradients",
            "CollectionParameters" : {
                "save_interval": "500"
            }
        }
    ]
}
```

这将使训练工作保存张量集合，gradients，每save_interval500 个步骤。要查找可用CollectionName值，请参阅调试器内置集合中的smDebug 客户端库文档. 要查找可用CollectionParameters参数键和值，请参阅sagemaker.debugger.CollectionConfig中的类SageMaker Python 开发工具包文档.

启用调试器规则来调试输出张量

以下DebugRuleConfigurationsAPI 示例演示如何运行内置VanishingGradient对已保存的规则gradients收集。

```
"DebugRuleConfigurations": [
    {
        "RuleConfigurationName": "VanishingGradient",
```

```
        "RuleEvaluatorImage": "503895931360.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
debugger-rules:latest",
        "RuleParameters": {
            "rule_to_invoke": "VanishingGradient",
            "threshold": "20.0"
        }
    }
]
```

通过使用类似于该示例中的配置，调试器使用VanishingGradient关于收集的规则gradients张量。要查找用于使用调试器规则的 Docker 映像的完整列表，请参阅将调试程序 Docker 映像用于内置规则或自定义规则 (p. 1933). 要查找的键值对RuleParameters，请参阅调试程序内置规则列表 (p. 1826).

为分析系统和框架指标配置调试器内置规则

以下示例代码显示如何指定 ProfilerConfig API 操作以启用收集系统和框架指标。

启用调试器分析以收集系统和框架指标

Target Step

```
"ProfilerConfig": {
    // Optional. Path to an S3 bucket to save profiling outputs
    "S3OutputPath": "s3://<default-bucket>/<training-job-name>/profiler-output",
    // Available values for ProfilingIntervalInMilliseconds: 100, 200, 500, 1000 (1
 second), 5000 (5 seconds), and 60000 (1 minute) milliseconds.
    "ProfilingIntervalInMilliseconds": 500,
    "ProfilingParameters": {
        "DataloaderProfilingConfig": "{ \"StartStep\": 5, \"NumSteps\": 3,
 \"MetricsRegex\": \".*\" }",
        "DetailedProfilingConfig": "{ \"StartStep\": 5, \"NumSteps\": 3 }",
        // For PythonProfilingConfig,
        // available ProfilerName options: cProfile, Pyinstrument
        // available cProfileTimer options only when using cProfile: cpu, off_cpu,
 total_time
        "PythonProfilingConfig": "{ \"StartStep\": 5, \"NumSteps\": 3, \"ProfilerName
\": \"cProfile\", \"cProfileTimer\": \"total_time\" }",
        // Optional. Local path for profiling outputs
        "LocalPath": "/opt/ml/output/profiler/"
    }
}
```

Target Time Duration

```
"ProfilerConfig": {
    // Optional. Path to an S3 bucket to save profiling outputs
    "S3OutputPath": "s3://<default-bucket>/<training-job-name>/profiler-output",
    // Available values for ProfilingIntervalInMilliseconds: 100, 200, 500, 1000 (1
 second), 5000 (5 seconds), and 60000 (1 minute) milliseconds.
    "ProfilingIntervalInMilliseconds": 500,
    "ProfilingParameters": {
        "DataloaderProfilingConfig": "{ \"StartTimeInSecSinceEpoch\": 12345567789,
 \"DurationInSeconds\": 10, \"MetricsRegex\": \".*\" }",
        "DetailedProfilingConfig": "{ \"StartTimeInSecSinceEpoch\": 12345567789,
 \"DurationInSeconds\": 10 }",
        // For PythonProfilingConfig,
        // available ProfilerName options: cProfile, Pyinstrument
        // available cProfileTimer options only when using cProfile: cpu, off_cpu,
 total_time
        "PythonProfilingConfig": "{ \"StartTimeInSecSinceEpoch\": 12345567789,
 \"DurationInSeconds\": 10, \"ProfilerName\": \"cProfile\", \"cProfileTimer\":
 \"total_time\" }",
        // Optional. Local path for profiling outputs
```

```
            "LocalPath": "/opt/ml/output/profiler/"
        }
    }
```

启用调试器规则来分析指标

以下示例代码说明了如何配置ProfilerReport规则。

```
"ProfilerRuleConfigurations": [
    {
        "RuleConfigurationName": "ProfilerReport",
        "RuleEvaluatorImage": "895741380848.dkr.ecr.us-west-2.amazonaws.com/sagemaker-
debugger-rules:latest",
        "RuleParameters": {
            "rule_to_invoke": "ProfilerReport",
            "CPUBottleneck_cpu_threshold": "90",
            "IOBottleneck_threshold": "90"
        }
    }
]
```

要查找用于使用调试器规则的 Docker 映像的完整列表，请参阅将调试程序 Docker 映像用于内置规则或自定义规则 (p. 1933). 要查找的键值对RuleParameters，请参阅调试程序内置规则列表 (p. 1826).

使用更新调试器性能分析配置UpdateTrainingJobAPI 操作

在训练作业运行期间，可以使用更新培训工作API 操作。配置新的ProfilerConfig和ProfilerRules 配置对象，然后将训练作业名称指定为TrainingJobName参数。

```
{
    "ProfilerConfig": {
        "DisableProfiler": boolean,
        "ProfilingIntervalInMilliseconds": number,
        "ProfilingParameters": {
            "string" : "string"
        }
    },
    "ProfilerRuleConfigurations": [
        {
            "RuleConfigurationName": "string",
            "RuleEvaluatorImage": "string",
            "RuleParameters": {
                "string" : "string"
            }
        }
    ],
    "TrainingJobName": "your-training-job-name-YYYY-MM-DD-HH-MM-SS-SSS"
}
```

将调试器自定义规则配置添加到 CreateTrainingJob API 操作

可以使用为训练作业配置自定义规则debugHookconfig和调试规则配置中的对象CreateTrainingJobAPI 操作。以下代码示例演示了如何配置自定义ImproperActivation用smdebug使用此 SageMaker API 操作的库。此示例假定您已在中编写自定义规则custom_rules.py文件并将其上传到 Amazon S3 存储桶。该示例提供了预构建的 Docker 映像，您可以使用这些映像运行自定义规则。Amazon SageMaker 调试程序的自定义规则评估程序的注册表 URL (p. 1934) 中列出了这些映像。您可以在 RuleEvaluatorImage 参数中为预构建的 Docker 映像指定 URL 注册表地址。

```
"DebugHookConfig": {
    "S3OutputPath": "s3://<default-bucket>/<training-job-name>/debug-output",
```

```
    "CollectionConfigurations": [
        {
            "CollectionName": "relu_activations",
            "CollectionParameters": {
                "include_regex": "relu",
                "save_interval": "500",
                "end_step": "5000"
            }
        }
    ]
},
"DebugRulesConfigurations": [
    {
        "RuleConfigurationName": "improper_activation_job",
        "RuleEvaluatorImage": "552407032007.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-
debugger-rule-evaluator:latest",
        "InstanceType": "ml.c4.xlarge",
        "VolumeSizeInGB": 400,
        "RuleParameters": {
           "source_s3_uri": "s3://bucket/custom_rules.py",
           "rule_to_invoke": "ImproperActivation",
           "collection_names": "relu_activations"
        }
    }
]
```

要查找用于使用调试器规则的 Docker 映像的完整列表，请参阅将调试程序 Docker 映像用于内置规则或自定义规则 (p. 1933). 要查找的键值对`RuleParameters`，请参阅调试程序内置规则列表 (p. 1826).

AmazonBoto3

可以使用为训练作业配置 Amazon SageMaker 调试器内置规则`create_training_job()`的函数 AmazonBoto3 SageMaker 客户端。你需要在`RuleEvaluatorImage`参数，以下示例向您介绍如何为`create_training_job()`function.

下面的代码显示了如何为`create_training_job()`申请身体并开始培训工作`us-west-2`，假设训练脚本`entry_point/train.py`是使用 TensorFlow 准备的。要查找端到端示例笔记本，请参阅使用 Amazon SageMaker 调试器 (Boto3) 分析 TensorFlow 多 GPU 多节点训练 Job.

> **Note**
>
> 确保使用正确的 Docker 容器镜像。要查找可用Amazon深度学习容器映像，请参阅可用的 Deep Learning Containers 映像. 要查找用于使用调试器规则的 Docker 映像的完整列表，请参阅将调试程序 Docker 映像用于内置规则或自定义规则 (p. 1933).

```
import sagemaker, boto3
import datetime, tarfile

# Start setting up a SageMaker session and a Boto3 SageMaker client
session = sagemaker.Session()
region = session.boto_region_name
bucket = session.default_bucket()

# Upload a training script to a default Amazon S3 bucket of the current SageMaker session
source = 'source.tar.gz'
project = 'debugger-boto3-test'

tar = tarfile.open(source, 'w:gz')
tar.add ('entry_point/train.py') # Specify the directory and name of your training script
tar.close()

s3 = boto3.client('s3')
s3.upload_file(source, bucket, project+'/'+source)
```

```
# Set up a Boto3 session client for SageMaker
sm = boto3.Session(region_name=region).client("sagemaker")

# Start a training job
sm.create_training_job(
    TrainingJobName='debugger-boto3-'+datetime.datetime.now().strftime('%Y-%m-%d-%H-%M-
%S'),
    HyperParameters={
        'sagemaker_submit_directory': 's3://'+bucket+'/'+project+'/'+source,
        'sagemaker_program': '/entry_point/train.py' # training scrip file location and
 name under the sagemaker_submit_directory
    },
    AlgorithmSpecification={
        # Specify a training Docker container image URI (Deep Learning Container or your
 own training container) to TrainingImage.
        'TrainingImage': '763104351884.dkr.ecr.us-west-2.amazonaws.com/tensorflow-
training:2.4.1-gpu-py37-cu110-ubuntu18.04',
        'TrainingInputMode': 'File',
        'EnableSageMakerMetricsTimeSeries': False
    },
    RoleArn='arn:aws:iam::111122223333:role/service-role/AmazonSageMaker-
ExecutionRole-20201014T161125',
    OutputDataConfig={'S3OutputPath': 's3://'+bucket+'/'+project+'/output'},
    ResourceConfig={
        'InstanceType': 'ml.p3.8xlarge',
        'InstanceCount': 1,
        'VolumeSizeInGB': 30
    },
    StoppingCondition={
        'MaxRuntimeInSeconds': 86400
    },
    DebugHookConfig={
        'S3OutputPath': 's3://'+bucket+'/'+project+'/debug-output',
        'CollectionConfigurations': [
            {
                'CollectionName': 'losses',
                'CollectionParameters' : {
                    'train.save_interval': '500',
                    'eval.save_interval': '50'
                }
            }
        ]
    },
    DebugRuleConfigurations=[
        {
            'RuleConfigurationName': 'LossNotDecreasing',
            'RuleEvaluatorImage': '895741380848.dkr.ecr.us-west-2.amazonaws.com/sagemaker-
debugger-rules:latest',
            'RuleParameters': {'rule_to_invoke': 'LossNotDecreasing'}
        }
    ],
    ProfilerConfig={
        'S3OutputPath': 's3://'+bucket+'/'+project+'/profiler-output',
        'ProfilingIntervalInMilliseconds': 500,
        'ProfilingParameters': {
            'DataloaderProfilingConfig': '{"StartStep": 5, "NumSteps": 3, "MetricsRegex":
 ".*", }',
            'DetailedProfilingConfig': '{"StartStep": 5, "NumSteps": 3, }',
            'PythonProfilingConfig': '{"StartStep": 5, "NumSteps": 3, "ProfilerName":
 "cprofile", "cProfileTimer": "total_time"}',
            'LocalPath': '/opt/ml/output/profiler/' # Optional. Local path for profiling
 outputs
        }
    },
    ProfilerRuleConfigurations=[
```

```
        {
            'RuleConfigurationName': 'ProfilerReport',
            'RuleEvaluatorImage': '895741380848.dkr.ecr.us-west-2.amazonaws.com/sagemaker-
debugger-rules:latest',
            'RuleParameters': {'rule_to_invoke': 'ProfilerReport'}
        }
    ]
)
```

配置调试器规则以调试模型参数

以下代码示例演示了如何配置内置VanishingGradient使用此 SageMaker API 的规则。

启用调试器收集输出张量

按如下方式指定调试器挂接配置：

```
DebugHookConfig={
    'S3OutputPath': 's3://<default-bucket>/<training-job-name>/debug-output',
    'CollectionConfigurations': [
        {
            'CollectionName': 'gradients',
            'CollectionParameters' : {
                'train.save_interval': '500',
                'eval.save_interval': '50'
            }
        }
    ]
}
```

这将使训练工作保存张量集合，gradients，每save_interval500 个步骤。要查找可用CollectionName值，请参阅调试器内置集合中的smDebug 客户端库文档. 要查找可用CollectionParameters参数键和值，请参阅sagemaker.debugger.CollectionConfig中的类SageMaker Python 开发工具包文档.

启用调试器规则来调试输出张量

以下DebugRuleConfigurationsAPI 示例演示如何运行内置VanishingGradient对已保存的规则gradients收集。

```
DebugRuleConfigurations=[
    {
        'RuleConfigurationName': 'VanishingGradient',
        'RuleEvaluatorImage': '895741380848.dkr.ecr.us-west-2.amazonaws.com/sagemaker-
debugger-rules:latest',
        'RuleParameters': {
            'rule_to_invoke': 'VanishingGradient',
            'threshold': '20.0'
        }
    }
]
```

通过使用类似于该示例中的配置，调试器使用VanishingGradient关于收集的规则gradients张量。要查找用于使用调试器规则的 Docker 映像的完整列表，请参阅将调试程序 Docker 映像用于内置规则或自定义规则 (p. 1933). 要查找的键值对RuleParameters，请参阅调试程序内置规则列表 (p. 1826).

为分析系统和框架指标配置调试器内置规则

以下示例代码显示如何指定 ProfilerConfig API 操作以启用收集系统和框架指标。

启用调试器分析以收集系统和框架指标

Target Step

```
ProfilerConfig={
    'S3OutputPath': 's3://<default-bucket>/<training-job-name>/profiler-output', #
 Optional. Path to an S3 bucket to save profiling outputs
    # Available values for ProfilingIntervalInMilliseconds: 100, 200, 500, 1000 (1
 second), 5000 (5 seconds), and 60000 (1 minute) milliseconds.
    'ProfilingIntervalInMilliseconds': 500,
    'ProfilingParameters': {
        'DataloaderProfilingConfig': '{
            "StartStep": 5,
            "NumSteps": 3,
            "MetricsRegex": ".*"
        }',
        'DetailedProfilingConfig': '{
            "StartStep": 5,
            "NumSteps": 3
        }',
        'PythonProfilingConfig': '{
            "StartStep": 5,
            "NumSteps": 3,
            "ProfilerName": "cprofile",  # Available options: cprofile, pyinstrument
            "cProfileTimer": "total_time"  # Include only when using cprofile.
 Available options: cpu, off_cpu, total_time
        }',
        'LocalPath': '/opt/ml/output/profiler/' # Optional. Local path for profiling
 outputs
    }
}
```

Target Time Duration

```
ProfilerConfig={
    'S3OutputPath': 's3://<default-bucket>/<training-job-name>/profiler-output', #
 Optional. Path to an S3 bucket to save profiling outputs
    # Available values for ProfilingIntervalInMilliseconds: 100, 200, 500, 1000 (1
 second), 5000 (5 seconds), and 60000 (1 minute) milliseconds.
    'ProfilingIntervalInMilliseconds': 500,
    'ProfilingParameters': {
        'DataloaderProfilingConfig': '{
            "StartTimeInSecSinceEpoch": 12345567789,
            "DurationInSeconds": 10,
            "MetricsRegex": ".*"
        }',
        'DetailedProfilingConfig': '{
            "StartTimeInSecSinceEpoch": 12345567789,
            "DurationInSeconds": 10
        }',
        'PythonProfilingConfig': '{
            "StartTimeInSecSinceEpoch": 12345567789,
            "DurationInSeconds": 10,
            "ProfilerName": "cprofile",  # Available options: cprofile, pyinstrument
            "cProfileTimer": "total_time"  # Include only when using cprofile.
 Available options: cpu, off_cpu, total_time
        }',
        'LocalPath': '/opt/ml/output/profiler/' # Optional. Local path for profiling
 outputs
    }
}
```

启用调试器规则来分析指标

以下示例代码说明了如何配置ProfilerReport规则。

```
ProfilerRuleConfigurations=[
    {
        'RuleConfigurationName': 'ProfilerReport',
        'RuleEvaluatorImage': '895741380848.dkr.ecr.us-west-2.amazonaws.com/sagemaker-debugger-rules:latest',
        'RuleParameters': {
            'rule_to_invoke': 'ProfilerReport',
            'CPUBottleneck_cpu_threshold': '90',
            'IOBottleneck_threshold': '90'
        }
    }
]
```

要查找用于使用调试器规则的 Docker 映像的完整列表，请参阅将调试程序 Docker 映像用于内置规则或自定义规则 (p. 1933). 要查找的键值对`RuleParameters`，请参阅调试程序内置规则列表 (p. 1826).

使用更新调试器性能分析配置`UpdateTrainingJob`API 操作

在训练作业运行期间，可以使用`update_training_job()`的函数AmazonBoto3 SageMaker 客户端。配置新的ProfilerConfig和ProfilerRules 配置对象，然后将训练作业名称指定为`TrainingJobName`参数。

```
ProfilerConfig={
    'DisableProfiler': boolean,
    'ProfilingIntervalInMilliseconds': number,
    'ProfilingParameters': {
        'string' : 'string'
    }
},
ProfilerRuleConfigurations=[
    {
        'RuleConfigurationName': 'string',
        'RuleEvaluatorImage': 'string',
        'RuleParameters': {
            'string' : 'string'
        }
    }
],
TrainingJobName='your-training-job-name-YYYY-MM-DD-HH-MM-SS-SSS'
```

将调试器自定义规则配置添加到 CreateTrainingJob API 操作

可以使用为训练作业配置自定义规则debugHookconfig和调试规则配置使用的对象AmazonBoto3 SageMaker 客户端的`create_training_job()`function. 以下代码示例演示了如何配置自定义`ImproperActivation`用smdebug使用此 SageMaker API 操作的库。此示例假定您已在中编写自定义规则custom_rules.py文件并将其上传到 Amazon S3 存储桶。该示例提供了预构建的 Docker 映像，您可以使用这些映像运行自定义规则。Amazon SageMaker 调试程序的自定义规则评估程序的注册表 URL (p. 1934) 中列出了这些映像。您可以在 `RuleEvaluatorImage` 参数中为预构建的 Docker 映像指定 URL 注册表地址。

```
DebugHookConfig={
    'S3OutputPath': 's3://<default-bucket>/<training-job-name>/debug-output',
    'CollectionConfigurations': [
        {
            'CollectionName': 'relu_activations',
            'CollectionParameters': {
                'include_regex': 'relu',
                'save_interval': '500',
                'end_step': '5000'
            }
        }
    ]
```

```
},
DebugRulesConfigurations=[
    {
        'RuleConfigurationName': 'improper_activation_job',
        'RuleEvaluatorImage': '552407032007.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-
debugger-rule-evaluator:latest',
        'InstanceType': 'ml.c4.xlarge',
        'VolumeSizeInGB': 400,
        'RuleParameters': {
           'source_s3_uri': 's3://bucket/custom_rules.py',
           'rule_to_invoke': 'ImproperActivation',
           'collection_names': 'relu_activations'
        }
    }
]
```

要查找用于使用调试器规则的 Docker 映像的完整列表，请参阅将调试程序 Docker 映像用于内置规则或自定义规则 (p. 1933). 要查找的键值对`RuleParameters`，请参阅调试程序内置规则列表 (p. 1826).

调试程序内置规则列表

使用 Amazon SageMaker 调试器提供的调试器内置规则，并分析训练模型时发出的张量。调试器内置规则监控对训练作业至关重要的各种常见情况。您可以使用内置规则调用Amazon SageMaker Python 开发工具包或低级别 SageMaker API 操作。根据您选择的深度学习框架，内置规则有四个有效性范围，如下表所示。

Note

培训作业的内置规则的最大数量为 20`ProfilerRule`20 用于`Rule`. SageMaker 调试器完全管理内置规则并行分析你的训练作业。有关计费的更多信息，请参阅Amazon SageMaker Studio 不收取额外费用的 部分Amazon SageMaker 定价页.

Important

要使用新的调试器功能，您需要升级 SageMaker Python SDK 和 smDebug 客户端库。在你的 iPython 内核、Jupyter 笔记本或 JupyterLab 环境中，运行以下代码以安装最新版本的库并重新启动内核。

```
import sys
import IPython
!{sys.executable} -m pip install -U sagemaker smdebug
IPython.Application.instance().kernel.do_shutdown(True)
```

调试程序 ProfilerRule

以下规则是调试器内置规则，可以使用`ProfilerRule.sagemaker`类方法。

用于生成分析报告的调试器内置规则

有效性范围	内置规则
任何 SageMaker 培训作业的分析报告	• `ProfilerReport (p. 1828)`

调试器用于监视硬件系统资源利用率的内置规则（系统指标）

有效性范围	内置规则
任何 SageMaker 培训作业的通用系统监控规则	• `BatchSize (p. 1829)` • `CPUBottleneck (p. 1831)`

有效性范围	内置规则
	• GPUMemoryIncrease (p. 1832) • IOBottleneck (p. 1833) • LoadBalancing (p. 1834) • LowGPUUtilization (p. 1834) • OverallSystemUsage (p. 1835)

用于分析框架指标的调试器内置规则

有效性范围	内置规则
深度学习框架的分析规则（TensorFlow 和 PyTorch）	• MaxInitializationTime (p. 1836) • OverallFrameworkMetrics (p. 1836) • StepOutlier (p. 1837)

调试程序规则

以下规则是调试器内置规则，可以使用`Rule.sagemaker`类方法。

用于生成训练报告的调试器内置规则

有效性范围	内置规则
SageMaker xgBoost 培训工作的培训报告	• create_xgboost_report (p. 1838)

调试器用于调试模型训练数据（输出张量）的内置规则

有效性范围	内置规则
深度学习框架（TensorFlow、MXNet 和 PyTorch）	• dead_relu (p. 1838) • exploding_tensor (p. 1839) • poor_weight_initialization (p. 1841) • saturated_activation (p. 1842) • vanishing_gradient (p. 1844) • weight_update_ratio (p. 1845)
深度学习框架（TensorFlow、MXNet 和 PyTorch）和 XGBoost 算法	• all_zero (p. 1847) • class_imbalance (p. 1848) • loss_not_decreasing (p. 1850) • overfit (p. 1852) • overtraining (p. 1853) • similar_across_runs (p. 1854) • stalled_training_rule (p. 1855) • tensor_variance (p. 1857) • unchanged_tensor (p. 1858)
深度学习应用程序	• check_input_images (p. 1860) • nlp_sequence_ratio (p. 1861)
XGBoost 算法	• confusion (p. 1862)

有效性范围	内置规则
	• feature_importance_overweight (p. 1864) • tree_depth (p. 1865)

使用具有默认参数值的内置规则— 采用以下配置格式：

```
from sagemaker.debugger import Rule, ProfilerRule, rule_configs

rules = [
    ProfilerRule.sagemaker(rule_configs.BuiltInRuleName_1()),
    ProfilerRule.sagemaker(rule_configs.BuiltInRuleName_2()),
    ...
    ProfilerRule.sagemaker(rule_configs.BuiltInRuleName_n()),
    Rule.sagemaker(rule_configs.built_in_rule_name_1()),
    Rule.sagemaker(rule_configs.built_in_rule_name_2()),
    ...
    Rule.sagemaker(rule_configs.built_in_rule_name_n())
]
```

使用内置规则来自定义参数值— 采用以下配置格式：

```
from sagemaker.debugger import Rule, ProfilerRule, rule_configs

rules = [
    ProfilerRule.sagemaker(
        base_config=rule_configs.BuiltInRuleName(),
        rule_parameters={
                "key": "value"
        }
    )
    Rule.sagemaker(
        base_config=rule_configs.built_in_rule_name(),
        rule_parameters={
                "key": "value"
        }
        collections_to_save=[
            CollectionConfig(
                name="tensor_collection_name",
                parameters={
                    "key": "value"
                }
            )
        ]
    )
]
```

查找可用的密钥`rule_parameters`参数，请参阅参数描述表。

参数描述表下方为每个内置规则提供了示例规则配置代码。

- 有关使用调试器内置规则的完整说明和示例，请参阅调试程序内置规则示例代码 (p. 1813).
- 有关在低级 SageMaker API 操作中使用内置规则的完整说明，请参阅使用 Amazon SageMaker API 配置调试器 (p. 1816).

剖析器报告

ProfilerReport 规则调用所有内置规则来执行监控和分析。它会创建一个性能分析报告，并在触发单个规则时进行更新。您可以在训练作业运行期间或培训作业完成后下载全面的分析报告。您可以调整规则参数值以自

定义内置监视和分析规则的敏感度。以下示例代码显示了通过 ProfilerREPORT 规则调整内置规则参数的基本格式。

```
rules=[
    ProfilerRule.sagemaker(
        rule_configs.ProfilerReport(
            <BuiltInRuleName>_<parameter_name> = value
        )
    )
]
```

如果您在没有任何自定义参数的情况下触发此 ProfilerPORT 规则（如以下示例代码所示），则 ProfilerPORT 规则将触发所有内置规则，以便使用默认参数值进行监视和分析。

```
rules=[ProfilerRule.sagemaker(rule_configs.ProfilerReport())]
```

以下示例代码说明了如何指定和调整 CPUBalyeck 规则。`cpu_threshold`参数和 IO瓶颈规则`threshold`参数。

```
rules=[
    ProfilerRule.sagemaker(
        rule_configs.ProfilerReport(
            CPUBottleneck_cpu_threshold = 90,
            IOBottleneck_threshold = 90
        )
    )
]
```

整体系统使用规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`<BuiltInRuleName>_<parameter_name>`	可自定义参数来调整其他内置监控和分析规则的阈值。 可选 默认值：`None`

BatchSize

BatchSize 规则有助于检测 GPU 是否由于批量大小而未充分利用。要检测此问题，此规则将监视平均 CPU 利用率、GPU 利用率和 GPU 内存利用率。如果 CPU、GPU 和 GPU 内存的使用率平均较低，则可能表明训练作业可以在较小的实例类型上运行，也可以使用更大的批量运行。这种分析不适用于大量分配内存过度的框架。但是，增加批量大小可能会导致处理或数据加载瓶颈，因为每次迭代都需要更长的数据预处理时间。

BatchSize 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
cpu_threshold_p95	定义 CPU 利用率 95 分位数的阈值（以百分比为单位）。 可选 有效值：整数 默认值：70（百分比）
gpu_threshold_p95	定义 GPU 利用率第 95 分位数的阈值（以百分比为单位）。 可选 有效值：整数 默认值：70（百分比）
gpu_memory_threshold_p95	定义 GPU 内存使用率第 95 分位数的阈值（以百分比为单位）。 可选 有效值：整数 默认值：70（百分比）
patience	定义规则开始评估之前要跳过的数据点数。培训作业的前几个步骤通常显示大量的数据流程，因此请保持耐心，并防止使用您使用此参数指定的给定数量的分析数据过早调用规则。 可选 有效值：整数 默认值：100
window	用于计算分位数的窗口大小。 可选 有效值：整数 默认值：500
scan_interval_us	扫描时间轴文件的时间间隔。 可选 有效值：整数

参数名称	描述
	默认值：60000000（以微秒为单位）

CPU瓶颈

CPU瓶颈规则有助于检测 GPU 是否由于 CPU 瓶颈而未充分利用。如果 CPU 瓶颈数超过预定义的阈值，则规则返回 True。

CPUReck 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
threshold	定义瓶颈时间与总训练时间的比例的阈值。如果比例超过为阈值参数指定的百分比，则规则会将规则状态切换为 True。 可选 有效值：整数 默认值：50（百分比）
gpu_threshold	定义 GPU 利用率低的阈值。 可选 有效值：整数 默认值：10（百分比）
cpu_threshold	定义 CPU 利用率高的阈值。 可选 有效值：整数 默认值：90（百分比）
patience	定义规则开始评估之前要跳过的数据点数。培训作业的前几个步骤通常显示大量的数据流程，因此请保持耐心，并防止使用您使用此参数指定的给定数量的分析数据过早调用规则。 可选 有效值：整数 默认值：100
scan_interval_us	扫描时间轴文件的时间间隔。 可选

参数名称	描述
	有效值：整数 默认值：60000000（以微秒为单位）

gpumMory 增加

gpumemoryReader 规则有助于检测 GPU 上的内存使用量大幅增加。

Gpumemory 增加规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
increase	定义绝对内存增加的阈值。 可选 有效值：整数 默认值：10（百分比）
patience	定义规则开始评估之前要跳过的数据点数。培训作业的前几个步骤通常显示大量的数据流程，因此请保持耐心，并防止使用您使用此参数指定的给定数量的分析数据过早调用规则。 可选 有效值：整数 默认值：100
window	用于计算分位数的窗口大小。 可选 有效值：整数 默认值：500
scan_interval_us	扫描时间轴文件的时间间隔。 可选 有效值：整数 默认值：60000000（以微秒为单位）

IO瓶颈

此规则有助于检测 GPU 是否由于数据 IO 瓶颈而未充分利用。如果 IO 瓶颈数超过预定义的阈值，则规则返回 True。

IOBakeck 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`threshold`	定义规则返回 True 时的阈值。 可选 有效值：整数 默认值：`50`（百分比）
`gpu_threshold`	定义何时认为 GPU 被认为未充分利用的阈值。 可选 有效值：整数 默认值：`70`（百分比）
`io_threshold`	定义高 IO 等待时间的阈值。 可选 有效值：整数 默认值：`50`（百分比）
`patience`	定义规则开始评估之前要跳过的数据点数。培训作业的前几个步骤通常显示大量的数据流程，因此请保持耐心，并防止使用您使用此参数指定的给定数量的分析数据过早调用规则。 可选 有效值：整数 默认值：`1000`
`scan_interval_us`	扫描时间轴文件的时间间隔。 可选 有效值：整数 默认值：`60000000`（以微秒为单位）

负载均衡

LoadBalancing 规则有助于检测多个 GPU 之间的工作负载均衡中的问题。

负载平衡规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`threshold`	定义工作负载百分比。 可选 有效值：整数 默认值：`0.5`（无单位比例）
`patience`	定义规则开始评估之前要跳过的数据点数。培训作业的前几个步骤通常显示大量的数据流程，因此请保持耐心，并防止使用您使用此参数指定的给定数量的分析数据过早调用规则。 可选 有效值：整数 默认值：`10`
`scan_interval_us`	扫描时间轴文件的时间间隔。 可选 有效值：整数 默认值：`60000000`（以微秒为单位）

GPU 利用率低

LowGPU 利用率规则有助于检测 GPU 利用率是低还是受到波动影响。对于每个工作线程上的每个 GPU，都会检查这个 如果第 95 个分位数低于阈hold_p95 表示利用率不足，则规则返回 True。如果第 95 个分位数高于 tarhold_p95 且第 5 个分位数低于 tarhold_p5，则规则返回 true，这表示波动。

低 GPU 利用率规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串

参数名称	描述
threshold_p95	95 分位数的阈值，低于此阈值，GPU 被认为未充分利用。 可选 有效值：整数 默认值：70（百分比）
threshold_p5	第 5 个分位数的阈值。默认值为 10%。 可选 有效值：整数 默认值：10（百分比）
patience	定义规则开始评估之前要跳过的数据点数。培训作业的前几个步骤通常显示大量的数据流程，因此请保持耐心，并防止使用您使用此参数指定的给定数量的分析数据过早调用规则。 可选 有效值：整数 默认值：1000
window	用于计算分位数的窗口大小。 可选 有效值：整数 默认值：500
scan_interval_us	扫描时间轴文件的时间间隔。 可选 有效值：整数 默认值：60000000（以微秒为单位）

整体系统使用情况

OverallSystemUse 规则衡量每个工作节点的整体系统使用率。该规则目前仅聚合每个节点的值并计算其百分位数。

整体系统使用规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填

参数名称	描述
	有效值：字符串
scan_interval_us	扫描时间轴文件的时间间隔。 可选 有效值：整数 默认值：60000000（以微秒为单位）

最大初始化时间

maxInitialize Time 规则有助于检测训练初始化是否花费了太长的时间。规则一直等到第一步可用。

MaxInitialize 时间规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
threshold	定义等待第一步可用的阈值（以分钟为单位）。 可选 有效值：整数 默认值：20（分钟）
scan_interval_us	扫描时间轴文件的时间间隔。 可选 有效值：整数 默认值：60000000（以微秒为单位）

整体框架指标

OverallFramerameMetrics 规则总结了花在框架指标上的时间，例如向前和向后传递以及数据加载。

总体框架指标规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
scan_interval_us	扫描时间轴文件的时间间隔。

参数名称	描述
	可选 有效值：整数 默认值：60000000（以微秒为单位）

stepOtLier

stepOutLier 规则有助于检测步长持续时间内的异常值。此规则返回`True`如果有异常值的步长持续时间大于`stddev`在一个时间范围内整个步骤持续时间的 sigmas。

StepOutlier 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`stddev`	定义将标准差乘以的因子。例如，当步长持续时间大于或小于标准差的 5 倍时，默认情况下会调用该规则。 可选 有效值：整数 默认值：5（分钟）
`mode`	已保存步骤的模式以及应在哪个规则上运行的模式。默认规则将在从 EVAL 和 TRAIN 阶段开始运行的步骤 可选 有效值：整数 默认值：5（分钟）
`n_outliers`	规则返回 True 之前要忽略多少异常值 可选 有效值：整数 默认值：10
`scan_interval_us`	扫描时间轴文件的时间间隔。 可选 有效值：整数 默认值：60000000（以微秒为单位）

CreateexGBoost 报告

CreatexgBoost报告规则收集 xgBoost 训练作业的输出张量，并自动生成全面的训练报告。您可以在训练作业运行期间或培训作业完成后下载全面的概况分析报告，并检查培训进度或培训作业的最终结果。默认情况下，CreatexgBoostReport 规则收集以下输出张量：

- `hyperparameters`— 在第一步保存
- `metrics`— 每 5 个步骤节省损失和准确性
- `feature_importance`— 每 5 个步骤保存
- `predictions`— 每 5 个步骤保存
- `labels`— 每 5 个步骤保存

CreatexGBoost 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串

```
rules=[
    Rule.sagemaker(
        rule_configs.create_xgboost_report()
    )
]
```

DeadRelu

此规则检测试验中的修正线性单元 (ReLU) 激活函数的百分比是否因它们的激活活动已降至阈值以下而被视为死亡。如果层中非活跃 ReLU 的百分比大于非活跃 ReLU 的 `threshold_layer` 值，则该规则会返回 `True`。

DeadRelu 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`tensor_regex`	正则表达式模式列表，用于将该比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串列表或逗号分隔的字符串

参数名称	描述
	默认值：".*relu_output"
threshold_inactivity	定义一个活动级别，低于该级别的 ReLU 将被视为死亡。ReLU 可能在试验开始时处于活动状态，然后在训练过程中慢慢死亡。如果 ReLU 处于活动状态并低于 threshold_inactivity，则将其视为死亡。 可选 有效值：Float 默认值：1.0（百分比）
threshold_layer	如果层中非活跃 ReLU 百分比大于 threshold_layer，则返回 True。 如果层中非活跃 ReLU 百分比小于 threshold_layer，则返回 False。 可选 有效值：Float 默认值：50.0（百分比）

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.dead_relu(),
        rule_parameters={
                "tensor_regex": ".*relu_output|.*ReLU_output",
                "threshold_inactivity": "1.0",
                "threshold_layer": "50.0"
        }
        collections_to_save=[
            CollectionConfig(
                name="custom_relu_collection",
                parameters={
                    "include_regex: ".*relu_output|.*ReLU_output",
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

Note

此规则不适用于 XGBoost 算法。

ExplodingTensor

此规则检测训练期间发射的张量是否具有非有限值，即无限值或 NaN（不是数字）。如果检测到非有限值，则该规则返回 True。

ExplodingTensor 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
collection_names	该规则检查其张量的集合名称列表。 可选 有效值：字符串 默认值：None
tensor_regex	正则表达式模式列表，用于将该比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串 默认值：None
only_nan	如果为 True，则仅监视 base_trial 张量是否为 NaN 值，而不监视无穷大。 如果为 False，则将 NaN 和无穷大都视为爆炸值并监视两者。 可选 默认值：False

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.exploding_tensor(),
        rule_parameters={
                "tensor_regex": ".*gradient",
                "only_nan": "False"
        }
        collections_to_save=[
            CollectionConfig(
                name="gradients",
                parameters={
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

Note

此规则不适用于 XGBoost 算法。

PoorWeightInitialization

此规则检测模型参数是否初始化不当。

良好的初始化会破坏神经网络中权重和梯度的对称性，并在各层之间保持相应的激活方差。否则，神经网络无法有效地学习。诸如 Xavier 之类的初始化程序旨在使激活之间的方差保持恒定，这对于训练非常深的神经网络尤其重要。初始化太小可能会导致梯度消失。初始化过大可能会导致梯度爆炸。此规则检查各层激活输入的方差、梯度分布以及初始步骤的损失收敛性，确定神经网络是否初始化不当。

PoorWeightInitialization 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`activation_inputs_regex`	正则表达式模式列表，用于将该比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串 默认值：`".*relu_input"`
`threshold`	如果每个层的权重的最小方差和最大方差之间的比率超过某个步骤中的 `threshold`，则该规则返回 `True`。 可选 有效值：Float 默认值：`10.0`
`distribution_range`	如果梯度分布的第 5 百分位数和第 95 百分位数之间的最小差值小于 `distribution_range`，则该规则返回 `True`。 可选 有效值：Float 默认值：`0.001`
`patience`	等到损失被认为不再减少的步骤数。 可选 有效值：整数

参数名称	描述
	默认值：5
steps	此规则分析的步骤数。您通常只需要检查前几次迭代。 可选 有效值：Float 默认值：10

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.poor_weight_initialization(),
        rule_parameters={
                "activation_inputs_regex": ".*relu_input|.*ReLU_input",
                "threshold": "10.0",
                "distribution_range": "0.001",
                "patience": "5",
                "steps": "10"
        },
        collections_to_save=[
            CollectionConfig(
                name="custom_relu_collection",
                parameters={
                    "include_regex": ".*relu_input|.*ReLU_input",
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

Note

此规则不适用于 XGBoost 算法。

SaturatedActivation

此规则检测 tanh 和 sigmoid 激活层是否正在变得饱和。当层的输入接近激活函数的最大或最小值时，激活层将饱和。tanh 和 sigmoid 激活函数的最小和最大值由它们各自的 min_threshold 和 max_thresholds 值定义。如果节点的活动降到 threshold_inactivity 百分比以下，则将其视为饱和。如果超过 threshold_layer 百分比的节点处于饱和状态，则该规则返回 True。

SaturatedActivation 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
collection_names	该规则检查其张量的集合名称列表。

参数名称	描述
	可选 有效值：字符串列表或逗号分隔的字符串 默认值：无
tensor_regex	正则表达式模式列表，用于将该比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串 默认值：".*tanh_input|.*sigmoid_input".
threshold_tanh_min	定义 tanh 激活函数输入极值的最小和最大阈值，定义为：。(min_threshold, max_threshold). 默认值是根据梯度消失阈值 0.0000001 确定的。 可选 有效值：Float 默认值：-9.4999
threshold_tanh_max	定义 tanh 激活函数输入极值的最小和最大阈值，定义为：。(min_threshold, max_threshold). 默认值是根据梯度消失阈值 0.0000001 确定的。 可选 有效值：Float 默认值：9.4999
threshold_sigmoid_min	定义 sigmoid 激活函数输入极值的最小和最大阈值，定义为：。(min_threshold, max_threshold). 默认值是根据梯度消失阈值 0.0000001 确定的。 可选 有效值：Float 默认值：-23
threshold_sigmoid_max	定义 sigmoid 激活函数输入极值的最小和最大阈值，定义为：。(min_threshold, max_threshold). 默认值是根据梯度消失阈值 0.0000001 确定的。 可选 有效值：Float 默认值：16.99999

参数名称	描述
threshold_inactivity	不活动百分比，低于该百分比的激活层被视为饱和。激活可能在试验开始时处于活动状态，然后在训练过程中逐渐变得不那么活跃。 可选 有效值：Float 默认值：1.0
threshold_layer	如果层中的饱和激活数大于 threshold_layer 百分比，则返回 True。 如果层中的饱和激活数小于 threshold_layer 百分比，则返回 False。 可选 有效值：Float 默认值：50.0

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.saturated_activation(),
        rule_parameters={
                "tensor_regex": ".*tanh_input|.*sigmoid_input",
                "threshold_tanh_min": "-9.4999",
                "threshold_tanh_max": "9.4999",
                "threshold_sigmoid_min": "-23",
                "threshold_sigmoid_max": "16.99999",
                "threshold_inactivity": "1.0",
                "threshold_layer": "50.0"
        },
        collections_to_save=[
            CollectionConfig(
                name="custom_activations_collection",
                parameters={
                    "include_regex": ".*tanh_input|.*sigmoid_input"
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

Note

此规则不适用于 XGBoost 算法。

VanishingGradient

该规则检测试验中的梯度是否变得非常小或降至零幅度。如果梯度绝对值的平均值低于指定的 threshold，则该规则返回 True。

VanishingGradient 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
threshold	确定梯度消失的值。 可选 有效值：Float 默认值：0.0000001。

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.vanishing_gradient(),
        rule_parameters={
                "threshold": "0.0000001"
        },
        collections_to_save=[
            CollectionConfig(
                name="gradients",
                parameters={
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

Note

此规则不适用于 XGBoost 算法。

WeightUpdateRatio

此规则跟踪训练期间更新与权重的比率，并检测该比率是过大还是过小。如果更新与权重的比率大于 large_threshold value 或者此比率小于 small_threshold，则该规则返回 True。

当更新与梯度相当时，训练条件是最佳的。过大的更新会使权重远离最佳值，而过小的更新会导致收敛速度非常慢。此规则要求权重可用于两个训练步骤，train.save_interval需要设置等于num_steps.

WeightUpdateRatio 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串

参数名称	描述
num_steps	该规则检查以确定张量是否已更改的步骤数。 要比较权重比率的步骤数。如果未传递任何值，则默认情况下，将针对当前步骤和紧接的上一个保存步骤运行该规则。如果您通过传递此参数的值来覆盖默认值，则会在步骤之间进行比较。s然后一步>= s - num_steps. 可选 有效值：整数 默认值：None
large_threshold	在规则返回之前，要执行加权的更新比率可以取的最大值。True. 可选 有效值：Float 默认值：10.0
small_threshold	要执行加权的更新比率可以取的最小值，低于该值时规则将返回。True. 可选 有效值：Float 默认值：0.00000001
epsilon	一个小常量，用于确保在计算要执行加权的更新时，调试器不除以零。 可选 有效值：Float 默认值：0.000000001

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.weight_update_ratio(),
        rule_parameters={
                "num_steps": "100",
                "large_threshold": "10.0",
                "small_threshold": "0.00000001",
                "epsilon": "0.000000001"
        },
        collections_to_save=[
            CollectionConfig(
                name="weights",
                parameters={
                    "train.save_interval": "100"
                }
            )
        ]
```

```
    )
]
```

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

Note

此规则不适用于 XGBoost 算法。

AllZero

此规则检测张量值的全部或值的指定百分比是否为零。

此规则可应用于受支持的深度学习框架（TensorFlow、MXNet 和 PyTorch）之一，也可以应用于 XGBoost 算法。必须指定 `collection_names` 或 `tensor_regex` 参数。如果指定了这两个参数，则规则将检查两个集合内张量的并集。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

AllZero 的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`collection_names`	该规则检查其张量的集合名称列表。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：`None`
`tensor_regex`	正则表达式模式列表，用于将该比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：`None`
`threshold`	指定张量中必须为零才能调用此规则的值的百分比。 可选 有效值：Float 默认值：100（以百分比为单位）

```
built_in_rules = [
```

```
    Rule.sagemaker(
        base_config=rule_configs.all_zero(),
        rule_parameters={
                "tensor_regex": ".*",
                "threshold": "100"
        },
        collections_to_save=[
            CollectionConfig(
                name="all",
                parameters={
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

ClassImbalance

此规则用于衡量分类之间的采样不平衡，如果不平衡状况超过阈值，或者由于不平衡而导致代表性不足的分类发生太多错误预测，则会引发错误。

分类模型需要训练数据集中的分类保持良好平衡，或者在训练期间对分类执行适当的加权/取样。此规则执行以下检查：

- 它计算每个分类的发生次数。如果最小分类和最大分类之间的样本数比率大于 `threshold_imbalance`，则会引发错误。
- 它检查每个分类的预测精度。如果未正确应用重采样或加权，则模型对于具有许多训练样本的分类来说可以达到较高的精度，但对于训练样本少的分类来说，精度较低。如果某个分类的错误预测比例大于 `threshold_misprediction`，则会引发错误。

此规则可应用于受支持的深度学习框架（TensorFlow、MXNet 和 PyTorch）之一，也可以应用于 XGBoost 算法。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

ClassImbalance 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`threshold_imbalance`	最小分类和最大分类中样本数之间可接受的不平衡。超过此阈值会引发错误。 可选 有效值：Float 默认值：`10`
`threshold_misprediction`	对每个分类允许的错误预测部分的限制。超过此阈值会引发错误。代表性不足的分类最有可能超过此阈值。

参数名称	描述
	可选 有效值：Float 默认值：0.7
samples	在评估不平衡之前必须处理的标签数。在多个步骤中看到足够的样本之前，可能不会触发该规则。数据集包含的分类越多，此 sample 数值应越大。 可选 有效值：整数 默认值：500（假设像 MNIST 这样的数据集有 10 个类）
argmax	如果为 True，则 np.argmax 应用于预测张量。当您对每个分类都有概率向量时，此项是必需的。它用于确定哪个分类具有最高概率。 条件 有效值：Boolean 默认值：False
labels_regex	包含标签的张量的名称。 可选 有效值：字符串 默认值：".*labels"
predictions_regex	包含预测的张量的名称。 可选 有效值：字符串 默认值：".*predictions"

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.class_imbalance(),
        rule_parameters={
                "threshold_imbalance": "10",
                "threshold_misprediction": "0.7",
                "samples": "500",
                "argmax": "False",
                "labels_regex": ".*labels",
                "predictions_regex": ".*predictions"
        },
        collections_to_save=[
            CollectionConfig(
                name="custom_output_collection",
                parameters={
                    "include_regex": ".*labels|.*predictions",
```

```
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

LossNotDecreasing

此规则检测值损失是否以适当的速度减少。这些损失必须是标量。

此规则可应用于受支持的深度学习框架（TensorFlow、MXNet 和 PyTorch）之一，也可以应用于 XGBoost 算法。必须指定 `collection_names` 或 `tensor_regex` 参数。如果指定了这两个参数，则规则将检查两个集合内张量的并集。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

LossNotDecreasing 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`collection_names`	该规则检查其张量的集合名称列表。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：`None`
`tensor_regex`	一个正则表达式模式列表，用于将此比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：`None`
`use_losses_collection`	如果设置为 `True`，则当名为“losses”的集合存在时，查找该集合中的损失。 可选 有效值：Boolean 默认值：`True`
`num_steps`	该规则检查损失是否已减少的最小步骤数。规则评估每 `num_steps` 个步骤发生一次。该规则将此步骤的损失与至少比当前步骤落后 `num_steps` 的步骤的损失进行比较。例如，假设每三个步骤保存

参数名称	描述
	一次损失，但 num_steps 设置为 10。在步骤 21 中，将步骤 21 的损失与步骤 9 的损失进行比较。检查损失的下一个步骤是步骤 33，因为步骤 21 之后的第十个步骤是步骤 31，并且在步骤 31 和步骤 32 中不保存损失。 可选 有效值：整数 默认值：10
diff_percent	在 num_steps 个步骤之间，损失应减少的最小百分比差异。 可选 有效值：0.0 < 浮点值 < 100 默认值：0.1（百分比）
increase_threshold_percent	在损失一直在增加的情况下，允许损失增加的最大阈值百分比 可选 有效值：0 < 浮点值 < 100 默认值：5（百分比）
mode	用于查询张量值以进行规则检查的调试器模式的名称。如果未传递该参数，规则默认按顺序进行检查，先检查 mode.EVAL，然后检查 mode.TRAIN，最后检查 mode.GLOBAL。 可选 有效值：String (EVAL、TRAIN，或者GLOBAL) 默认值：GLOBAL

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.loss_not_decreasing(),
        rule_parameters={
                "tensor_regex": ".*",
                "use_losses_collection": "True",
                "num_steps": "10",
                "diff_percent": "0.1",
                "increase_threshold_percent": "5",
                "mode": "GLOBAL"
        },
        collections_to_save=[
            CollectionConfig(
                name="losses",
                parameters={
                    "save_interval": "500"
                }
```

```
            )
        ]
    )
]
```

Overfit

此规则通过比较验证和训练损失来检测模型是否与训练数据过度拟合。

此规则可应用于受支持的深度学习框架（TensorFlow、MXNet 和 PyTorch）之一，也可以应用于 XGBoost 算法。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

> **Note**
>
> 防止过度拟合的标准方法是对模型进行规范化。

Overfit 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`tensor_regex`	正则表达式模式列表，用于将该比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：无
`start_step`	开始比较验证和训练损失的步骤。 可选 有效值：整数 默认值：`0`
`patience`	在模型被视为过度拟合之前允许 `ratio_threshold` 超过设置的值的步数。 可选 有效值：整数 默认值：`1`
`ratio_threshold`	平均验证损失和平均训练损失之间的差异与平均训练损失的最大比率。如果 `patience` 个步骤超过此阈值，则表示模型过度拟合，并且规则返回 `True`。

参数名称	描述
	可选 有效值：Float 默认值：0.1

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.overfit(),
        rule_parameters={
                "tensor_regex": ".*",
                "start_step": "0",
                "patience": "1",
                "ratio_threshold": "0.1"
        },
        collections_to_save=[
            CollectionConfig(
                name="losses",
                parameters={
                    "train.save_interval": "100",
                    "eval.save_interval": "10"
                }
            )
        ]
    )
]
```

Overtraining

此规则检测模型是否过度训练。在对行为良好的模型进行了多次训练迭代（训练和验证损失减少）之后，该模型接近损失函数的最低限度，并且不再改善。如果模型继续训练，则可能会发生验证损失开始增加，因为模型开始过度拟合。该规则设置阈值和条件，以确定模型是否没有改进，并防止由于过度训练而导致的过度拟合问题。

此规则可应用于受支持的深度学习框架（TensorFlow、MXNet 和 PyTorch）之一，也可以应用于 XGBoost 算法。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

> **Note**
>
> 可以通过提前停止来避免过度训练。有关提前停止的信息，请参阅提前停止训练作业 (p. 1957)。有关演示如何将 Spot 训练用于调试器的示例，请参阅。使用 Amazon SageMaker 调试器启用竞价培训.

Overtraining 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
patience_train	在认为训练损失不再有改进之前等待的步骤数。 可选

参数名称	描述
	有效值：整数 默认值：5
patience_validation	在认为验证损失不再有改进之前等待的步骤数。 可选 有效值：整数 默认值：10
delta	误差被视为新的最佳值之前应改进的误差程度的最小阈值。 可选 有效值：Float 默认值：0.01

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.overtraining(),
        rule_parameters={
                "patience_train": "5",
                "patience_validation": "10",
                "delta": "0.01"
        },
        collections_to_save=[
            CollectionConfig(
                name="losses",
                parameters={
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

SimilarAcrossRuns

此规则将从基本试验中收集的张量与从另一试验中收集的张量进行比较。

此规则可应用于受支持的深度学习框架（TensorFlow、MXNet 和 PyTorch）之一，也可以应用于 XGBoost 算法。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

SimilarAcrossRuns 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填

参数名称	描述
	有效值：字符串
other_trial	要将其张量与从当前收集的那些张量进行比较的已完成训练作业名称。base_trial. 必填 有效值：字符串
collection_names	该规则检查其张量的集合名称列表。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：无
tensor_regex	正则表达式模式列表，用于将该比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：无

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.similar_across_runs(),
        rule_parameters={
                "other_trial": "<specify-another-job-name>",
                "collection_names": "losses",
                "tensor_regex": ".*"
        },
        collections_to_save=[
            CollectionConfig(
                name="losses",
                parameters={
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

已停止的训练规则

stalledTrainingRule 检测训练作业是否没有进展，如果规则触发，则停止训练作业。该规则要求张量在其定义的时间间隔内定期保存张量`threshold`参数。该规则会继续监控新张量，如果没有发出新的张量，阈值间隔规则将被触发。

TalledTraininGRule 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
threshold	一个阈值，定义规则等待张量输出直到触发停滞的训练问题的时间（以秒为单位）。默认值为 1800 秒。 可选 有效值：整数 默认值：1800
stop_training_on_fire	如果设置为True，观察基础训练作业是否输出张量”threshold“秒。 可选 有效值：Boolean 默认值：False
training_job_name_prefix	基本训练任务名称的前缀。如果stop_training_on_fire是的，该规则会在同一账户中搜索带有此前缀的 SageMaker 培训作业。如果发现不活动状态，则规则需要StopTrainingJobaction. 请注意，如果找到多个具有相同前缀的作业，则规则将跳过终止。重要的是，每个培训作业的前缀都是唯一的。 可选 有效值：字符串

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.stalled_training_rule(),
        rule_parameters={
                "threshold": "1800",
                "stop_training_on_fire": "True",
                "training_job_name_prefix": "<specify-training-base-job-name>"
        },
        collections_to_save=[
            CollectionConfig(
                name="losses",
                parameters={
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

TensorVariance

该规则检测您是否具有方差非常高或非常低的张量。张量中的非常高或非常低的方差可能会导致神经元饱和，从而降低神经网络的学习能力。张量非常高的方差最终还会产生张量爆炸问题。使用此规则可以提前检测出此类问题。

此规则可应用于受支持的深度学习框架（TensorFlow、MXNet 和 PyTorch）之一，也可以应用于 XGBoost 算法。必须指定 `collection_names` 或 `tensor_regex` 参数。如果指定了这两个参数，则规则将检查两个集合内张量的并集。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

TensorVariance 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`collection_names`	该规则检查其张量的集合名称列表。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：无
`tensor_regex`	正则表达式模式列表，用于将该比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：无
`max_threshold`	张量方差上限的阈值。 可选 有效值：Float 默认值：无
`min_threshold`	张量方差下限的阈值。 可选 有效值：Float 默认值：无

```
built_in_rules = [
```

```
    Rule.sagemaker(
        base_config=rule_configs.tensor_variance(),
        rule_parameters={
                "collection_names": "weights",
                "max_threshold": "10",
                "min_threshold": "0.00001",
        },
        collections_to_save=[
            CollectionConfig(
                name="weights",
                parameters={
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

UnchangedTensor

此规则检测张量是否不再随步骤而发生更改。

此规则运行numpy.allclose检查张量是否未发生更改的方法。

此规则可应用于受支持的深度学习框架（TensorFlow、MXNet 和 PyTorch）之一，也可以应用于 XGBoost 算法。必须指定 `collection_names` 或 `tensor_regex` 参数。如果指定了这两个参数，则规则将检查两个集合内张量的并集。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

UnchangedTensor 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`collection_names`	该规则检查其张量的集合名称列表。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：无
`tensor_regex`	正则表达式模式列表，用于将该比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：无
`num_steps`	该规则检查以确定张量是否已更改的步骤数。

参数名称	描述
	规则将检查可用的最后 num_steps 个步骤。步骤不必是连续的。如果 num_steps 是 2，则在第 s 个步骤处，规则不一定检查第 s-1 个步骤和第 s 个步骤。如果第 s-1 个步骤不可用，规则会检查最后一个可用步骤以及第 s 个步骤。在这种情况下，规则会检查最后一个可用步骤及当前步骤。 可选 有效值：整数 默认值：3
rtol	要传递给 numpy.allclose 方法的相对容差参数。 可选 有效值：Float 默认值：1e-05
atol	要传递给 numpy.allclose 方法的绝对容差参数。 可选 有效值：Float 默认值：1e-08
equal_nan	比较 NaN 时是否视为相等。如果为 True，则在输出数组中将输入数组 a 中的 NaN 视为等于输入数组 b 中的 NaN。此参数传递给 numpy.allclose 方法。 可选 有效值：Boolean 默认值：False

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.unchanged_tensor(),
        rule_parameters={
                "collection_names": "losses",
                "tensor_regex": "",
                "num_steps": "3",
                "rtol": "1e-05",
                "atol": "1e-08",
                "equal_nan": "False"
        },
        collections_to_save=[
            CollectionConfig(
                name="losses",
                parameters={
                    "save_interval": "500"
```

```
                }
            )
        ]
    )
]
```

CheckInputImages

此规则检查输入图像是否已正确归一化。具体而言，规则会检测样本数据的平均值与零之间的差异是否超过阈值。许多计算机视觉模型要求输入数据的平均值和单位方差为零。

此规则适用于深度学习应用程序。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

CheckInputImages 规则的参数描述

参数名称	描述
`base_trial`	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
`threshold_mean`	一个阈值，该阈值定义输入数据的平均值可以与 0 相差多少。 可选 有效值：Float 默认值：`0.2`
`threshold_samples`	在引发错误之前必须采样的图像数量。如果值过低，则对数据集平均值的估计将不准确。 可选 有效值：整数 默认值：`500`
`regex`	输入数据张量的名称。 可选 有效值：字符串 默认值：`".*hybridsequential0_input_0"`（使用 HybridSequential 的 Apache MXNet 模型的输入张量的名称）
`channel`	颜色通道在输入张量形状数组中的位置。 可选 有效值：整数

参数名称	描述
	默认值：1（例如，MXNet 需要输入数据采用以下形式：(batch_size, channel, height, width)

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.check_input_images(),
        rule_parameters={
                "threshold_mean": "0.2",
                "threshold_samples": "500",
                "regex": ".*hybridsequential0_input_0",
                "channel": "1"
        },
        collections_to_save=[
            CollectionConfig(
                name="custom_inputs_collection",
                parameters={
                    "include_regex": ".*hybridsequential0_input_0"
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

NLPSequenceRatio

给定其余输入序列（对于优化性能很有用）的情况下，此规则将计算特定令牌的比率。例如，您可以计算输入序列中填充句尾 (EOS) 令牌的百分比。如果 EOS 令牌数量过高，则应执行替代分桶策略。您还可以计算输入序列中未知令牌的百分比。如果未知单词的数量过高，可以使用替代词汇表。

此规则适用于深度学习应用程序。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

NLPSequenceRatio 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
tensor_regex	正则表达式模式列表，用于将该比较限制为特定的标量值张量。该规则仅检查与列表中指定的正则表达式模式匹配的张量。如果未传递任何模式，则该规则默认情况下比较试验中收集的所有张量。只能匹配标量值张量。 可选 有效值：字符串列表或逗号分隔的字符串 默认值：".*embedding0_input_0"（假设作为网络的初始层嵌入）

参数名称	描述
token_values	一个由令牌数值列表构成的字符串。例如，"3, 0"。 可选 有效值：以逗号分隔的数值字符串 默认值：0
token_thresholds_percent	一个字符串，由与 token_values 的每个令牌对应的阈值（以百分比为单位）列表构成。例如，"50.0, 50.0"。 可选 有效值：以逗号分隔的浮点值字符串 默认值："50"

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.nlp_sequence_ratio(),
        rule_parameters={
                "tensor_regex": ".*embedding0_input_0",
                "token_values": "0",
                "token_thresholds_percent": "50"
        },
        collections_to_save=[
            CollectionConfig(
                name="custom_inputs_collection",
                parameters={
                    "include_regex": ".*embedding0_input_0"
                }
            )
        ]
    )
]
```

Confusion

此规则评估分类问题的混淆矩阵的优点。

它创建一个大小为 category_no*category_no 的矩阵，并使用来自 (labels, predictions) 对的数据填充该矩阵。对于每个 (labels, predictions) 对，confusion[labels][predictions] 中的计数增加 1。当完全填充矩阵后，按以下所示计算数据对角线值和非对角线值的比率：

- 对于对角线上的元素：confusion[i][i]/sum_j(confusion[j][j])>=min_diag
- 对于非对角线上的元素：confusion[j][i])/sum_j(confusion[j][i])<=max_off_diag

此规则可应用于 XGBoost 算法。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

Confusion 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
category_no	类别的数量。 可选 有效值：整数 ≥2 默认值："None"
labels	这些区域有：labels张量集合或真实标签的 1-d 向量。 可选 有效值：字符串 默认值："labels"
predictions	这些区域有：predictions张量集合或估计标签的 1-d 向量。 可选 有效值：字符串 默认值："predictions"
labels_collection	该规则针对 labels 检查此集合中的张量。 可选 有效值：字符串 默认值："labels"
predictions_collection	该规则针对 predictions 检查此集合中的张量。 可选 有效值：字符串 默认值："predictions"
min_diag	对角线上数据比率的最小阈值。 可选 有效值：0≤float ≤1 默认值：0.9
max_off_diag	对角线以外的数据比率的最大阈值。

参数名称	描述
	可选 有效值：0≤float ≤1 默认值：0.1

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.confusion(),
        rule_parameters={
                "category_no": "10",
                "labels": "labels",
                "predictions": "predictions",
                "labels_collection": "labels",
                "predictions_collection": "predictions",
                "min_diag": "0.9",
                "max_off_diag": "0.1"
        },
        collections_to_save=[
            CollectionConfig(
                name="labels",
                parameters={
                    "save_interval": "500"
                }
            ),
            CollectionConfig(
                name="predictions",
                parameters={
                    "include_regex": "500"
                }
            )
        ]
    )
]
```

Note

如果未指定可选参数的值，则此规则会推断采用默认值。

功能重要性超重

此规则累积每个步骤 n 个最大要素重要性值的权重，并确保它们不会超过阈值。例如，您可以将前 3 个要素的阈值设置为不超过模型总权重的 80%。

此规则仅对 XGBoost 算法有效。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

功能重要性超重规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串

参数名称	描述
threshold	定义累计总和的比例的阈值n最大的功能。该数字n由定义nfeatures参数。 可选 有效值：Float 默认值：0.8
nfeatures	最大功能的数量。 可选 有效值：整数 默认值：3
tensor_regex	张量的正则表达式（正则表达式）命名要分析的规则。 可选 有效值：字符串 默认值：".*feature_importance/weight"

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.feature_importance_overweight(),
        rule_parameters={
                "threshold": "0.8",
                "nfeatures": "3",
                "tensor_regex": ".*feature_importance/weight"
        },
        collections_to_save=[
            CollectionConfig(
                name="feature_importance",
                parameters={
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

TreeDepth

此规则测量 XGBoost 模型中树的深度。XGBoost 会拒绝拆分，如果拆分不能改进损失。这会将训练规范化。因此，树的深度可能无法达到由depth参数。

此规则仅对 XGBoost 算法有效。

有关如何配置和部署内置规则的示例，请参阅配置调试程序内置规则 (p. 1810)。

TreeDepth 规则的参数描述

参数名称	描述
base_trial	基本试用训练任务名称。Amazon SageMaker 调试器会自动将此参数设置为当前的训练作业。 必填 有效值：字符串
depth	树的深度。通过计算最大节点 ID 的以 2 为底的对数，可以获得树的深度。 可选 有效值：Float 默认值：4

```
built_in_rules = [
    Rule.sagemaker(
        base_config=rule_configs.tree_depth(),
        rule_parameters={
                "depth": "4"
        },
        collections_to_save=[
            CollectionConfig(
                name="tree",
                parameters={
                    "save_interval": "500"
                }
            )
        ]
    )
]
```

为训练 Job 分析创建调试器自定义规则

您可以使用调试器规则 API 和开源创建自定义规则来监控训练作业smdebug Python 库它提供了构建自己的规则容器的工具。

主题

- 创建调试程序自定义规则的前提 (p. 1866)
- 使用调试器客户端库smdebug创建自定义规则 Python 脚本 (p. 1867)
- 使用调试器 API 运行自己的自定义规则 (p. 1867)

创建调试程序自定义规则的前提

要创建调试程序自定义规则，您需要满足以下先决条件。

- SageMaker 调试器规则。自定义 API
- 开源 smdebug Python 库
- 你自己的自定义规则 python 脚本
- Amazon SageMaker 调试程序的自定义规则评估程序的注册表 URL (p. 1934)

使用调试器客户端库smdebug创建自定义规则 Python 脚本

这些区域有：smdebug规则 API 提供了一个界面来设置自己的自定义规则。以下 python 脚本是如何构建自定义规则的示例，CustomGradientRule. 本教程自定义规则监视渐变是否太大并将默认阈值设置为 10。自定义规则在启动训练作业时采用 SageMaker 估算器创建的基本试用版。

```
from smdebug.rules.rule import Rule

class CustomGradientRule(Rule):
    def __init__(self, base_trial, threshold=10.0):
        super().__init__(base_trial)
        self.threshold = float(threshold)

    def invoke_at_step(self, step):
        for tname in self.base_trial.tensor_names(collection="gradients"):
            t = self.base_trial.tensor(tname)
            abs_mean = t.reduction_value(step, "mean", abs=True)
            if abs_mean > self.threshold:
                return True
        return False
```

您可以在同一个 python 脚本中添加多个自定义规则类，并通过在下一节中构建自定义规则对象来将它们部署到任何训练作业试验中。

使用调试器 API 运行自己的自定义规则

以下代码示例演示如何使用Amazon SageMaker Python 开发工具包. 此示例假定您在上一步中创建的自定义规则脚本位于 'path/to/my_custom_rule.py'。

```
from sagemaker.debugger import Rule, CollectionConfig

custom_rule = Rule.custom(
    name='MyCustomRule',
    image_uri='759209512951.dkr.ecr.us-west-2.amazonaws.com/sagemaker-debugger-rule-
evaluator:latest',
    instance_type='ml.t3.medium',
    source='path/to/my_custom_rule.py',
    rule_to_invoke='CustomGradientRule',
    collections_to_save=[CollectionConfig("gradients")],
    rule_parameters={"threshold": "20.0"}
)
```

以下列表解释了调试程序。Rule.customAPI 参数。

- name(str): 根据需要指定自定义规则名称。
- image_uri(str): 这是具有理解自定义规则逻辑的容器的图像。它来源和评估你在训练作业中保存的指定张量集合。您可以从以下位置找到开源 SageMaker 规则评估器图像列表：Amazon SageMaker 调试程序的自定义规则评估程序的注册表 URL (p. 1934).
- instance_type(str): 您需要指定一个实例才能构建规则 docker 容器。这将与训练容器并行旋转实例。
- source(str): 这是自定义规则脚本的本地路径或 Amazon S3 URI。
- rule_to_invoke(str): 这指定了自定义规则脚本中的特定规则类实现。SageMaker 在规则作业中一次只支持一个要评估的规则。
- collections_to_save(str): 这指定了为了运行规则，您将保存哪些张量集合。
- rule_parameters（字典）：这接受字典格式的参数输入。您可以调整在自定义规则脚本中配置的参数。

在设置完custom_rule对象，您可以使用它为任何训练作业构建 SageMaker 估算器。指定entry_point到你的训练脚本中。您无需对训练脚本进行任何更改。

```
from sagemaker.tensorflow import TensorFlow

estimator = TensorFlow(
                role=sagemaker.get_execution_role(),
                base_job_name='smdebug-custom-rule-demo-tf-keras',
                entry_point='path/to/your_training_script.py'
                train_instance_type='ml.p2.xlarge'
                ...

                # debugger-specific arguments below
                rules = [custom_rule]
)

estimator.fit()
```

有关使用调试器自定义规则的更多变体和高级示例，请参阅以下示例笔记本。

- 使用 Amazon SageMaker 调试器自定义规则监控训练作业
- PyTorch 迭代模型修剪 ResNet 和 AlexNet
- 使用调试程序规则触发 Amazon CloudWatch Events 以使用 TensorFlow 根据训练状态执行操作

将调试器用于自定义训练容器

Amazon SageMaker 调试器适用于您带给 Amazon SageMaker 的任何深度学习模型。这些区域有：Amazon CLI，SageMakerEstimatorAPI 和调试器 API 使您能够使用任何 Docker 基本映像来构建和自定义容器以训练模型。要将调试器与自定义容器一起使用，您需要对训练脚本进行最小的更改，以实现调试器钩子回调并从训练作业中检索张量。

您需要以下资源来使用调试器构建自定义容器。

- Amazon SageMaker Python 开发工具包
- SMDebug 开源客户端库
- 你选择的 Docker 基本映像
- 注册了调试器钩子的训练脚本 — 有关将调试器钩子注册到训练脚本的更多信息，请参阅将调试器挂钩注册到你的训练脚本 (p. 1869).

有关将调试程序与自定义训练容器结合使用的端到端示例，请参阅以下示例笔记本。

- 使用调试器构建自定义训练容器和调试训练作业

> **Tip**
>
> 这个带调试器指南的自定义容器是调整您自己的训练容器 (p. 2416)指南向您详细介绍如何构建自定义培训容器并将其推送到 Amazon ECR。

准备构建自定义训练容器

要构建 docker 容器，文件的基本结构应如下所示：

```
### debugger_custom_container_test_notebook.ipynb      # a notebook to run python snippet
 codes
```

```
### debugger_custom_container_test_folder                    # this is a docker folder
    ###  your-training-script.py                             # your training script with Debugger
 hook
    ###  Dockerfile                                          # a Dockerfile to build your own
 container
```

将调试器挂钩注册到你的训练脚本

要调试模型训练，您需要在训练脚本中添加调试程序挂钩。

> **Note**
>
> 此步骤是收集模型参数（输出张量）以调试模型训练所必需的。如果您只想监控和配置文件，则可跳过该挂钩注册步骤并排除`debugger_hook_config`构造估计器时的参数。

以下示例代码显示了使用 Keras resNet50 模型的训练脚本的结构以及如何将调试器挂接作为 Keras 回调函数传递以进行调试。要查找完整的训练脚本，请参阅使用 SageMaker 调试器挂钩的 TensorFlow 训练脚本.

```
# An example of training script (your-training-script.py)
import tensorflow.compat.v2 as tf
from tensorflow.keras.applications.resnet50 import ResNet50
import smdebug.tensorflow as smd

def train(batch_size, epoch, model, hook):

    ...
    model.fit(X_train, Y_train,
              batch_size=batch_size,
              epochs=epoch,
              validation_data=(X_valid, Y_valid),
              shuffle=True,

              # smdebug modification: Pass the Debugger hook in the main() as a Keras
 callback
              callbacks=[hook])

def main():
    parser=argparse.ArgumentParser(description="Train resnet50 cifar10")

    # hyperparameter settings
    parser.add_argument(...)

    args = parser.parse_args()

    model=ResNet50(weights=None, input_shape=(32,32,3), classes=10)

    # Add the following line to register the Debugger hook for Keras.
    hook=smd.KerasHook.create_from_json_file()

    # Start the training.
    train(args.batch_size, args.epoch, model, hook)

if __name__ == "__main__":
    main()
```

有关为受支持的框架和算法注册调试器挂接的更多信息，请参阅 SMDebug 客户端库中的以下链接：

- SMDebug TensorFlow 挂钩
- SMDebug PyTorch 挂钩
- smDebug MXNet 挂钩

- SmDebug XGBoost 挂钩

在以下示例笔记本的训练脚本中，您可以找到更多关于如何将调试器挂钩添加到训练脚本中并详细收集输出张量的示例：

- 使用 TensorFlow 2.1 框架的脚本模式下的调试程序

  要查看在深度学习容器中与在脚本模式下使用调试程序之间的区别，请打开该笔记本并将其与前面的“深度学习容器 Tensorflow v2.1 中的调试程序”笔记本示例并排放置。

  在脚本模式下，挂钩配置部分将从您设置评估程序的脚本中删除。而将调试程序挂钩功能合并到训练脚本中，脚本模式下的 TensorFlow Keras resNet 训练脚本. 训练脚本在所需的 Tensorflow Keras 环境中导入 `smdebug` 库以与 Tensorflow ResNet50 算法进行通信。它还会手动实施 `smdebug` 挂钩功能，方法是在 `train` 函数中添加 `callbacks=[hook]` 参数（第 49 行），并添加通过 SageMaker Python 开发工具包提供的手动挂钩配置（第 89 行）。

  该脚本模式示例在 TF 2.1 框架中运行训练作业，以便与 TF 2.1 示例中的零脚本更改进行直接比较。在脚本模式下设置调试程序的好处是可以灵活地选择框架版本未涵盖的框架版本。AmazonDeep Learning Containers。
- 以脚本模式在 PyTorch 容器中使用 Amazon SageMaker 调试程序

  该笔记本在 PyTorch v1.3.1 框架中以脚本模式启用调试程序。SageMaker 容器支持 PyTorch v1.3.1，该示例显示了如何修改训练脚本的详细信息。

  默认情况下，SageMaker PyTorch 评估程序已处于脚本模式下。在该笔记本中，在评估程序配置中不包含激活 `script_mode` 的行。

  本笔记本显示了更改的详细步骤原始 PyTorch 训练脚本到启用了调试器的修改版本. 此外，该示例还说明了如何使用调试程序内置规则检测训练问题（如梯度消失问题），以及如何使用调试程序试验功能调用和分析保存的张量。

创建和配置 Dockerfile

打开 SageMaker JupyterLab 并创建一个新文件夹，`debugger_custom_container_test_folder`在此示例中，要保存训练脚本并`Dockerfile`. 下面的代码示例是`Dockerfile`这包括必不可少的 docker 构建赞扬。将以下代码粘贴到`Dockerfile`并将其保存到文本文件。将训练脚本上传到同一个文件夹。

```
# Specify a docker base image
FROM tensorflow/tensorflow:2.2.0rc2-gpu-py3
RUN /usr/bin/python3 -m pip install --upgrade pip
RUN pip install --upgrade protobuf

# Install required packages to enable the SageMaker Python SDK and the smdebug library
RUN pip install sagemaker-training
RUN pip install smdebug
CMD ["bin/bash"]
```

如果你想使用预构建的Amazon深度学习容器镜像，请参阅AvailableAmazonDeep Learning Containers 映像.

构建自定义培训容器并将其推送到亚马逊 ECR

创建测试笔记本，`debugger_custom_container_test_notebook.ipynb`，并在笔记本单元中运行以下代码。这将访问`debugger_byoc_test_docker`目录中，使用指定的构建 docker`algorithm_name`，然后将码头容器推送到您的亚马逊 ECR 中。

```
import boto3
```

```
account_id = boto3.client('sts').get_caller_identity().get('Account')
ecr_repository = 'sagemaker-debugger-mnist-byoc-tf2'
tag = ':latest'

region = boto3.session.Session().region_name

uri_suffix = 'amazonaws.com'
if region in ['cn-north-1', 'cn-northwest-1']:
    uri_suffix = 'amazonaws.com.cn'
byoc_image_uri = '{}.dkr.ecr.{}.{}/{}'.format(account_id, region, uri_suffix,
 ecr_repository + tag)

!docker build -t $ecr_repository docker
!$(aws ecr get-login --region $region --registry-ids $account_id --no-include-email)
!aws ecr create-repository --repository-name $ecr_repository
!docker tag {ecr_repository + tag} $byoc_image_uri
!docker push $byoc_image_uri
```

Tip

如果您使用其中一个Amazon深度学习容器基础映像，运行以下代码登录 Amazon ECR 并访问深度学习容器映像存储库。

```
! aws ecr get-login-password --region {region} | docker login --username Amazon --
password-stdin 763104351884.dkr.ecr.us-east-1.amazonaws.com
```

使用自定义训练容器运行和调试培训作业

在构建码头容器并将其推送到 Amazon ECR 之后，请使用训练脚本和调试器特定参数配置 SageMaker 估计器。执行之后`estimator.fit()`，调试器将收集输出张量、监控它们并检测训练问题。使用保存的张量，你可以使用smdebug核心功能和工具。使用 Amazon CloudWatch Events 配置调试器规则监控流程的工作流程和Amazon Lambda，只要调试器规则发现训练问题，就可以自动执行停止训练作业流程。

```
import sagemaker
from sagemaker.estimator import Estimator
from sagemaker.debugger import Rule, DebuggerHookConfig, CollectionConfig, rule_configs

profiler_config=ProfilerConfig(...)
debugger_hook_config=DebuggerHookConfig(...)
rules=[
    Rule.sagemaker(rule_configs.built_in_rule()),
    ProfilerRule.sagemaker(rule_configs.BuiltInRule())
]

estimator=Estimator(
    image_uri=byoc_image_uri,
    entry_point="./debugger_custom_container_test_folder/your-training-script.py"
    role=sagemaker.get_execution_role(),
    base_job_name='debugger-custom-container-test',
    instance_count=1,
    instance_type='ml.p3.2xlarge',

    # Debugger-specific parameters
    profiler_config=profiler_config,
    debugger_hook_config=debugger_hook_config,
    rules=rules
)

# start training
estimator.fit()
```

对 Amazon SageMaker 调试器规则的操作

根据调试器规则评估状态，您可以设置自动操作，例如停止培训作业和使用 Amazon Simple Notification Service (Amazon SNS) 发送通知。您还可以使用 Amazon CloudWatch Events 创建自己的操作和Amazon Lambda. 要了解如何根据调试器规则评估状态设置自动操作，请参阅以下主题。

主题

- 调试器内置的规则操作 (p. 1872)
- 使用 Amazon CloudWatch 创建规则操作和Amazon Lambda (p. 1876)

调试器内置的规则操作

使用调试器内置操作来响应调试程序规则 (p. 1827). 调试程序`rule_configs`类提供了配置操作列表的工具，包括在调试器规则发现培训问题时自动停止培训作业和使用 Amazon Simple Notification Service (Amazon SNS) 发送通知。

第 1 步：设置 Amazon SNS、创建 SMDebug 规则主题并订阅主题

本部分将指导您完成如何设置 Amazon SNS**`SMDebugRules`**主题，订阅它，然后确认订阅以接收来自调试器规则的通知。

> Note
>
> 有关 Amazon SNS 账单的更多信息，请参阅Amazon SNS 定价和Amazon SNS 常见问题.

创建 SMDebugrules 主题

1. 访问 https://console.aws.amazon.com/sns/v3/home，登录 Amazon Web Services Management Console 并打开 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择主题。
3. 在 Topics（主题）页面上，选择 Create topic（创建主题）。
4. 在 Create topic（创建主题）页面上，在 Details（详细信息）部分中，执行以下操作：
   a. 适用于类型，选择标准对于主题类型。
   b. 在 Name (名称) 中，输入 **`SMDebugRules`**。
5. 跳过所有其他可选设置然后选择创建主题. 如果要了解有关可选设置的更多信息，请参阅创建 Amazon SNS 主题.

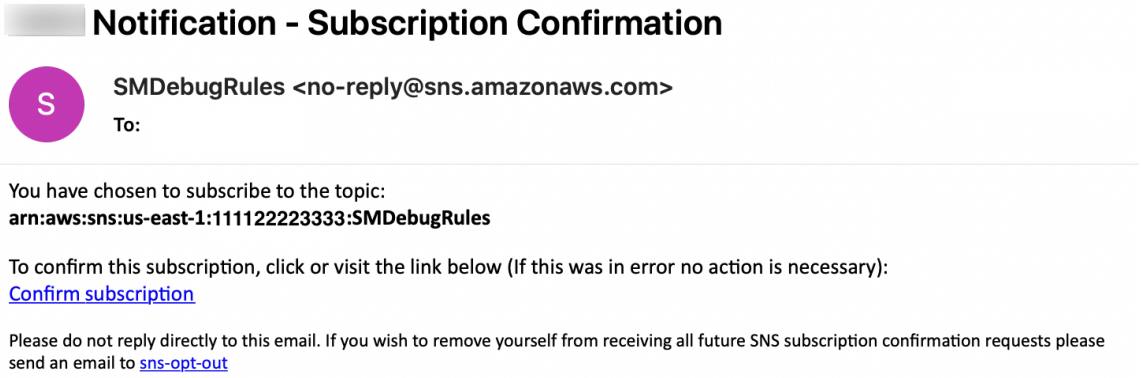
要订阅 SMDebug 规则主题

1. 通过以下网址打开 Amazon SNS 控制台：https://console.aws.amazon.com/sns/v3/home。
2. 在左侧导航窗格中，选择订阅。
3. 在订阅页面上，选择创建订阅。
4. 在创建订阅页上，在详细信息部分中，执行以下操作：
   a. 适用于主题 ARN，选择SMdebuGrules主题 ARN。ARN 应采用`arn:aws:sns:<region-id>:111122223333:SMDebugRules`.
   b. 对于 Protocol (协议)，选择 Email (电子邮件) 或 SMS。
   c. 适用于端点中，输入终端节点值，例如要接收通知的电子邮件地址或电话号码。

      > Note
      >
      > 确保键入正确的电子邮件地址和电话号码。电话号码必须包括+、国家/地区代码和电话号码，没有特殊字符或空格。例如，电话号码 +1 (222) 333-4444 的格式为**`+12223334444`**.

5. 跳过所有其他可选设置然后选择创建订阅. 如果要了解有关可选设置的更多信息，请参阅订阅 Amazon SNS 主题.

在您订阅SMdebuGrules主题，您通过电子邮件或电话收到以下确认消息：



有关 Amazon SNS 的更多信息，请参阅移动文本消息 (SMS)和电子邮件通知中的Amazon SNS 开发人员指南.

第 2 步：设置 IAM 角色以附加必需的策略

在此步骤中，您将所需策略添加到 IAM 角色。

将所需策略添加到 IAM 角色

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择策略，然后选择创建策略.
3. 在存储库的创建策略执行以下操作以创建新的 SNS 访问策略：
   a. 选择 JSON 选项卡。
   b. 将以粗体格式化的 JSON 字符串粘贴到以下代码中的`"Statement"`，取代 12 位数Amazon您的账户 IDAmazon账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sns:Publish",
                "sns:CreateTopic",
                "sns:Subscribe"
            ],
            "Resource": "arn:aws:sns:*:111122223333:SMDebugRules"
        }
    ]
}
```

   c. 在页面底部，选择查看策略.
   d. 在 Create policy（创建策略）页面上，对于 Name（名称），输入 **sns-access**。
   e. 在页面底部，选择创建策略.
4. 返回 IAM 控制台，然后选择角色在左侧导航窗格中。

5. 查找用于 SageMaker 模型培训的 IAM 角色，然后选择该 IAM 角色。
6. 在存储库的Permissions (权限)的选项卡摘要页面上，选择附加策略.
7. 搜索sns-Access选择策略旁边的复选框，然后选择附加策略.

有关为 Amazon SNS 设置 IAM 策略的更多示例，请参阅用于 Amazon SNS 访问控制的示例案例.

第 3 步：使用内置操作配置调试器规则

在上述步骤中成功完成所需的设置后，您可以配置调试规则的调试器内置操作，如以下示例脚本所示。你可以选择在构建时使用哪些内置操作`actions`列出对象。这些区域有：`rule_configs`是一个帮助模块，它提供了用于配置调试器内置规则和操作的高级工具。以下内置操作适用于调试器：

- `rule_configs.StopTraining()`— 当调试器规则发现问题时停止训练作业。
- `rule_configs.Email("abc@abc.com")`— 当调试器规则发现问题时，通过电子邮件发送通知。使用您在设置 SNS 主题订阅时使用的电子邮件地址。
- `rule_configs.SMS("+1234567890")`— 当调试器规则发现问题时，通过短信发送通知。使用设置 SNS 主题订阅时使用的电话号码。

  Note

  确保键入正确的电子邮件地址和电话号码。电话号码必须包括+、国家/地区代码和电话号码，没有特殊字符或空格。例如，电话号码 +1 (222) 333-4444 的格式为**+12223334444**.

您可以使用所有内置动作或动作子集，方法是使用`rule_configs.ActionList()`方法，它采取内置操作并配置操作列表。

将所有三个内置操作添加到单个规则

如果要将所有三个内置操作分配给一个规则，请在构建估计器时配置调试器内置操作列表。使用以下模板构建估计器，调试器将停止训练作业，并通过电子邮件和文本发送通知，告知用于监控训练作业进度的任何规则。

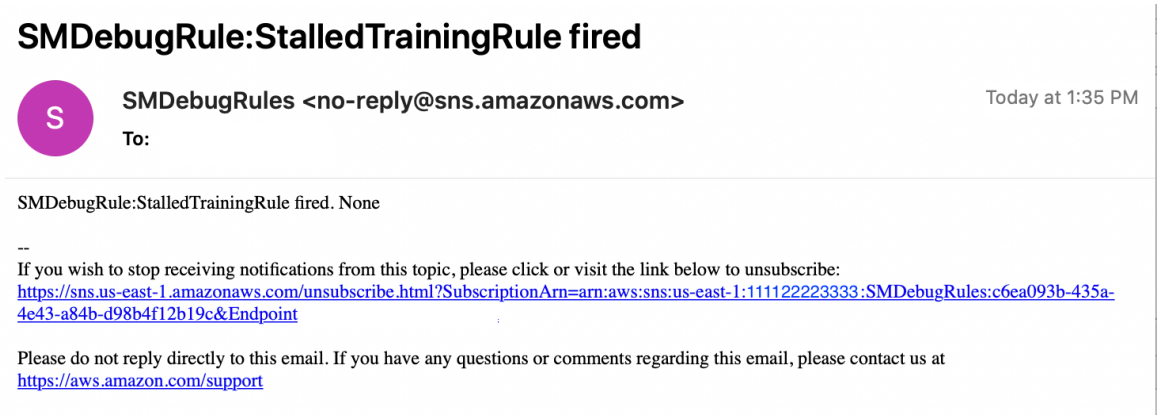
```
from sagemaker.debugger import Rule, rule_configs

# Configure an action list object for Debugger rules
actions = rule_configs.ActionList(
    rule_configs.StopTraining(),
    rule_configs.Email("abc@abc.com"),
    rule_configs.SMS("+1234567890")
)

# Configure rules for debugging with the actions parameter
rules = [
    Rule.sagemaker(
        base_config=rule_configs.built_in_rule(),         # Required
        rule_parameters={"paramter_key": value },        # Optional
        actions=actions
    )
]

estimator = Estimator(
    ...
    rules = rules
)

estimator.fit(wait=False)
```

创建多个内置操作对象以将不同的操作分配给单个规则

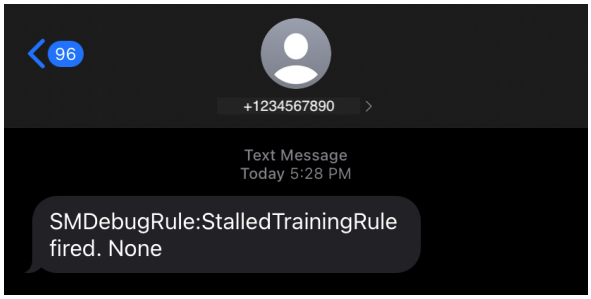
如果要分配以单个规则的不同阈值触发的内置操作，则可以创建多个内置操作对象，如以下脚本所示。要通过运行相同的规则避免冲突错误，必须提交不同的规则作业名称（为规则指定不同的字符串）name属性），如以下示例脚本模板所示。此示例演示如何设置已停止的训练规则 (p. 1855)要采取两种不同的操作：发送电子邮件至abc@abc.com当训练作业停顿 60 秒时，如果停顿 120 秒，则停止训练作业。

```
from sagemaker.debugger import Rule, rule_configs
import time

base_job_name_prefix= 'smdebug-stalled-demo-' + str(int(time.time()))

# Configure an action object for StopTraining
action_stop_training = rule_configs.ActionList(
    rule_configs.StopTraining()
)

# Configure an action object for Email
action_email = rule_configs.ActionList(
    rule_configs.Email("abc@abc.com")
)

# Configure a rule with the Email built-in action to trigger if a training job stalls for
 60 seconds
stalled_training_job_rule_email = Rule.sagemaker(
        base_config=rule_configs.stalled_training_rule(),
        rule_parameters={
                "threshold": "60",
                "training_job_name_prefix": base_job_name_prefix
        },
        actions=action_email
)
stalled_training_job_rule_text.name="StalledTrainingJobRuleEmail"

# Configure a rule with the StopTraining built-in action to trigger if a training job
 stalls for 120 seconds
stalled_training_job_rule = Rule.sagemaker(
        base_config=rule_configs.stalled_training_rule(),
        rule_parameters={
                "threshold": "120",
                "training_job_name_prefix": base_job_name_prefix
        },
        actions=action_stop_training
)
stalled_training_job_rule.name="StalledTrainingJobRuleStopTraining"

estimator = Estimator(
    ...
    rules = [stalled_training_job_rule_email, stalled_training_job_rule]
)

estimator.fit(wait=False)
```

在训练作业运行期间，每当规则发现训练作业存在问题时，调试器内置操作就会发送通知电子邮件和短信。以下屏幕截图显示了培训工作存在停滞问题的培训作业的电子邮件通知示例。



以下屏幕截图显示了在规则发现 StalledTrainING 问题时调试器发送的示例文本通知。



使用调试器内置操作的注意事项

- 要使用调试器内置操作，需要互联网连接。Amazon SageMaker 或 Amazon VPC 提供的网络隔离模式不支持此功能。
- 内置操作不能用于调试程序 ProfilerRule (p. 1826).
- 内置操作不能用于具有现场培训中断的培训作业。
- 在电子邮件或短信通知中，`None`出现在消息的末尾。这没有任何意义，所以你可以忽略文字`None`.

使用 Amazon CloudWatch 创建规则操作和Amazon Lambda

Amazon CloudWatch 收集 Amazon SageMaker 模型训练作业日志和 Amazon SageMaker 调试器规则处理作业日志。使用 Amazon CloudWatch Events 配置调试器和Amazon Lambda以根据调试器规则评估状态采取行动。

用于调试器规则和培训作业的 CloudWatch Logs

查找训练作业日志和调试器规则作业日志

1. 访问 https://console.aws.amazon.com/cloudwatch/，打开 CloudWatch 控制台。
2. 在左侧导航窗格中，日志节点，选择日志组.
3. 在日志组列表中，执行以下操作：
   - 选择/aws/sagemaker/TrainingJobs用于培训工作日志。
   - 选择/AWS/Sagemaker/处理工作对于调试器规则作业日志。

在出现培训问题时，您可以使用 CloudWatch 日志中的训练和调试器规则作业状态采取进一步的操作。

有关使用 CloudWatch 监控训练作业的更多信息，请参阅监控 Amazon SageMaker.

使用 CloudWatch 和 Lambda 为自动终止训练 Job 设置调试器

调试器规则监控训练作业状态，CloudWatch Event 规则监视调试器规则训练作业评估状态。

第 1 步：创建 Lambda 函数

创建 Lambda 函数

1. 打开 Amazon Lambda 控制台，地址：https://console.aws.amazon.com/lambda/。
2. 在左侧导航窗格中，选择函数然后选择创建函数.
3. 在存储库的创建函数页面上，选择从头开始创作选项。
4. 在基本信息部分中，输入函数名称（例如，调试器-规则-停止训练-工作)。
5. 对于 Runtime (运行时)，选择 Python 3.7。
6. 适用于Permissions (权限)，展开下拉选项，然后选择更改默认执行角色.
7. 适用于执行角色，选择使用现有角色然后选择 IAM 角色，用于在 SageMaker 上培训作业。

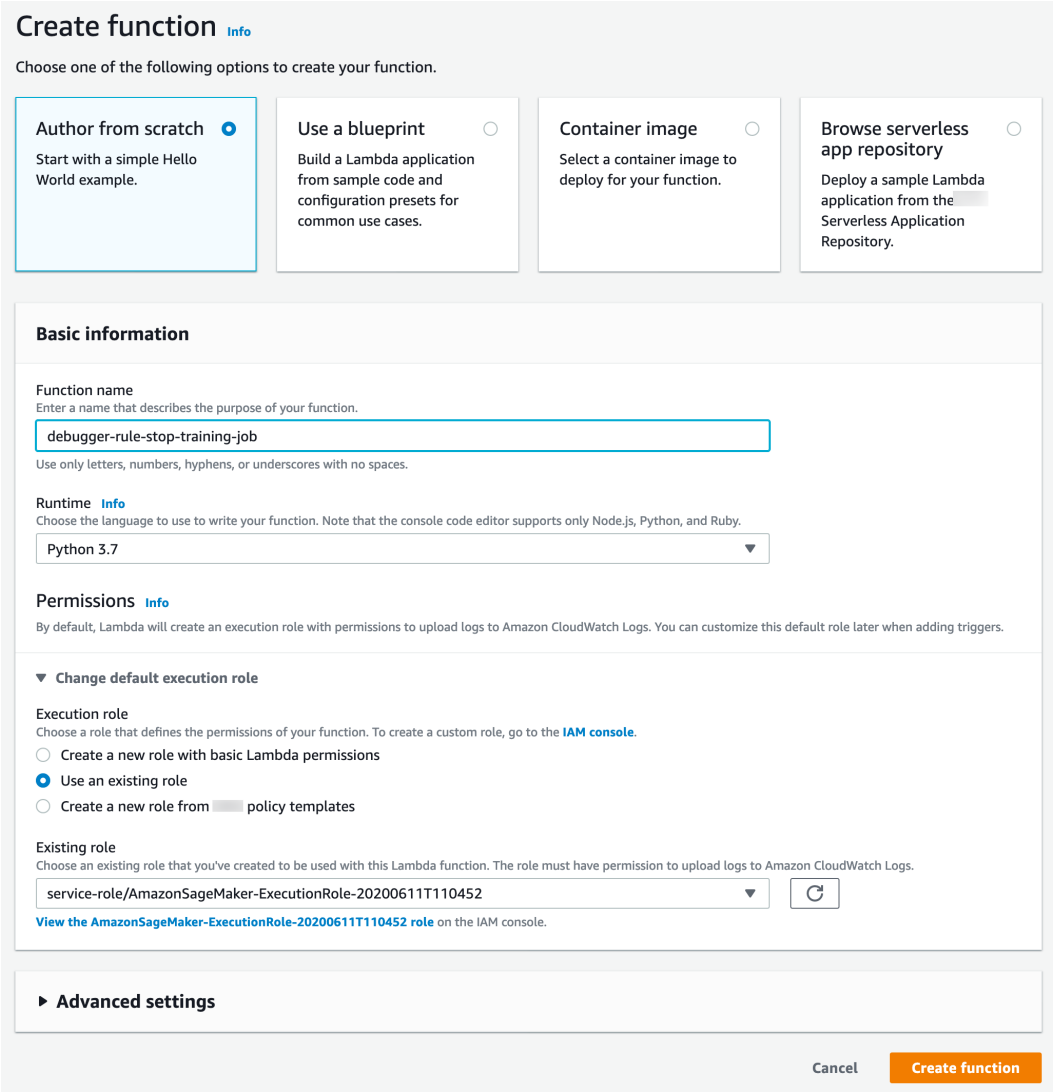
   Note

   确保你使用执行角色`AmazonSageMakerFullAccess`和`AWSLambdaBasicExecutionRole`附加。否则，Lambda 函数将无法对训练作业的调试器规则状态更改做出正确反应。如果您不确定正在使用哪个执行角色，请在 Jupyter notebook 单元格中运行以下代码以检索执行角色输出：

   ```
   import sagemaker
   sagemaker.get_execution_role()
   ```

8. 在页面底部，选择创建函数。

下图显示了创建函数页面中的输入字段和选择已完成。



第 2 步：配置 Lambda 函数

要配置 Lambda 函数

1. 在函数代码将以下 Python 脚本粘贴到 Lambda 代码编辑器窗格中。这些区域有：`lambda_handler`函数监控 CloudWatch 收集的调试器规则评估状态并触发`StopTrainingJob`API 操作。这些区域有：Amazon SDK for Python (Boto3) `client`对于 SageMaker 提供了高级别方法，`stop_training_job`，这会触发`StopTrainingJob`API 操作。

```
import json
import boto3
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    training_job_name = event.get("detail").get("TrainingJobName")
    logging.info(f'Evaluating Debugger rules for training job: {training_job_name}')
    eval_statuses = event.get("detail").get("DebugRuleEvaluationStatuses", None)
```

```
   if eval_statuses is None or len(eval_statuses) == 0:
       logging.info("Couldn't find any debug rule statuses, skipping...")
       return {
           'statusCode': 200,
           'body': json.dumps('Nothing to do')
       }

   # should only attempt stopping jobs with InProgress status
   training_job_status = event.get("detail").get("TrainingJobStatus", None)
   if training_job_status != 'InProgress':
       logging.debug(f"Current Training job status({training_job_status}) is not
'InProgress'. Exiting")
       return {
           'statusCode': 200,
           'body': json.dumps('Nothing to do')
       }

   client = boto3.client('sagemaker')

   for status in eval_statuses:
       logging.info(status.get("RuleEvaluationStatus") + ', RuleEvaluationStatus=' +
str(status))
       if status.get("RuleEvaluationStatus") == "IssuesFound":
           secondary_status = event.get("detail").get("SecondaryStatus", None)
           logging.info(
                   f'About to stop training job, since evaluation of rule
configuration {status.get("RuleConfigurationName")} resulted in "IssuesFound". ' +
                   f'\ntraining job "{training_job_name}" status is
"{training_job_status}", secondary status is "{secondary_status}"' +
                   f'\nAttempting to stop training job "{training_job_name}"'
               )
           )
           try:
               client.stop_training_job(
                   TrainingJobName=training_job_name
               )
           except Exception as e:
               logging.error(
                   "Encountered error while trying to "
                   "stop training job {}: {}".format(
                       training_job_name, str(e)
                   )
               )
               raise e
   return None
```

有关 Lambda 代码编辑器接口的更多信息，请参阅使用AmazonLambda 控制台编辑器.

2. 跳过所有其他设置然后选择Save（保存）在配置页面的顶部。

第 3 步：创建 CloudWatch 事件规则并链接到适用于调试器的 Lambda 函数

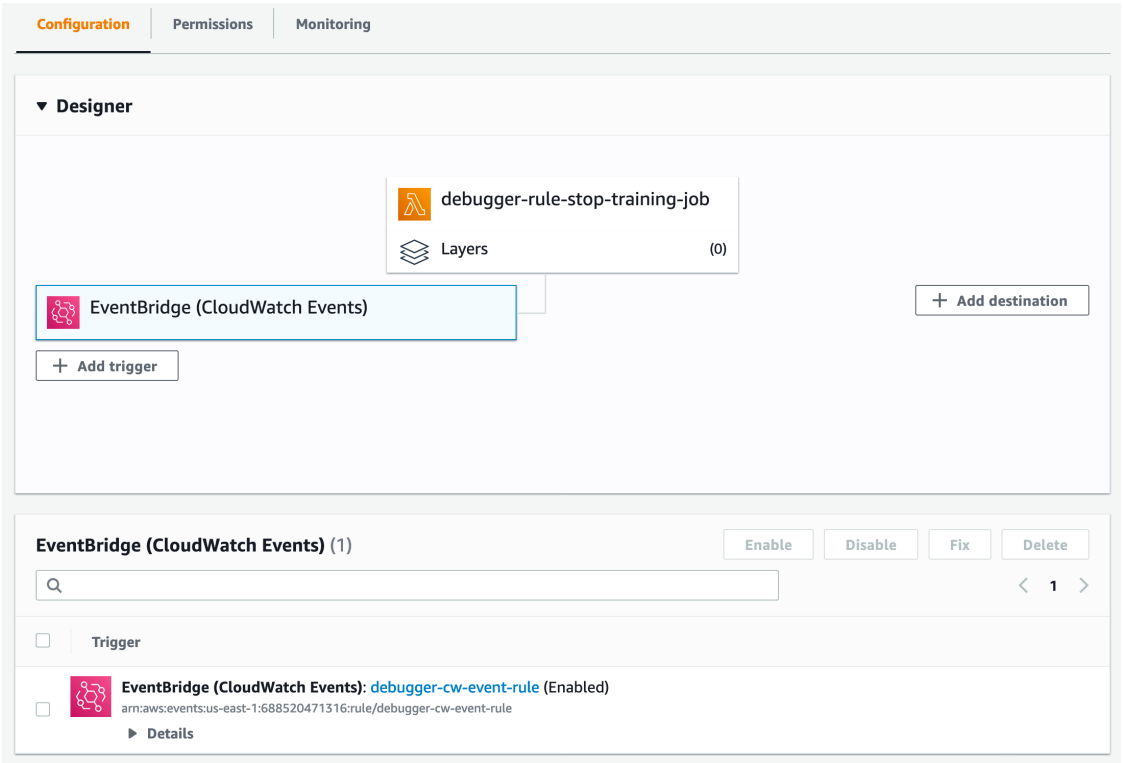
要创建 CloudWatch Events 规则并链接到适用于调试器的 Lambda 函数

1. 访问 https://console.aws.amazon.com/cloudwatch/，打开 CloudWatch 控制台。
2. 在左侧导航窗格中，选择Rule在事件节点。
3. 选择 Create rule (创建规则)。
4. 在事件源的 部分第 1 步：创建规则页面上，选择SageMaker为了服务名称，然后选择SageMaker 培训 Job 状态更改为了事件类型. 事件模式预览应类似于以下示例 JSON 字符串：

```
{
```

```
    "source": [
        "aws.sagemaker"
    ],
    "detail-type": [
        "SageMaker Training Job State Change"
    ]
}
```

5. 在目标部分，选择添加目标 *，然后选择调试器-规则-停止训练-工作您创建的 Lambda 函数。此步骤将 CloudWatch Events 规则与 Lambda 函数相关联。
6. 选择配置细节然后转至步骤 2: 配置规则详细信息页.
7. 指定 CloudWatch 规则定义名称。例如，调试器-cw-事件规则.
8. 选择创建规则要完成操作。
9. 返回 Lambda 函数配置页面并刷新页面。确认它是否在设计师面板。应将 CloudWatch Event 规则注册为 Lambda 函数的触发器。配置设计应类似于以下示例：



运行示例笔记本测试自动化培训 Job 终止

您可以运行以下示例笔记本，这些笔记本已准备好尝试使用调试器的内置规则停止训练作业。

- Amazon SageMaker 调试器-根据规则对 CloudWatch Events 作出反应

  此示例笔记本运行的培训作业存在渐变问题消失。调试程序VanishingGradient (p. 1844)构建 SageMaker TensorFlow 估计器时使用内置规则。当调试器规则检测到问题时，培训作业将终止。
- 使用 SageMaker 调试器规则检测停滞的训练并调用操作

  此示例笔记本运行一个训练脚本，其代码行强制其睡眠 10 分钟。调试程序已停止的训练规则 (p. 1855)内置规则会调用问题并停止训练作业。

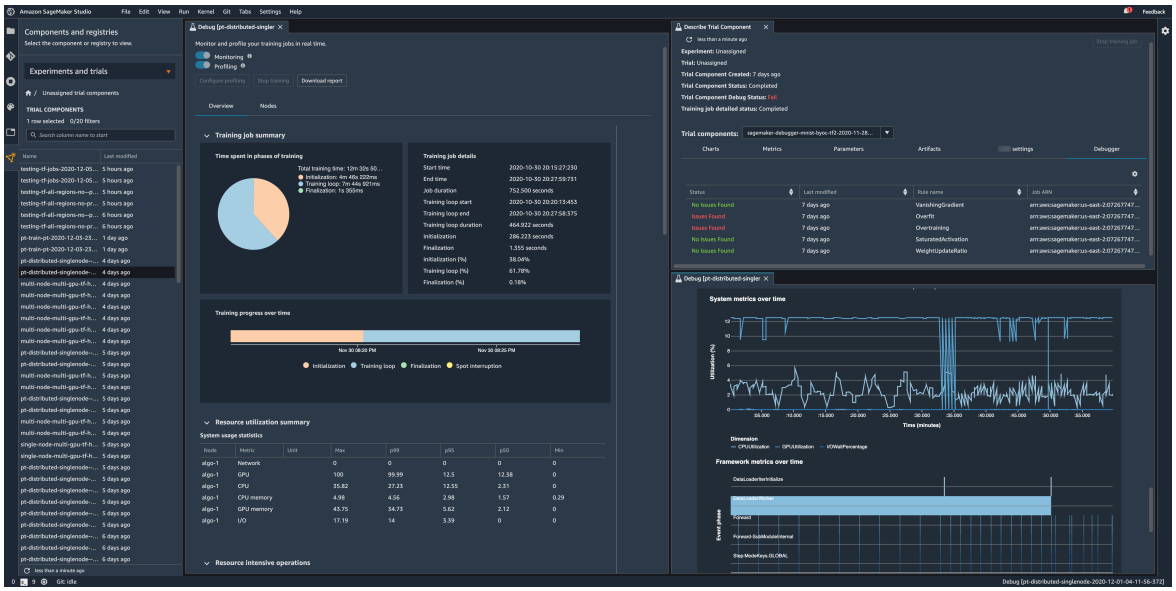
禁用 CloudWatch Event 规则以停止使用自动培训 Job 终止

如果要禁用自动终止培训作业，则需要禁用 CloudWatch Events 规则。在 Lambda 中设计师面板中，选择EventBridge (CloudWatch Events)链接到 Lambda 函数的块。这显示了EventBridge下面的面板设计师面板（例如，请参阅上一个屏幕截图)。选中 旁边的复选框EventBridge（CloudWatch 事件）：调试器-cw 事件规则，然后选择禁用. 如果您想以后使用自动终止功能，可以再次启用 CloudWatch Events 规则。

Amazon SageMaker Studio 中的 Amazon SageMaker 调试器

在亚马逊 Elastic Compute Cloud (Amazon EC2) 实例上运行培训作业时，使用 Amazon SageMaker Studio 中的 Amazon SageMaker 调试器仪表板来分析您的模型性能和系统瓶颈。借助调试器仪表板，深入了解您的训练工作，提高模型训练绩效和准确性。默认情况下，调试器每 500 毫秒监控系统指标（CPU、GPU、CPU 和 GPU 内存、网络和数据 I/O），每 500 次迭代用于训练作业的基本输出张量（损耗和准确度）。您还可以进一步自定义调试器配置参数值，并通过 Studio UI 或使用Amazon SageMaker Python 开发工具包.

> **Important**
>
> 如果您使用的是现有的 Studio 应用程序，请重新启动它们以使用新功能。有关如何重启和更新 Studio 环境的说明，请参阅更新 Amazon SageMaker Studio.



主题

- 打开 Amazon SageMaker 调试程序见解控制面板 (p. 1881)
- Amazon SageMaker 调试器见解仪表板控制器 (p. 1882)
- Amazon SageMaker 调试器见解仪表板 (p. 1886)
- 关闭 Amazon SageMaker 调试器见解实例 (p. 1893)
- Studio 实验中的 Amazon SageMaker 调试器 (p. 1894)

打开 Amazon SageMaker 调试程序见解控制面板

在 Studio 中打开调试器见解仪表板，查看分析进度、资源利用率结果以及在 Amazon EC2 实例上运行的培训作业的系统瓶颈。

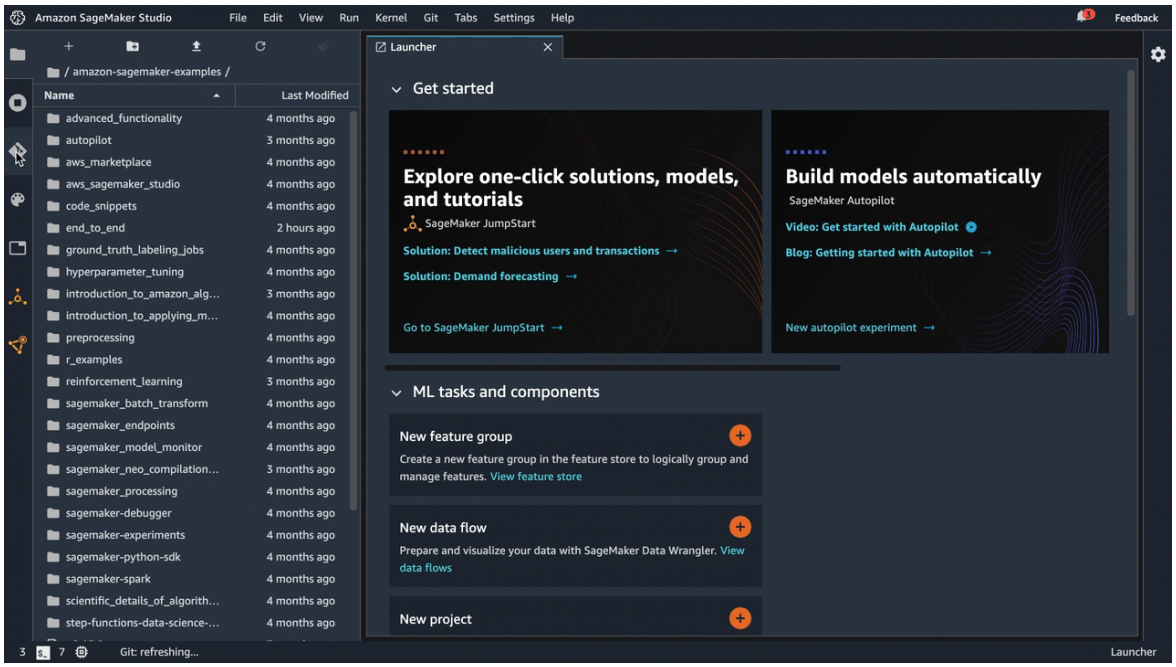
Note

Studio 调试器见解仪表板在ml.m5.4xlarge实例来处理和渲染可视化效果。每个调试器见解选项卡运行一个 Studio 内核会话。在单个实例上运行多个调试器见解选项卡的多个内核会话。关闭“调试器见解”选项卡时，相应的内核会话也将关闭。Studio 应用程序仍然处于活动状态，并会计ml.m5.4xlarge实例使用情况。有关定价的信息，请参阅Amazon SageMaker 定价页.

Important

使用调试器见解仪表板完成后，必须关闭ml.m5.4xlarge实例以避免累积费用。有关如何关闭实例的说明，请参阅关闭 Amazon SageMaker 调试器见解实例 (p. 1893).

打开调试程序见解控制面板



1. 选择SageMaker 组件和注册表图标 ( )。
2. 打开下拉列表，然后选择实验和试验.
3. 查找你的训练作业名称。如果您尚未将 SageMaker 实验试用组件分配给训练作业，则该作业将在未分配试用组件列出。
4. 右键单（或等效的 UI 交互）以打开培训工作试用组件的上下文菜单。有两个菜单项可以访问 Studio 中的调试器功能：打开调试程序获取洞察和在试用详情中打开.
5. 选择打开调试程序获取洞察. 这将打开调试 [你的训练-工作名称]选项卡。在此选项卡上，调试器概述了您在 Amazon EC2 实例上的模型训练性能，并确定了系统瓶颈问题。虽然监控系统资源利用率，你也可以启用分析捕获框架指标，这些指标包括来自前向和向后传递以及数据加载期间执行的神经网络操作的数据。有关如何启用的更多信息分析使用调试器见解仪表板控制器，请参阅启用和配置调试器分析以获取详细见解 (p. 1883).

调试器将系统资源利用率指标与框架指标关联起来，并帮助识别可能是系统瓶颈根源的资源密集型操作员。您还可以下载汇总的调试器性能分析报告。有关更多信息，请参阅 Amazon SageMaker 调试器见解仪表板控制器 (p. 1882)。

Amazon SageMaker 调试器见解仪表板控制器

调试器控制器有不同的组件用于监视和分析。在本指南中，您将了解调试程序控制器组件。

Note

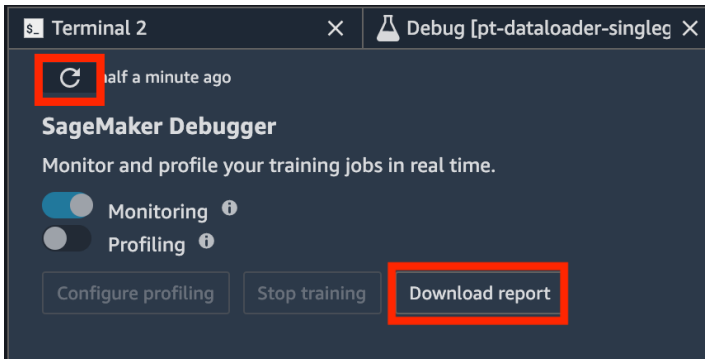
Studio 调试器见解仪表板在ml.m5.4xlarge实例来处理和渲染可视化效果。每个调试器见解选项卡运行一个 Studio 内核会话。在单个实例上运行多个调试器见解选项卡的多个内核会话。关闭“调试器见解”选项卡时，相应的内核会话也将关闭。Studio 应用程序仍然处于活动状态，并会计ml.m5.4xlarge实例使用情况。有关定价的信息，请参阅Amazon SageMaker 定价页.

Important

使用调试器见解仪表板完成后，请关闭ml.m5.4xlarge实例以避免累积费用。有关如何关闭实例的说明，请参阅关闭 Amazon SageMaker 调试器见解实例 (p. 1893).

SageMaker 调试器见解控制器 UI

使用位于见解仪表板左上角的调试器控制器，您可以刷新仪表板、配置或更新用于监控系统指标和分析框架指标的调试器设置、停止训练作业以及下载调试器性能分析报告。



- 如果你想手动刷新调试 [你的训练-工作名称]页面中，选择刷新按钮（左上角的圆形箭头），如前面的屏幕截图所示。
- 监控默认情况下，对于任何 SageMaker 培训作业启用。通过监控你的训练任务，调试器只收集系统指标来检测资源利用率问题，例如 CPU 瓶颈和 GPU 利用不足。有关调试程序监控的资源利用率问题的完整列表，请参阅调试器用于监视硬件系统资源利用率的内置规则（系统指标）(p. 1826).
- 要下载包含培训作业的详细信息和分析的综合调试器性能分析报告，请选择下载报告. 有关调试程序性能分析报告的更多信息，请参阅SageMaker 调试器性能分析报告 (p. 1897).

启用和配置调试器分析以获取详细见解

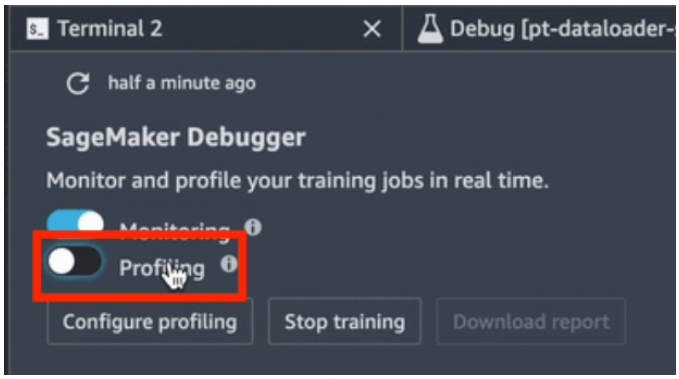
启用时分析，调试器开始收集框架指标。框架指标是从模型的机器学习框架操作中收集的模型数据，例如向前传递、向后传递、批量标准化和数据加载器流程。调试器将系统性能瓶颈与框架操作关联起来，然后运行用于分析框架指标的调试器内置规则 (p. 1827).

Note

启用分析后，调试器会收集每个步骤中执行的每个框架操作调用：在向前传递期间向下卷积输入图层的操作、在向后传递期间更新数百万神经元的权重以及数据加载器进程。尽管分析可以帮助您更深入地了解模型性能，但收集框架指标可能会影响训练时间和绩效。我们建议您启用分析功能，以便一次最多检查两个步骤。有关如何使用以下功能配置调试程序的更多信息：Amazon SageMaker Python 开发工具包，请参阅配置调试器框架分析 (p. 1801)和在训练 Job 运行时更新调试器系统监控和框架分析配置 (p. 1804).

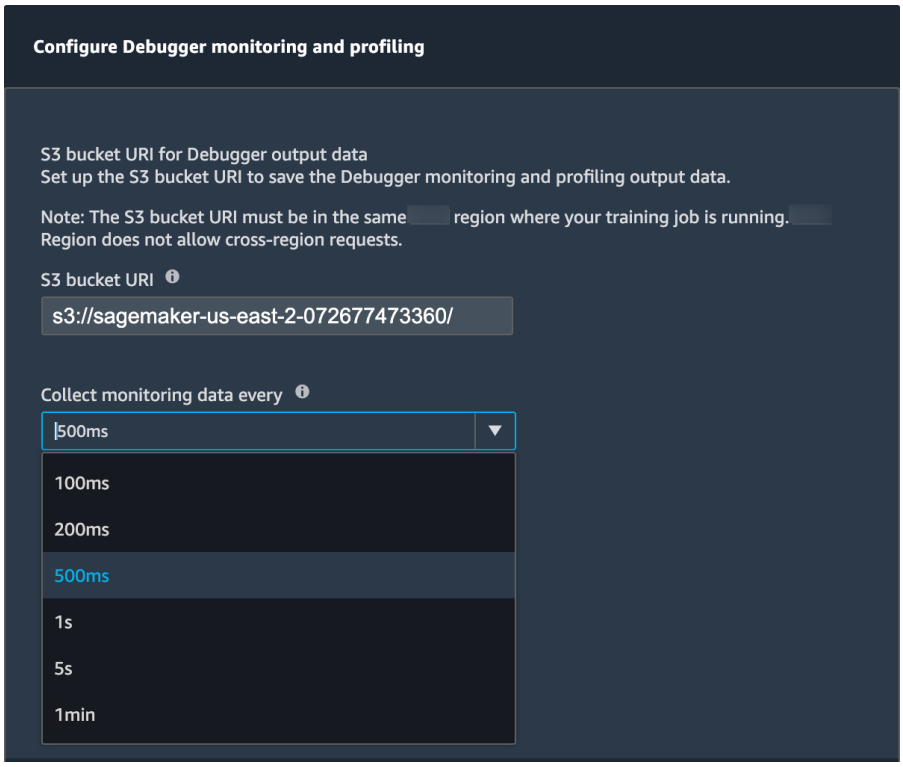
如果要在训练作业运行期间启用分析功能，请使用以下步骤开始分析。

1. 在 Studio 中，打开分析以启用调试器框架分析。这将打开调试器监视和分析配置页面。



2. 这些区域有：配置调试器监控和分析、S3 存储桶 URI和每隔收集监控数据字段已设置为默认值。

   您可以使用下拉列表选择更改监视间隔。从以下可用选项中进行选择：100 毫秒、200 毫秒、500 毫秒（默认值）、1 秒、5 秒和 1 分钟。



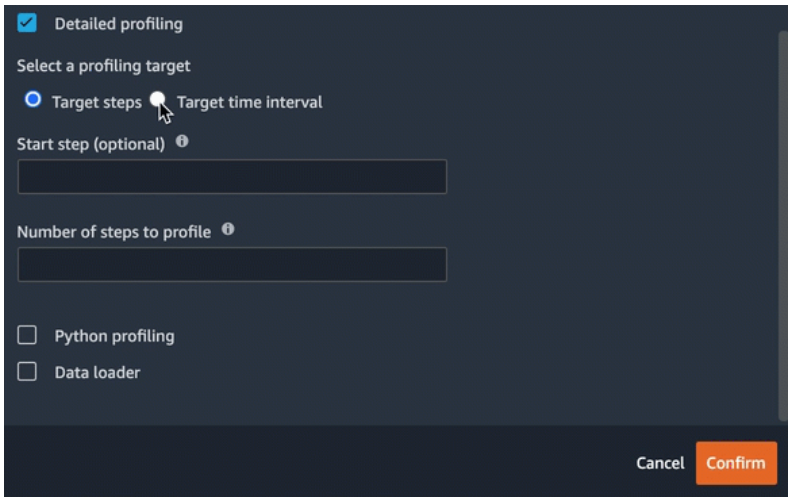
   指定以下字段的值：

   - S3 存储桶 URI：指定基本 S3 存储桶 URI。
   - 每隔收集监控数据：选择时间间隔以收集系统指标。

     Note

     如果选择较低的时间间隔之一，则可以增加监控系统指标的粒度。它允许您以更高的时间分辨率捕获尖峰和异常。但是，随着要处理的系统指标的规模按比例增加，可能会影响整体培训和处理时间。

3. In分析的高级设置中，配置框架指标分析选项。指定开始步骤（或开始时间）和配置文件的步骤数（或个人资料的持续时间）来配置文件。您也可以将输入字段留空。默认值会自动配置为使用当前步骤和 1 步持续时间。



- 详细的分析：指定目标步长或时间范围，以使用本机框架分析器（TensorFlow 分析器和 PyTorch 分析器）来分析框架操作。例如，如果您使用的是 TensorFlow，则调试器钩子使 TensorFlow 分析器能够收集 TensorFlow 特定的框架指标。通过详细的分析，您可以在培训作业的前一步（在第一步之前）、步骤内以及在培训作业的步骤之间对所有框架运营商进行分析。

  Note

  详细的分析可能会显著增加 GPU 内存消耗。建议不要在几个以上的步骤中启用详细的分析。

- Python 分析：指定目标步长或时间范围来分析 Python 函数。您还可以在两个 Python 分析器之间进行选择：cProfile 和 Py仪器。
  - cProfile — 标准 Python 分析器。cProfile 为训练期间调用的每个 Python 运算符收集。借助 cProfile，调试器可以节省累积时间和每个函数调用的注释，从而提供了 Python 函数的完整详细信息。例如，在深度学习中，最常被调用的函数可能是卷积过滤器和向后传递运算符，以及 cProfile 对其中的每一个函数进行分析。对于 cProfile 选项，您可以进一步选择计时器选项：总时间、CPU 时间和 CPU 外时间。虽然您可以分析 CPU 时间内在处理器（包括 CPU 和 GPU）上执行的每个函数调用，但也可以使用 “非 CPU 时间” 选项识别 I/O 或网络瓶颈。默认值为总时间，调试器会分析 CPU 和 CPU 外时间。借助 cProfile，您可以在分析配置文件数据时向下钻取到每个功能。
  - Py仪器/py仪器是基于采样工作的低开销 Python 分析器。使用 Py仪器选项，调试器每毫秒对性能分析事件进行一次采样。由于 Py仪器测量了挂钟时间而不是 CPU 时间，因此 Py仪器选项可能是比 cProfile 选项的更好选择，用于减少性能分析噪音（过滤出累积速度不相关的函数调用）和捕获实际计算密集型的运算符（训练模型的速度累计较慢）。使用 Py仪器，您可以看到一棵函数调用树，并更好地了解缓慢的结构和根本原因。

  Note

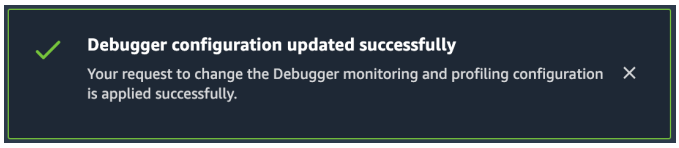
  启用 Python 分析可能会减慢整体训练时间。在 cProfile 的情况下，每次调用时都会对最频繁调用的 Python 运算符进行分析，因此性能分析的处理时间随调用次数而增加。对于 Py仪器而言，由于其取样机制，累计分析时间相对于时间而增加。

- 数据加载程序分析：指定目标步长或时间范围以分析深度学习框架数据加载器流程。调试器收集框架的每个数据加载器事件。

  Note

  数据加载器分析可以降低训练性能，同时从数据加载器那里收集信息。我们不建议您启用数据加载器分析功能超过几个步骤。
  调试器已预先配置为注释数据加载器进程仅针对Amazon深度学习容器。调试器无法在任何其他自定义或外部容器上分析数据加载器进程。

4. 选择“确认以完成您的性能分析配置。当配置成功更新后，您应该能够看到已成功更新调试器配置确认消息，如下图所示。



Amazon SageMaker 调试器见解仪表板

当您启动 SageMaker 培训任务时，调试器默认开始监控 Amazon EC2 实例的硬件系统资源利用率。您可以通过见解仪表板跟踪系统利用率、统计数据概述和瓶颈检测状态。本指南将向您介绍以下选项卡下的调试器见解仪表板的内容：概述、节点, 和模型洞察.

Note

Studio 调试器见解仪表板在`ml.m5.4xlarge`实例来处理和渲染可视化效果。每个调试器见解选项卡运行一个 Studio 内核会话。在单个实例上运行多个调试器见解选项卡的多个内核会话。关闭 “调试器见解” 选项卡时，相应的内核会话也将关闭。Studio 应用程序仍然处于活动状态，并会计`ml.m5.4xlarge`实例使用情况。有关定价的信息，请参阅Amazon SageMaker 定价页.

Important

使用调试器见解仪表板完成后，请关闭`ml.m5.4xlarge`实例以避免累积费用。有关如何关闭实例的说明，请参阅关闭 Amazon SageMaker 调试器见解实例 (p. 1893).

Important

报告中提供了地块和建议供参考，但不是确定性的。您有责任对信息进行自己的独评估。

主题

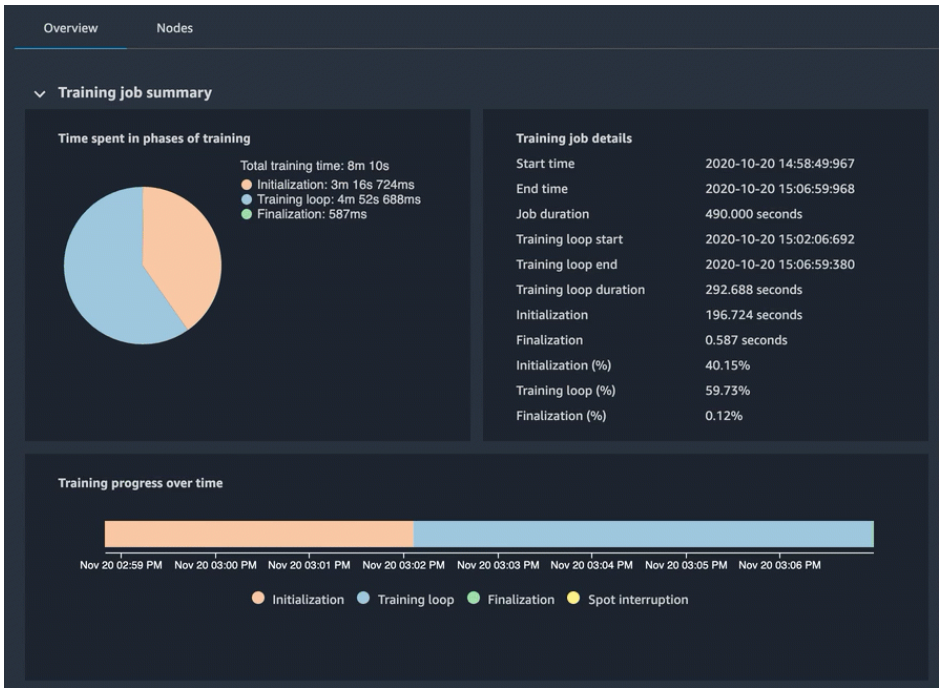
- 调试器见解 — 概述 (p. 1886)
- 调试器见解 — 节点 (p. 1889)
- 调试器见解 — 模型见解 (p. 1892)

调试器见解 — 概述

在存储库的概述选项卡中，您可以找到培训工作摘要、资源利用率摘要、资源密集型操作和见解。

训练任务摘要

这些区域有：训练任务摘要一节显示了在培训的不同阶段所花费的总体训练时间：初始化、训练循环和最终完成。饼图显示了在不同培训阶段花费的时间使用百分比和绝对时间。例如，您可以对初始化训练作业需要多长时间进行高级概述，检查初始化是否由于数据下载而花费的时间过长，从而使 GPU 处于空闲状态。

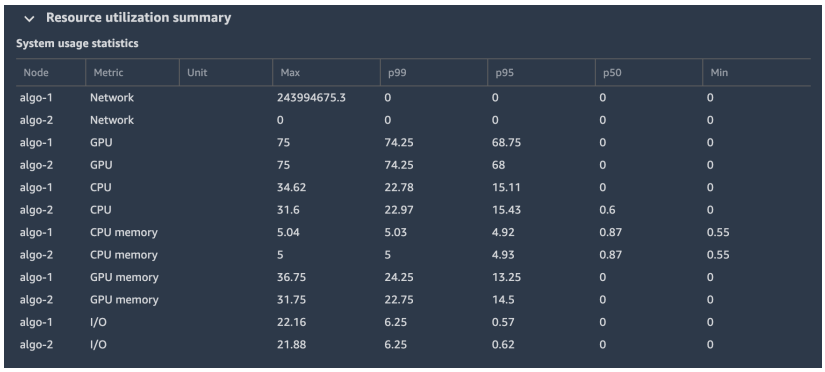


本节具有以下功能：

- 这些区域有：随着时间的推移培训图表显示了一段时间内不同培训阶段的时间表。如果你正在使用现场训练，你也可以在时间线图表中找到现货中断。
- 这些区域有：训练任务详情面板显示确切的时间戳和利用率百分比数字。
  - 开始时间：培训工作开始的确切时间。
  - 结束时间：培训工作完成的确切时间。
  - Job 时间：总训练时间从开始时间到结束时间.
  - 训练循环开始：第一个时代的第一步开始的确切时间。
  - 训练循环结束：最后一个时代的最后一步完成的确切时间。
  - 训练循环：训练循环开始时间和训练循环结束时间之间的总时间。
  - 初始化：花在初始化培训任务上的时间，例如编译培训脚本、启动 Amazon EC2 实例和下载培训数据。
  - FINAL：花在完成培训任务上的时间，例如完成模型训练、更新模型工件和关闭 Amazon EC2 实例。
  - 初始化 (%)：花费的时间百分比初始化总数超过Job 时间.
  - 训练循环 (%)：花费的时间百分比训练循环总数超过Job 时间.
  - 最后定稿 (%)：花费的时间百分比FINAL总数超过Job 时间.

资源使用率摘要

此摘要表显示了所有工作线程的硬件系统资源利用率统计信息 (algo-n)。系统指标包括 CPU 总使用率、GPU 总使用率、CPU 内存总利用率、GPU 内存总利用率、总 I/O 等待时间以及总网络（以字节为单位）。该表显示了最小值和最大值，以及 p99、p90 和 p50 百分位数。



Resource utilization summary

System usage statistics

Node	Metric	Unit	Max	p99	p95	p50	Min
algo-1	Network		243994675.3	0	0	0	0
algo-2	Network		0	0	0	0	0
algo-1	GPU		75	74.25	68.75	0	0
algo-2	GPU		75	74.25	68	0	0
algo-1	CPU		34.62	22.78	15.11	0	0
algo-2	CPU		31.6	22.97	15.43	0.6	0
algo-1	CPU memory		5.04	5.03	4.92	0.87	0.55
algo-2	CPU memory		5	5	4.93	0.87	0.55
algo-1	GPU memory		36.75	24.25	13.25	0	0
algo-2	GPU memory		31.75	22.75	14.5	0	0
algo-1	I/O		22.16	6.25	0.57	0	0
algo-2	I/O		21.88	6.25	0.62	0	0

资源密集型操作

这些区域有：资源密集型操作部分提供了更详细的分析结果，显示哪些培训作业操作是计算密集型的 在下面的示例中，它显示卷积神经网络向后传递运算符是 GPU 上资源密集度最高的。

Resource intensive operations

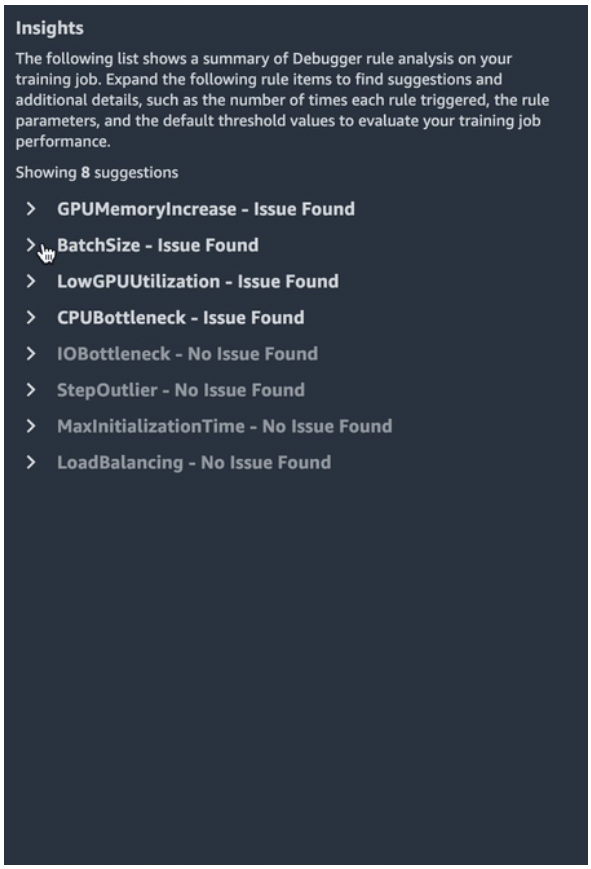
Top operations on GPU

Percentage (%)	Cumulative time	GPU operator
34.51	13905349	CudnnConvolutionBackward
34.48	13895879	cudnn_convolution_backward
6.13	2469970	conv2d
6.11	2460408	convolution
6.08	2450894	_convolution
6.03	2431585	cudnn_convolution
3.35	1351617	CudnnBatchNormBackward
3.31	1332746	cudnn_batch_norm_backward

见解

在见解窗格中，你可以找到调试器内置规则检测到的训练问题。您可以展开列表中的每个条目，以查找有用的见解、建议、规则描述以及启动规则的标准。

有关调试程序内置规则的更多信息，请参阅调试程序内置规则列表 (p. 1826).

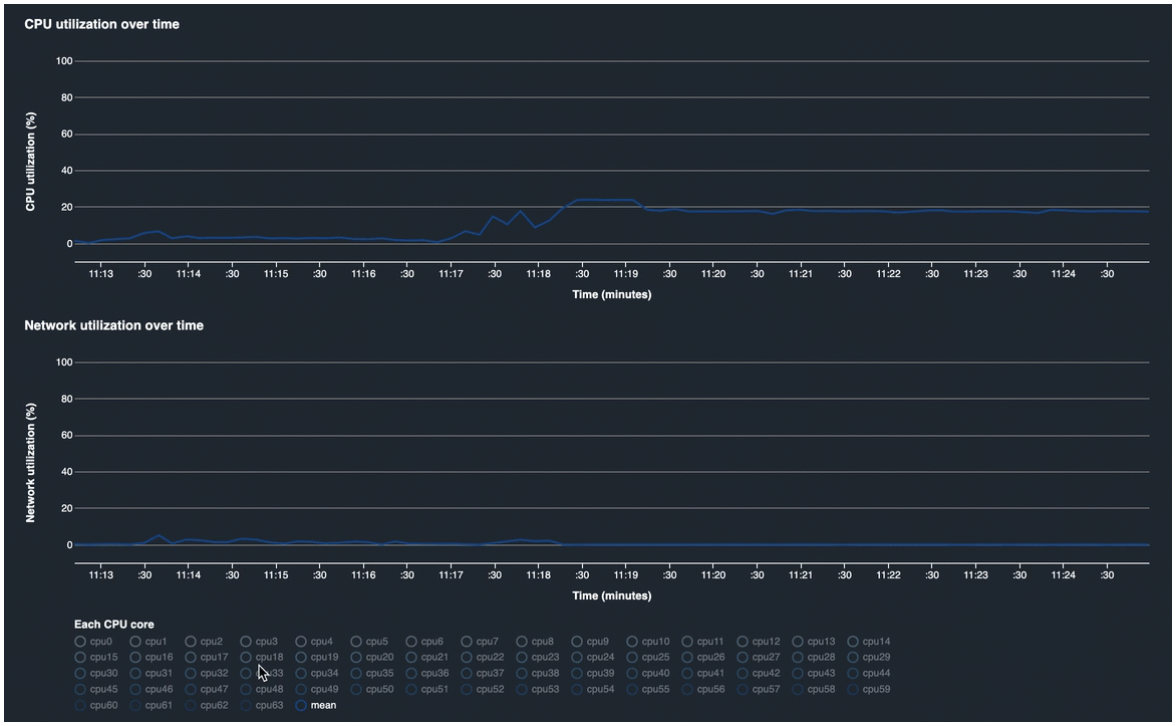


调试器见解 — 节点

在存储库的节点选项卡中，调试器提供了详细的图表，用于跟踪运行训练作业的每个计算节点。

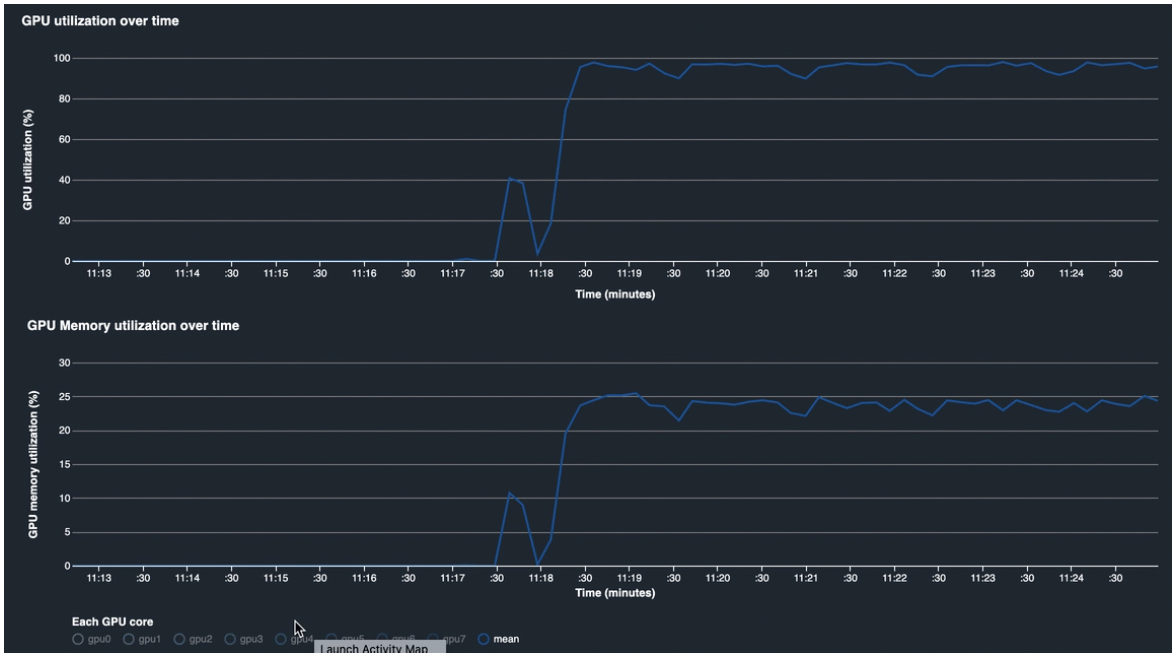
CPU 和网络使用率

前两个图表显示了一段时间内的 CPU 利用率和网络利用率。默认情况下，这些图表显示平均值：CPU 和网络利用率与 CPU 内核总数相比的平均值。您可以选择一个或多个 CPU 核心，方法是选择标签以在单个图表上绘制它们并比较核心的利用率。时间轴图是交互式的，两个图表同步。你可以拖放和缩小来仔细查看特定的时间窗口。



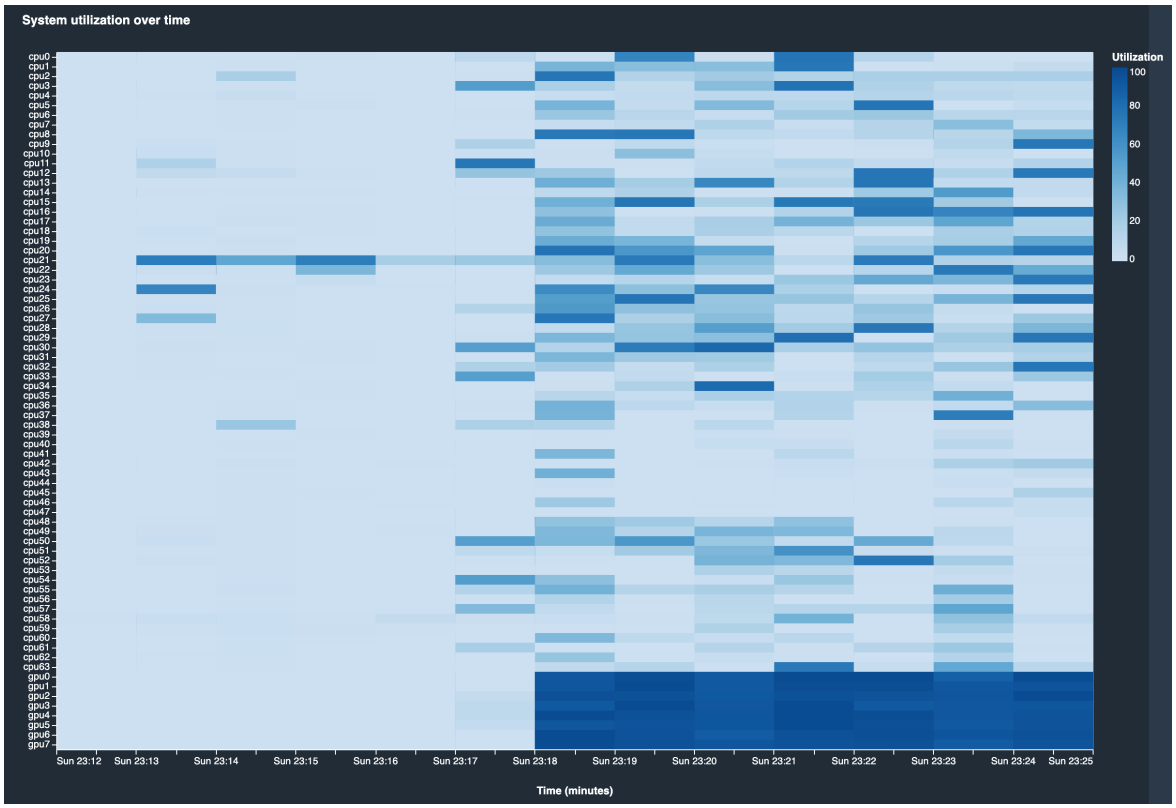
GPU 和 GPU 内存利用率

下图显示了随时间推移的 GPU 利用率和 GPU 内存利用率。默认情况下，这些图表显示了一段时间内的平均利用率。您可以选择 GPU 核心标签以查看每个核心的利用率。将利用率平均值与 GPU 内核总数相比，表示整个硬件系统资源的平均利用率。通过查看平均利用率，您可以查看 Amazon EC2 实例的整体系统资源使用情况。下图显示了在具有 8 个 GPU 内核的 ml.p3.16xlarge 实例上的示例训练作业。您可以监控培训作业是否分布良好，充分利用所有 GPU。



长期以来的整体系统利用

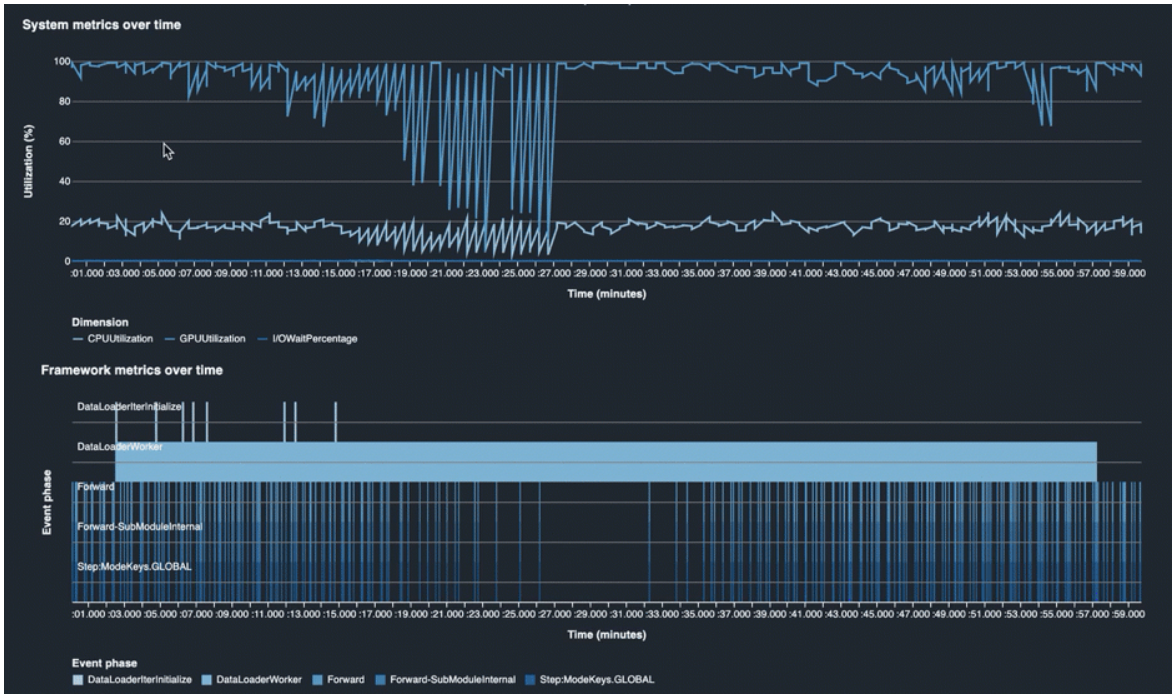
下面的热图显示了在二维图上投影的时间内的整个系统利用率。每个 CPU 和 GPU 核心都列在垂直轴中，并且随着时间的推移使用情况以颜色记录。请参阅图右侧带标记的颜色条，以了解哪个颜色级别与哪个利用率对应。例如，在下面的热图中，初始化阶段结束后，在 Sun 23:18 左右，你可以发现训练作业充分利用了 ml.p3.16xlarge 实例：GPU 内核已充分利用，CPU 适度用于处理 Python 操作。在不同的时间，CPU 中存在几个 CPU 瓶颈问题。



随着时间推移和框架事件阶段的系统资源

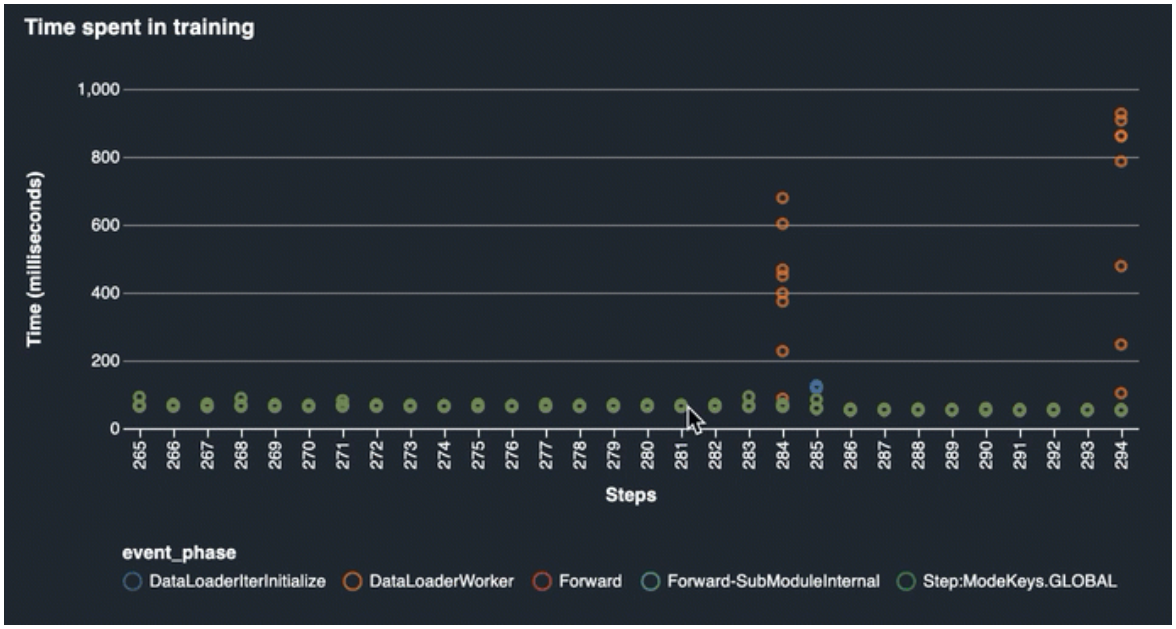
这些区域有：系统指标随时间变化图表显示了 CPU、GPU 和数据 I/O 的整体利用率。这些区域有：框架指标随时间变化图显示了框架指标，这些指标是框架事件阶段，您可以与系统指标随时间变化图。

您可以在系统资源使用时间轴中选择感兴趣的时间间隔，框架事件阶段点击该时间间隔以显示在所选时间间隔内发生的事件。在每个事件阶段块中，你可以找到训练循环实际花费的时间间隔，然后将训练循环分解为向后传和前进通过事件。总体而言，你可以看到，实际训练时间间隔在整个训练时间内只占用了很小的百分比。



在训练上花费的时间

下图显示了上次训练循环最后 30 个步骤的框架指标，以及不同事件在每个步骤中所花费的累计时间。



调试器见解 — 模型见解

在存储库的模型洞察选项卡中，您可以浏览 SageMaker XGBoost 培训作业的自动生成的培训报告。该报告提供了对 xgBoost 培训工作的见解，例如准确性随时间的推移提高、迭代损失价值、班级失衡以及用于评估模型性能的其他统计数据。根据监督学习的类型（二进制分类、多分类和回归），模型洞察选项卡会根据调试器的 xgBoost 训练 HTML 报告.

Note

此功能适用于 SageMaker xGBoost 培训职位。培训作业必须使用SageMaker xgBoost 培训容器版本 1.2-1 或更高版本。

Note

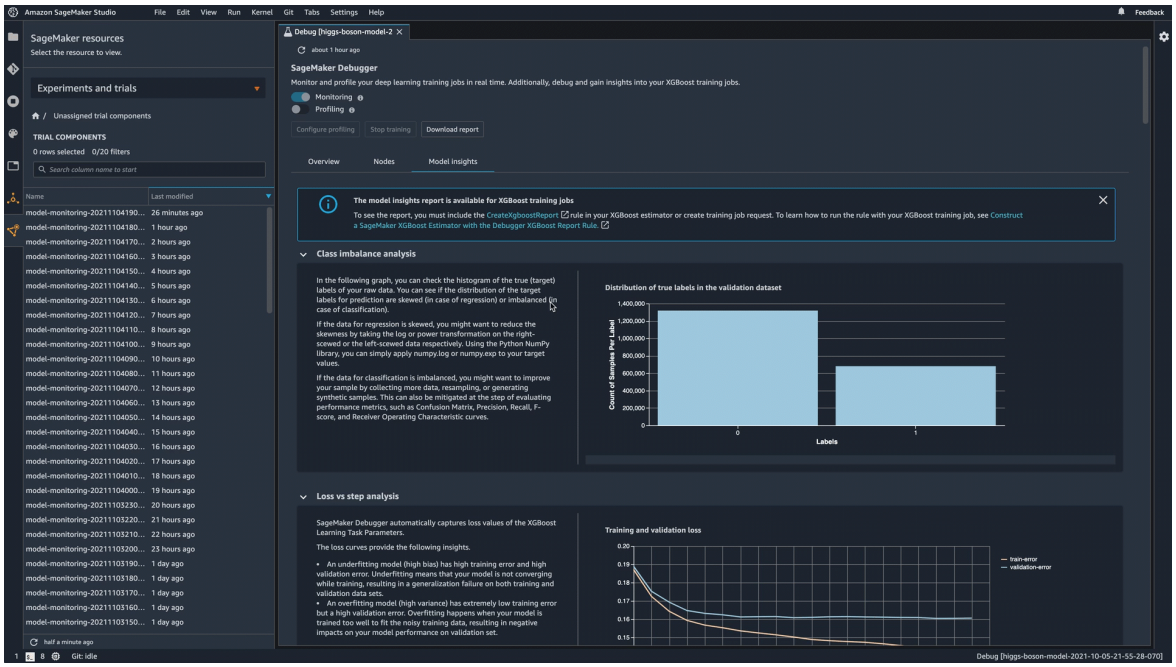
在报告中，提供地块和建议仅供参考，不是确定性的。您有责任对信息进行自己的独评估。

要加载模型见解内容，请启用CreateexGBoost 报告 (p. 1838)规则你什么时候开始培训工作。有关通过 SageMaker 估算程序启用规则的更多信息，请参阅使用调试器 xgBoost 报告规则构建 SageMaker xgBoost 估算器 (p. 1907).

有关模型见解选项卡上内容的更多信息，请参阅调试器 xgBoost 培训报告演练.

要下载 HTML 报告，请按照下载调试器 xgBoost 培训报告.

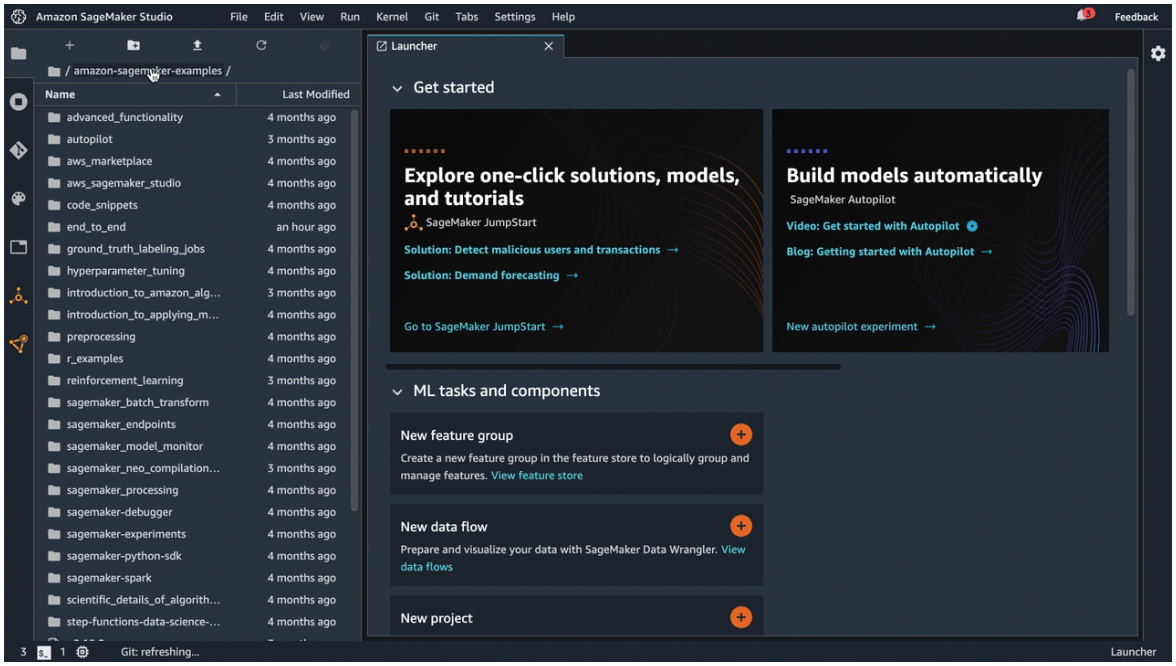
下图显示了一个示例视图模型洞察选项卡查看 SageMaker xgBoost 培训工作。



关闭 Amazon SageMaker 调试器见解实例

当您不使用调试器见解仪表板时，请关闭其运行的实例以避免产生额外费用。

在 Studio 中关闭调试程序见解实例



1. 在 Studio 中，选择运行实例和内核图标 ( )。
2. 在运行应用列出，查找ageMaker-调试程序 -1.0应用。选择关机图标 ( ) 在应用程序旁边。调试器见解仪表板在`ml.m5.4xlarge`实例。此实例也会从运行实例当你关闭ageMaker-调试程序 -1.0应用。

Studio 实验中的 Amazon SageMaker 调试器

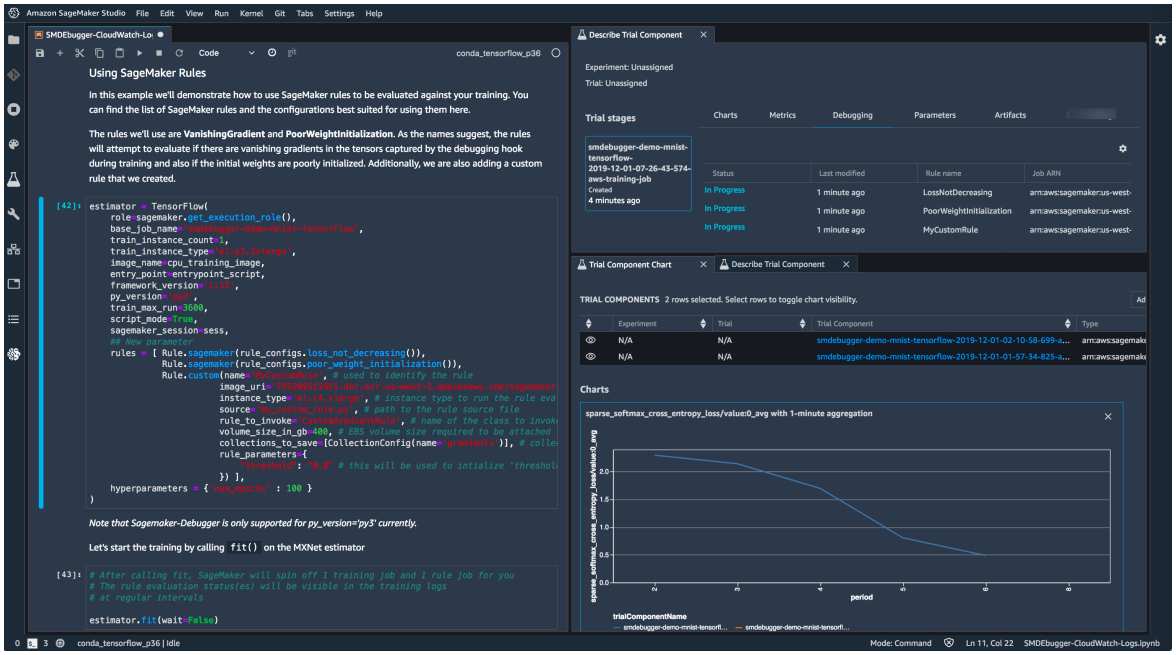
在本节中，您将了解如何在 Studio 实验中使用调试程序。您可以从实验试用列表中选择任何训练作业以查看模型输出数据图表，例如精度和损耗曲线、调试内置规则状态以及调试用于调试的调试器配置信息。

使用调试程序和 Studio 可视化张量

Studio 提供了可视化来解释调试程序捕获的张量输出。

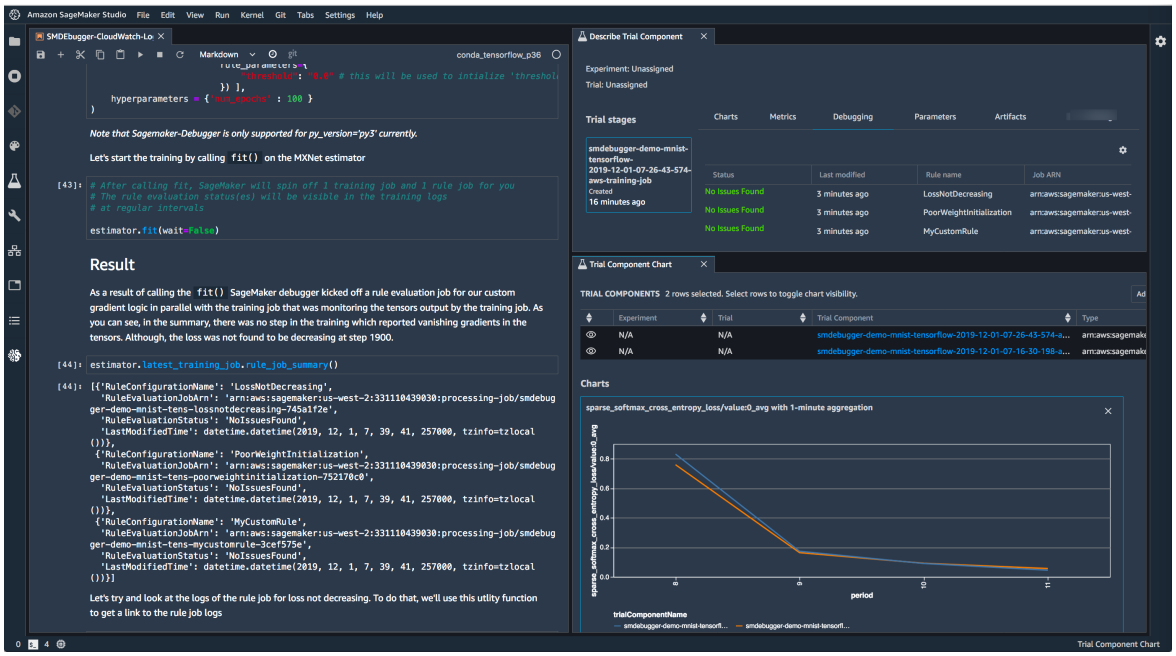
训练正在进行时的损失曲线

以下屏幕截图显示了训练损失曲线的可视化。训练正在进行中。



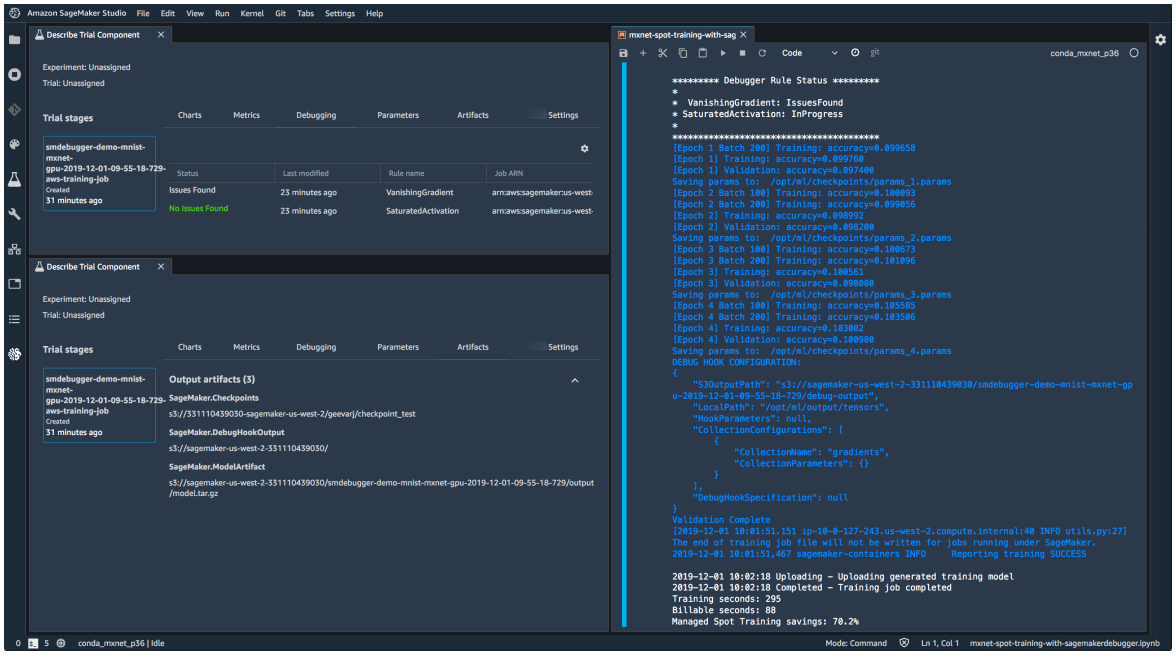
分析训练任务：比较多个作业的损失曲线

Studio 允许您在多个作业间进行比较（在此示例中，比较的是损失）。这有助于确定最出色的训练作业。



启动规则并查看作业中的日志

当规则因异常情况而被启动时，Studio 会提供失败规则的日志，以帮助您分析导致出现该异常情况的原因。



SageMaker 调试器交互式报告

接收调试器自动生成的训练和性能分析报告。调试器报告提供了有关训练工作的见解，并建议提高模型性能的建议。以下屏幕截图显示调试器分析报告的拼贴。要了解更多信息，请参阅 SageMaker 调试器性能分析报告 (p. 1897)。

> Note
>
> 您可以在训练作业运行期间或作业完成后下载调试器报告。在训练期间，调试器同时更新报告，反映当前规则的评估状态。只有在培训作业完成后，才能下载完整的调试器报告。
>
> Important
>
> 报告中提供了地块和建议供参考，但不是确定性的。您有责任对信息进行单独评估。

主题

- SageMaker 调试器性能分析报告 (p. 1897)
- SageMaker 调试器 xgBoost 培训报告 (p. 1907)

SageMaker 调试器性能分析报告

对于任何 SageMaker 培训作业，调试器剖析器报告 (p. 1828)规则调用所有监控和分析规则 (p. 1826)并将规则分析汇总到一份综合报告中。遵循本指南，使用Amazon SageMaker Python 开发工具包或 S3 控制台，然后从性能分析结果中了解可以解释什么。

> **Important**
>
> 报告中提供了地块和建议供参考，但不是确定性的。您有责任对信息进行单独评估。

主题

- 下载调试器性能分析报告 (p. 1897)
- 调试器性能分析报告演练 (p. 1900)

下载调试器性能分析报告

在训练作业运行期间或作业完成后使用Amazon SageMaker Python 开发工具包和Amazon Command Line Interface(CLI)。

> **Tip**
>
> 您还可以通过 SageMaker Studio 调试器见解仪表板只需单击一下即可下载报告，而无需进行其他脚本编写。要了解如何从 Studio 下载报告，请参阅打开 Amazon SageMaker 调试程序见解控制面板 (p. 1881).

Download using SageMaker Python SDK and Amazon CLI

1. 检查当前作业的默认 S3 输出基础 URI。

```
estimator.output_path
```

2. 检查当前作业名称。

```
estimator.latest_training_job.job_name
```

3. 调试器性能分析报告存储在<default-s3-output-base-uri>/<training-job-name>/rule-output. 按如下所示配置规则输出路径：

```
rule_output_path = estimator.output_path + estimator.latest_training_job.job_name +
 "/rule-output"
```

4. 要检查报告是否已生成，请在rule_output_path使用aws s3 ls使用--recursive选项。

```
! aws s3 ls {rule_output_path} --recursive
```

这应返回命名为自动生成的文件夹下的完整文件列表ProfilerReport-1234567890. 文件夹名称是字符串的组合：ProfilerReport以及基于 ProfilerReport 规则启动时 Unix 时间戳的唯一 10 位数标签。

```
s3://sagemaker-us-east-2-111122223333 /sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output
2020-11-28 07:26:08     452088 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-report.html
2020-11-28 07:26:07     324474 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-report.ipynb
2020-11-28 07:26:03       1122 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/BatchSize.json
2020-11-28 07:26:03      10349 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/CPUBottleneck.json
2020-11-28 07:26:03        126 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/Dataloader.json
2020-11-28 07:26:03        130 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/GPUMemoryIncrease.json
2020-11-28 07:26:03       1997 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/IOBottleneck.json
2020-11-28 07:26:03        785 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/LoadBalancing.json
2020-11-28 07:26:03        728 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/LowGPUUtilization.json
2020-11-28 07:26:03        233 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/MaxInitializationTime.json
2020-11-28 07:26:03       1585 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/OverallFrameworkMetrics.json
2020-11-28 07:26:03        575 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/OverallSystemUsage.json
2020-11-28 07:26:03       2208 sagemaker-debugger-mnist-byoc-tf2-2020-11-28-06-32-33-097/rule-output/ProfilerReport-1606545153/profiler-output/profiler-reports/StepOutlier.json
```

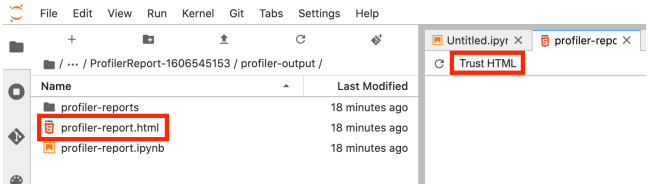
这些区域有：profiler-report.html是调试器自动生成的性能分析报告。其余文件是存储在 JSON 中的内置规则分析组件和用于将它们聚合到报告中的 Jupyter 笔记本。

5. 使用递归下载文件aws s3 cp. 以下命令将所有规则输出文件保存到ProfilerReport-1234567890当前工作目录下的文件夹。

```
! aws s3 cp {rule_output_path} ./ --recursive
```

> **Tip**
>
> 如果使用 Jupyter 笔记本服务器，请运行!pwd仔细检查当前的工作目录。

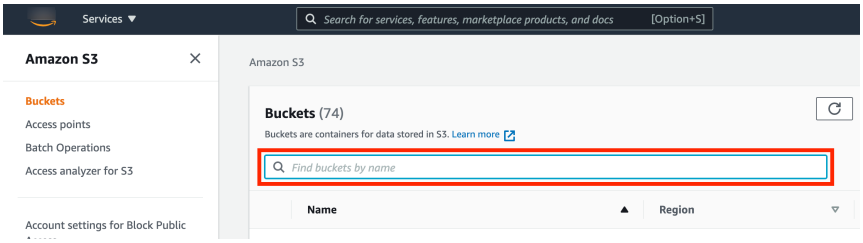
6. 在/ProfilerReport-1234567890/profiler-output目录，打开profiler-report.html. 如果使用 jupyterLab，请选择信任 HTML以查看自动生成的调试器性能分析报告。



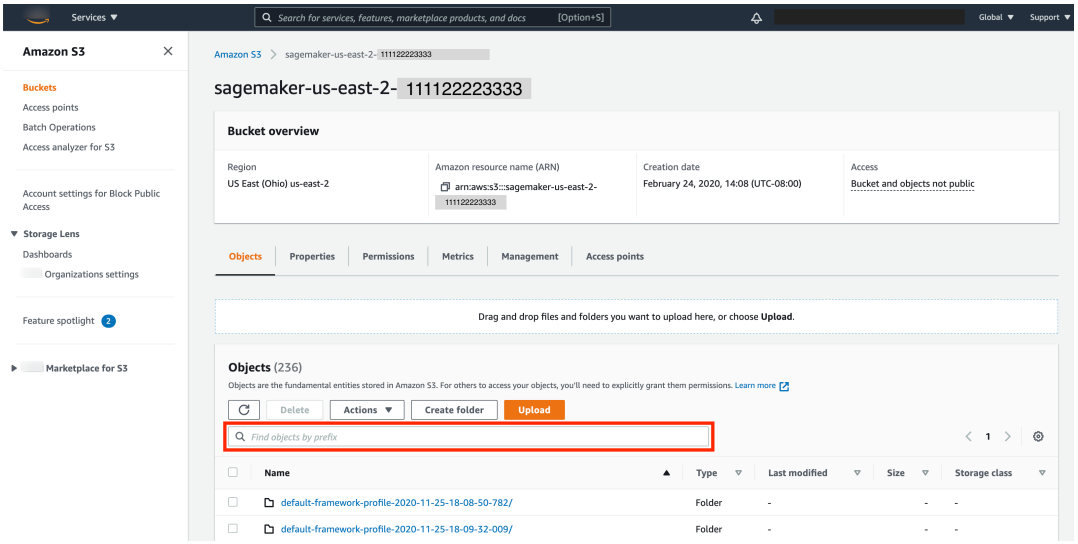
7. 打开profiler-report.ipynb文件以探索报告的生成方式。您还可以使用 Jupyter 笔记本文件自定义和扩展分析报告。

Download using Amazon S3 Console

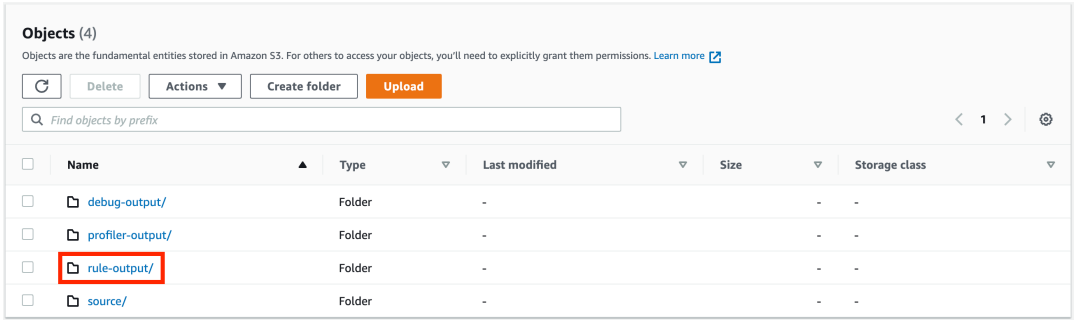
1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 搜索基本 S3 存储桶。例如，如果您尚未指定任何基本任务名称，则基本 S3 存储桶名称应采用以下格式：sagemaker-<region>-111122223333. 通过查找基本 S3 存储桶按名称查找存储桶字段中返回的子位置类型。



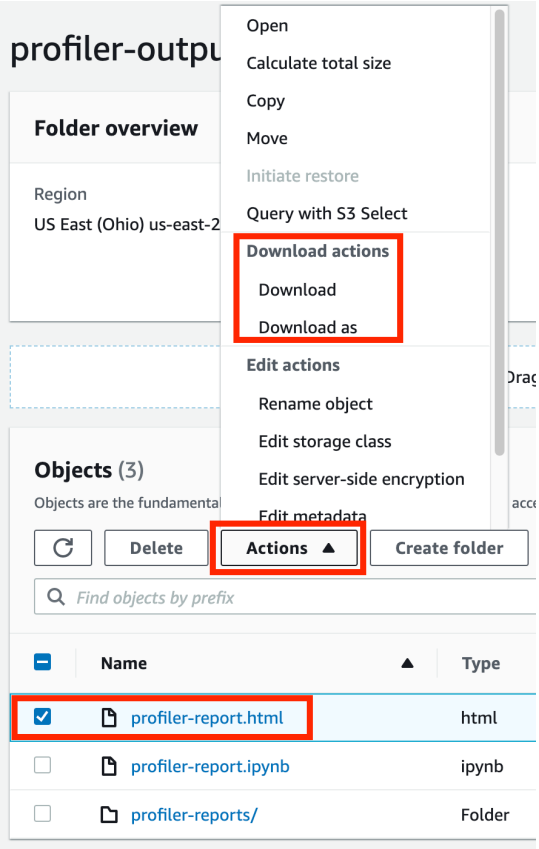
3. 在基本 S3 存储桶中，通过在按前缀查找对象输入字段。选择训练任务名称。



4. 在培训作业的 S3 存储桶中，调试器收集的训练数据必须有三个子文件夹：调试输出/、剖析器输出/, 和规则输出/. 选择规则输出/.



5. 在规则输出/文件夹，选择分析器报告 -1234567890，然后选择剖析器输出/folder。这些区域有：剖析器输出/包含文件夹profiler-report.html（html 中的自动生成的分析报告），分析器报告 .ipynb（带有用于生成报告的脚本的 Jupyter 笔记本）和分析器报告/文件夹（包含用作报告组件的规则分析 JSON 文件）。
6. 选择profiler-report.html文件，选择操作, 和下载.



7. 打开已下载的profiler-report.html在 Web 浏览器中使用文件。

Note

如果您在没有配置调试器特定参数的情况下开始训练作业，则调试器仅根据系统监控规则生成报告，因为调试器参数未配置为保存框架指标。要启用框架指标分析并接收扩展的调试器性能分析报告，请配置`profiler_config`构建或更新 SageMaker 估计器时的参数。
要了解如何配置`profiler_config`参数在开始训练作业之前，请参阅配置调试器框架分析 (p. 1801).
要更新当前的培训作业并启用框架指标分析，请参阅更新调试器框架分析配置 (p. 1804).

调试器性能分析报告演练

本节指导您逐节完成调试器分析报告。分析报告是根据监控和分析的内置规则生成的。该报告仅显示发现问题的规则的结果图。

Important

报告中提供了地块和建议供参考，但不是确定性的。您有责任对信息进行单独评估。

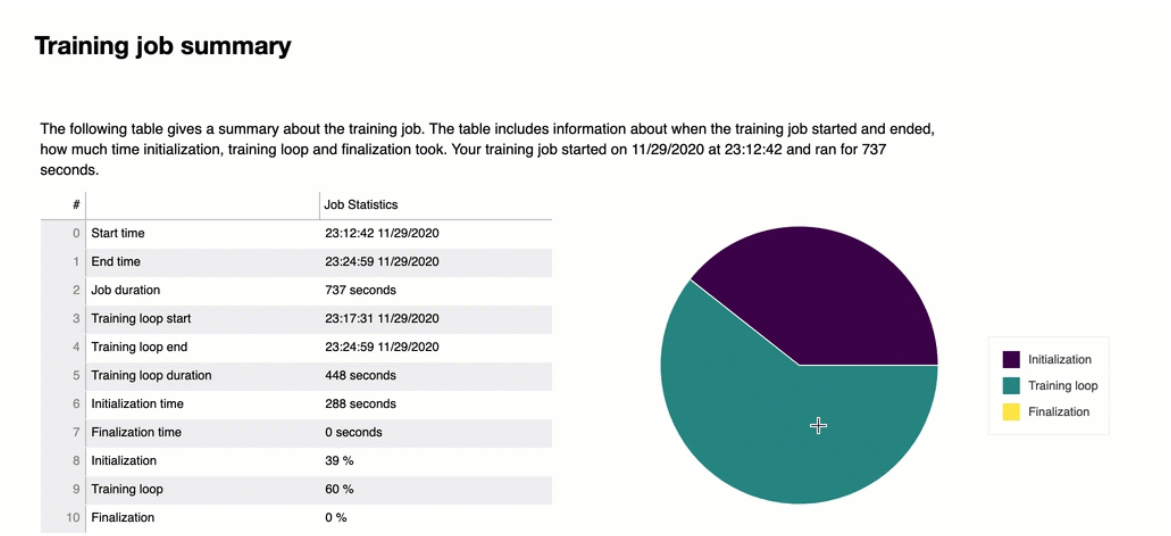
主题

- 训练 Job 摘要 (p. 1901)
- 系统计数据 (p. 1901)
- 框架指标摘要 (p. 1902)
- 规则摘要 (p. 1904)
- 分析训练循环 — 步骤持续时间 (p. 1904)

- GPU 利用率分析 (p. 1905)
- Batch 大小 (p. 1905)
- CPU 瓶颈 (p. 1906)
- I/O 瓶颈 (p. 1906)
- 多 GPU 训练中的负载均衡 (p. 1906)
- GPU 内存分析 (p. 1907)

训练 Job 摘要

在报告的开头，调试器提供了训练工作的摘要。在本节中，您可以概述不同训练阶段的时间持续时间和时间戳。



Training job summary

The following table gives a summary about the training job. The table includes information about when the training job started and ended, how much time initialization, training loop and finalization took. Your training job started on 11/29/2020 at 23:12:42 and ran for 737 seconds.

#		Job Statistics
0	Start time	23:12:42 11/29/2020
1	End time	23:24:59 11/29/2020
2	Job duration	737 seconds
3	Training loop start	23:17:31 11/29/2020
4	Training loop end	23:24:59 11/29/2020
5	Training loop duration	448 seconds
6	Initialization time	288 seconds
7	Finalization time	0 seconds
8	Initialization	39 %
9	Training loop	60 %
10	Finalization	0 %

汇总表包含以下信息：

- start_time— 培训工作开始的确切时间。
- end_time— 培训工作完成的确切时间。
- job_持续时间 _in_秒/秒— 从start_time到end_time.
- 训练_loop _start— 第一个时代的第一步开始的确切时间。
- 训练_loop _end— 最后一个时代的最后一步完成的确切时间。
- 训练_loop _ 持续时间 _in_秒— 训练循环开始时间和训练循环结束时间之间的总时间。
- 初始化 _in_秒数— 在初始化培训工作上花费的时间。初始化阶段涵盖从start_time到训练_loop _start时间。初始化时间花在编译训练脚本、启动训练脚本、创建和初始化模型、启动 EC2 实例和下载训练数据上。
- 最终化 _in_秒数— 花在完成培训任务上的时间，例如完成模型训练、更新模型工件和关闭 EC2 实例。最后定稿阶段涵盖从训练_loop _end是时候去end_time.
- 初始化 (%)— 花在上面的时间百分比初始化总数超过job_持续时间 _in_秒/秒.
- 训练循环 (%)— 花在上面的时间百分比训练循环超过总数job_持续时间 _in_秒/秒.
- 最后定稿 (%)— 花在上面的时间百分比最后确定总数超过job_持续时间 _in_秒/秒.

系统计数据

在本节中，您可以看到系统利用率统计信息的概述。

System usage statistics

The 95th quantile of the total GPU utilization on node algo-2 is 74%. GPUs on node algo-2 are well utilized

The following table shows usage statistics per worker node such as total CPU and GPU utilization, total CPU and memory footprint. The table also include total IO wait time and total sent/received bytes. The table shows min and max values as well as p99, p90 and p50 percentiles.

#	node	metric	unit	max	p99	p95	p50	min
0	algo-1	Network	bytes	218817581.57	168.02	0	0	0
10	algo-1	I/O	percentage	13.2653125	5.59283125000000	0.19559374999999	0	0
8	algo-1	GPU memory	percentage	32.25	26.25	21	0	0
2	algo-1	GPU	percentage	75	74.5	74.25	0	0
6	algo-1	CPU memory	percentage	5.05	5.01	4.98	2.17	0.55
4	algo-1	CPU	percentage	32.955625	22.629131250000	17.034	3.70249999999999	0
1	algo-2	Network	bytes	4135.24	0	0	0	0
11	algo-2	I/O	percentage	20.1875	8.1552500000000	1.74781249999999	0	0
9	algo-2	GPU memory	percentage	38	31.75	21.75	0	0
3	algo-2	GPU	percentage	75	74.5	74.25	0	0
7	algo-2	CPU memory	percentage	5.05	5.02	4.99	2.17	0.55
5	algo-2	CPU	percentage	35.004374999999	25.699968750000	18.334296875	3.77828125	0

调试器分析报告包括以下信息：

- 节点— 列出节点的名称。如果在多节点（多个 EC2 实例）上使用分布式训练，则节点名称的格式为`algo-n`.
- 公制— 调试器收集的系统指标：CPU、GPU、CPU 内存、GPU 内存、I/O 和网络指标。
- 单元— 系统指标的单位。
- max— 每个系统指标的最大值。
- p99— 每个系统利用率的 99%。
- p95— 每个系统利用率的第 95 个百分位。
- p50— 每个系统利用率的第 50 个百分位数（中位数）。
- 分钟— 每个系统指标的最小值。

框架指标摘要

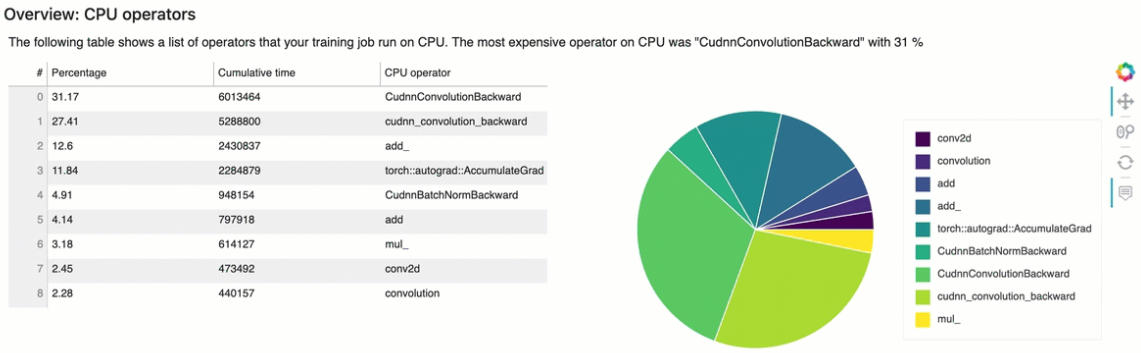
在本节中，以下饼图显示了 CPU 和 GPU 上的框架操作的细分。

每个饼图分析收集的框架指标的各个方面，如下所示：

- TRAIN/EVAL 阶段与其他阶段之间的比率— 显示在不同训练阶段花费的时间之间的比率。
- 向前和向后传递之间的比率— 显示训练循环中前向和向后传递花费的时间之间的比率。
- CPU/GPU 运营商之间的比率— 显示花在 CPU 或 GPU 上运行的操作员（例如卷积运算符）上的时间之间的比率。
- 框架中记录的一般指标— 显示花在主要框架指标（例如数据加载、向前和向后传递）上的时间之间的比率。

概述：CPU 运营商

本节详细介绍 CPU 操作员的信息。该表显示了花在最常称为 CPU 运算符的时间百分比和绝对累计时间。



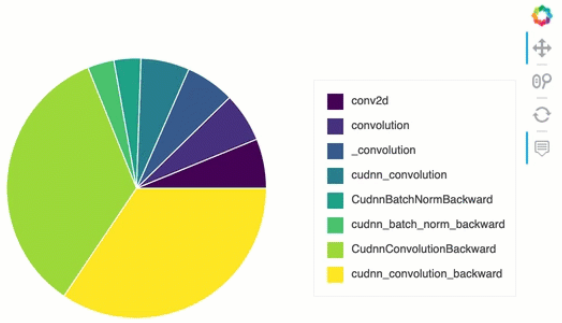
概述：GPU 运营商

本节详细介绍 GPU 运算符的信息。该表显示了花在最常称为 GPU 运算符的时间百分比和绝对累计时间。

Overview: GPU operators

The following table shows a list of operators that your training job run on GPU. The most expensive operator on GPU was "CudnnConvolutionBackward" with 34 %

#	Percentage	Cumulative time	GPU operator
0	34.46	13896596	CudnnConvolutionBackwa
1	34.44	13887210	cudnn_convolution_backwa
2	6.16	2482529	conv2d
3	6.13	2473099	convolution
4	6.11	2463505	_convolution
5	6.06	2444523	cudnn_convolution
6	3.34	1348774	CudnnBatchNormBackwar
7	3.3	1330005	cudnn_batch_norm_backw



规则摘要

在本节中，调试器汇总了所有规则评估结果、分析、规则描述和建议。

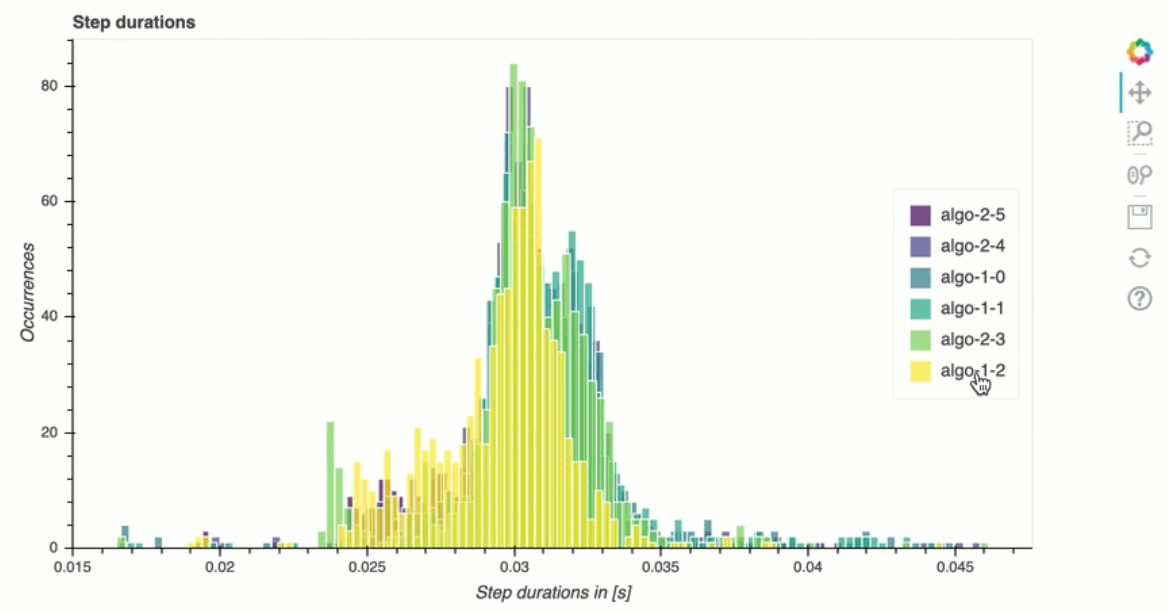
Rules summary

The following table shows a summary of the executed profiler rules. The table is sorted by the rules that triggered most frequently. In your training job this was the case for rule LoadBalancing. It has processed 5467 datapoints and triggered 263 times.

	Description	Recommendation	Number of times rule triggered	Number of datapoints	Rule parameters
LoadBalancing	Detect issues in workload balancing between multiple GPUs. Workload imbalance can for instance occur in data parallel training when gradients are accumulated on primary GPU so this GPU will be overused with regards to other GPUs limiting the effect of parallelization.	Choose different distributed training strategy or different distributed training framework	263	5467	threshold:0.2 patience:1000
LowGPUUtilization	Checks if GPU utilization is low or suffers from fluctuations. This can happen if there are bottlenecks, many blocking calls due to synchronizations or batch size too small.	Check for bottlenecks, minimize blocking calls, change distributed training strategy, increase batch-size.	244	5467	threshold_p95:70 threshold_p5:10 window:500 patience:1000
BatchSize	Checks if GPU is under-utilized because of the batch size being too small. To detect this the rule analyzes the average GPU memory footprint, CPU and GPU utilization.	Run on a smaller instance type or increase batch size	211	5466	cpu_threshold_p95:70 gpu_threshold_p95:70 gpu_memory_threshold_p95:70 patience:1000 window:500
GPUMemoryIncrease	If model and/or batch size is too large then training will run out of memory and crash.	Choose a larger instance type with more memory (if it is not a memory leak) or apply model parallelism (Rubik)	25	5467	increase:5 patience:1000 window:10
CPUBottleneck	Checks if CPU usage is high but GPU usage is low at the same time, it may indicate a CPU bottleneck where GPU is waiting for data to arrive from CPU. The rule triggers if number of CPU bottlenecks exceeds a predefined threshold.	CPU bottlenecks can happen when data preprocessing is very compute intensive. You should consider increasing the number of data-loader processes or apply pre-fetching.	18	10938	threshold:50 cpu_threshold:90 gpu_threshold:10 patience:1000
IOBottleneck	If IO wait time is high but at the same time GPU usage is low, it may indicate an IO bottleneck where GPU is waiting for data to arrive from disk. The rule triggers if number of IO bottlenecks exceeds a predefined threshold.	Pre-fetch data or choose different file formats such as binary formats which improves read performance.	0	10938	threshold:50 io_threshold:50 gpu_threshold:10 patience:1000
StepOutlier	Detect outliers in step duration. Time for forward and backward pass should be roughly the same throughout the training. If there are significant outliers it would indicate an issue due to a system stall or a bottleneck.	Check for bottlenecks	0	4803	threshold:3 mode:None n_outliers:10 stddev:3
MaxInitializationTime	Checks if the training initialization is taking too much time. The rule waits until first step is available. This can happen if you are running in File mode and a lot of data needs to be downloaded from Amazon S3.	Switch from File to Pipe mode	0	4803	threshold:20

分析训练循环 — 步骤持续时间

在本节中，您可以在每个节点中找到有关每个 GPU 核心的步长持续时间的详细统计信息。调试器评估步长持续时间的平均值、最大值、p99、p95、p50 和最小值，并计算步长异常值。下面的直方图显示了在不同工作节点和 GPU 上捕获的步长持续时间。您可以通过选择右侧的图例来启用或禁用每个工作人员的直方图。您可以检查是否有特定 GPU 导致步长持续时间异常值。



GPU 利用率分析

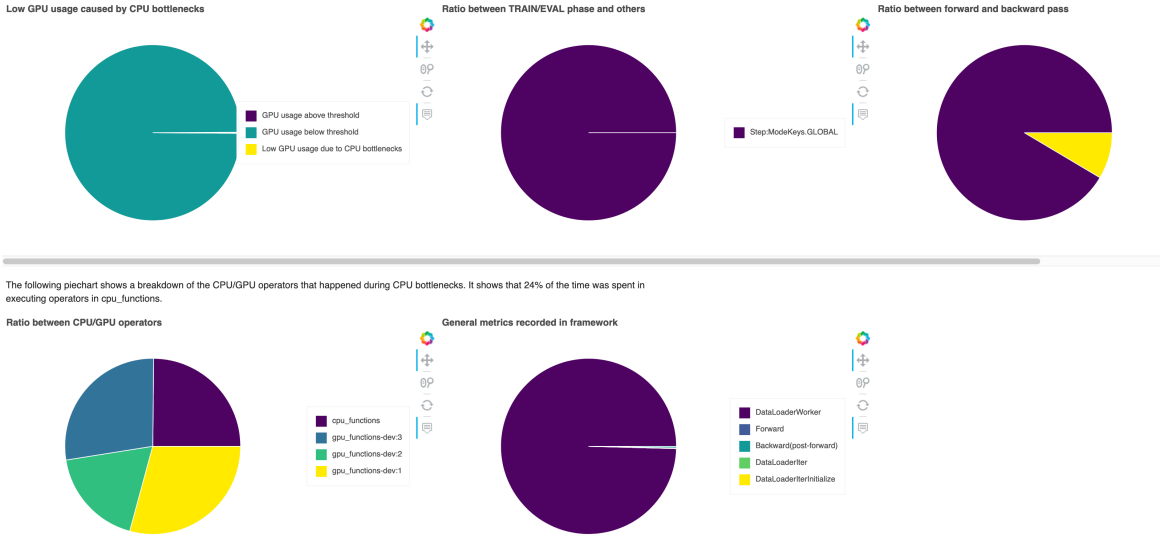
本节显示了基于 LowGPU 利用率规则的 GPU 核心利用率的详细统计信息。它还总结了 GPU 利用率统计数据，均值、p95 和 p5，以确定培训作业是否未充分利用 GPU。

Batch 大小

本节显示了 CPU 总使用率、单个 GPU 利用率和 GPU 内存占用量的详细统计信息。BatchSize 规则决定是否需要更改批处理大小以更好地利用 GPU。您可以检查批处理大小是太小导致利用不足还是过大导致过度利用和内存不足问题。在图中，方框显示中位数 (p50) 中的 p25 和 p75 百分位数范围（分别填充深紫色和亮黄色），误差条显示下限的第 5 个百分位数和上限的第 95 个百分位数。

CPU 瓶颈

在本节中，您可以深入了解 CPU瓶颈规则在培训作业中检测到的CPU 瓶颈。该规则检查 CPU 使用率是否高于`cpu_threshold`（默认情况下为 90%）以及 GPU 利用率低于`gpu_threshold`（默认情况下为 10%）。



饼图显示以下信息：

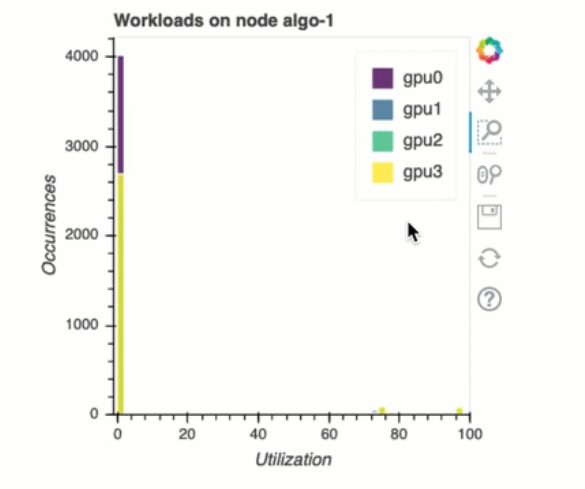
- CPU 瓶颈导致的 GPU 使用率低— 显示 GPU 使用率高于和低于阈值的数据点与符合 CPU 瓶颈标准的数据点之间的比率。
- TRAIN/EVAL 阶段与其他阶段之间的比率— 显示在不同训练阶段花费的时间之间的比率。
- 向前和向后传递之间的比率— 显示训练循环中前向和向后传递花费的时间之间的比率。
- CPU/GPU 运营商之间的比率— 显示 Python 操作员（例如数据加载器进程以及前向和向后传递运算符）在 GPU 和 CPU 上花费的时间之间的比率。
- 框架中记录的一般指标— 显示主要框架指标以及在指标上花费的时间持续时间之间的比率。

I/O 瓶颈

在本节中，您可以找到 I/O 瓶颈的摘要。该规则评估 I/O 等待时间和 GPU 利用率，并监控在 I/O 请求上花费的时间是否超过总训练时间的阈值百分比。这可能表明 GPU 正在等待数据从存储到达的 I/O 瓶颈。

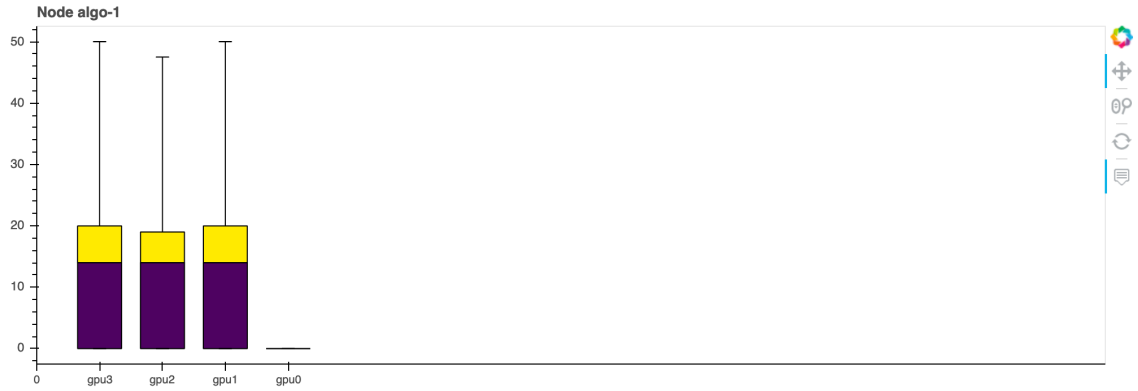
多 GPU 训练中的负载均衡

I在本节中，您可以识别 GPU 之间的工作负载平衡问题。



GPU 内存分析

在本部分中，您可以分析 GpumemoryGress 规则收集的 GPU 内存利用率。在图中，方框显示中位数 (p50) 中的 p25 和 p75 百分位数范围（分别填充深紫色和亮黄色），误差条显示下限的第 5 个百分位数和上限的第 95 个百分位数。



SageMaker 调试器 xgBoost 培训报告

对于 SageMaker xGBoost 培训作业，请使用调试器CreateexGBoost 报告 (p. 1838)规则接受关于培训进度和结果的综合培训报告。按照本指南，指定CreateexGBoost 报告 (p. 1838)在构建 xgBoost 估算器时规则，请使用Amazon SageMaker Python 开发工具包或 Amazon S3 控制台，并深入了解培训结果。

> **Important**
>
> 报告中提供了地块和建议供参考，但不是确定性的。您有责任对信息进行单独评估。

主题

- 使用调试器 xgBoost 报告规则构建 SageMaker xgBoost 估算器 (p. 1907)
- 下载调试器 xgBoost 培训报告 (p. 1908)
- 调试器 xgBoost 培训报告演练 (p. 1911)

使用调试器 xgBoost 报告规则构建 SageMaker xgBoost 估算器

这些区域有：CreateexGBoost 报告 (p. 1838)规则从你的训练作业中收集以下输出张量：

- `hyperparameters`— 在第一步中保存。
- `metrics`— 每 5 个步骤节省损失和准确性。
- `feature_importance`— 每 5 个步骤保存一次。
- `predictions`— 每 5 个步骤保存一次。
- `labels`— 每 5 个步骤保存一次。

输出张量保存在默认的 S3 存储桶中。例如：`s3://sagemaker-<region>-<12digit_account_id>/<base-job-name>/debug-output/`。

当您为 xGBoost 训练作业构建 SageMaker 估计器时，请指定如下示例代码所示的规则。

Using the SageMaker generic estimator

```
import boto3
import sagemaker
from sagemaker.estimator import Estimator
from sagemaker import image_uris
from sagemaker.debugger import Rule, rule_configs

rules=[
    Rule.sagemaker(rule_configs.create_xgboost_report())
]

region = boto3.Session().region_name
xgboost_container=sagemaker.image_uris.retrieve("xgboost", region, "1.2-1")

estimator=Estimator(
    role=sagemaker.get_execution_role()
    image_uri=xgboost_container,
    base_job_name="debugger-xgboost-report-demo",
    instance_count=1,
    instance_type="ml.m5.2xlarge",

    # Add the Debugger XGBoost report rule
    rules=rules
)

estimator.fit(wait=False)
```

下载调试器 xgBoost 培训报告

在训练作业运行期间或作业完成后使用Amazon SageMaker Python 开发工具包和Amazon Command Line Interface(CLI)。

Download using the SageMaker Python SDK and Amazon CLI

1. 检查当前作业的默认 S3 输出基础 URI。

   ```
   estimator.output_path
   ```

2. 检查当前作业名称。

   ```
   estimator.latest_training_job.job_name
   ```

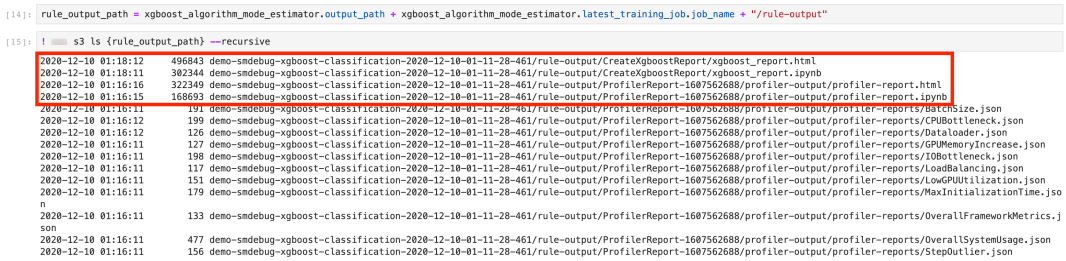
3. 调试器 xgBoost 报告存储在`<default-s3-output-base-uri>/<training-job-name>/rule-output`. 按如下所示配置规则输出路径：

```
rule_output_path = estimator.output_path + "/" +
 estimator.latest_training_job.job_name + "/rule-output"
```

4. 要检查报告是否已生成，请在`rule_output_path`使用`aws s3 ls`使用`--recursive`选项。

```
! aws s3 ls {rule_output_path} --recursive
```

这应返回自动生成的文件夹下名为的完整文件列表`CreateXgboostReport`和`ProfilerReport-1234567890`. xgBoost 培训报告存储在`CreateXgboostReport`，并将分析报告存储在`ProfilerReport-1234567890`folder。要了解有关使用 xgBoost 培训作业默认生成的性能分析报告的更多信息，请参阅SageMaker 调试器性能分析报告 (p. 1897).



这些区域有：`xgboost_report.html`是调试器自动生成的 xgBoost 训练报告。这些区域有：`xgboost_report.ipynb`是一款 Jupyter 笔记本，用于将训练结果汇总到报告中。您可以下载所有文件、浏览 HTML 报告文件以及使用笔记本修改报告。

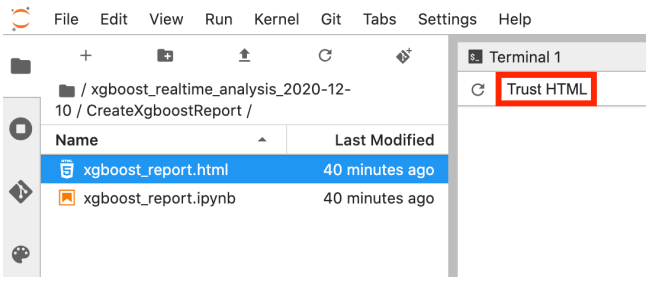
5. 使用递归下载文件`aws s3 cp`. 以下命令将所有规则输出文件保存到`ProfilerReport-1234567890`当前工作目录下的文件夹。

```
! aws s3 cp {rule_output_path} ./ --recursive
```

Tip

如果你使用的是 Jupyter 笔记本服务器，请运行`!pwd`验证当前的工作目录。

6. 在`/CreateXgboostReport`目录，打开`xgboost_report.html`. 如果你正在使用 jupyterLab，请选择信任 HTML查看自动生成的调试器训练报告。

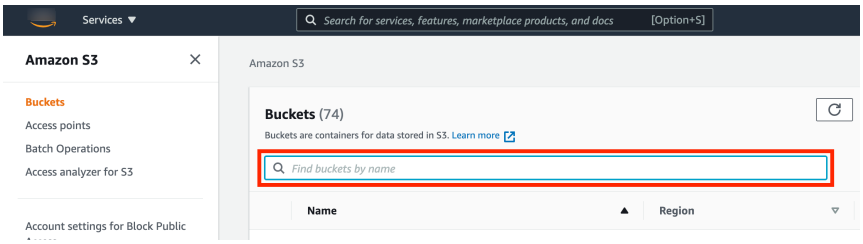


7. 打开`xgboost_report.ipynb`文件以探索报告的生成方式。您可以使用 Jupyter 笔记本文件自定义和扩展培训报告。

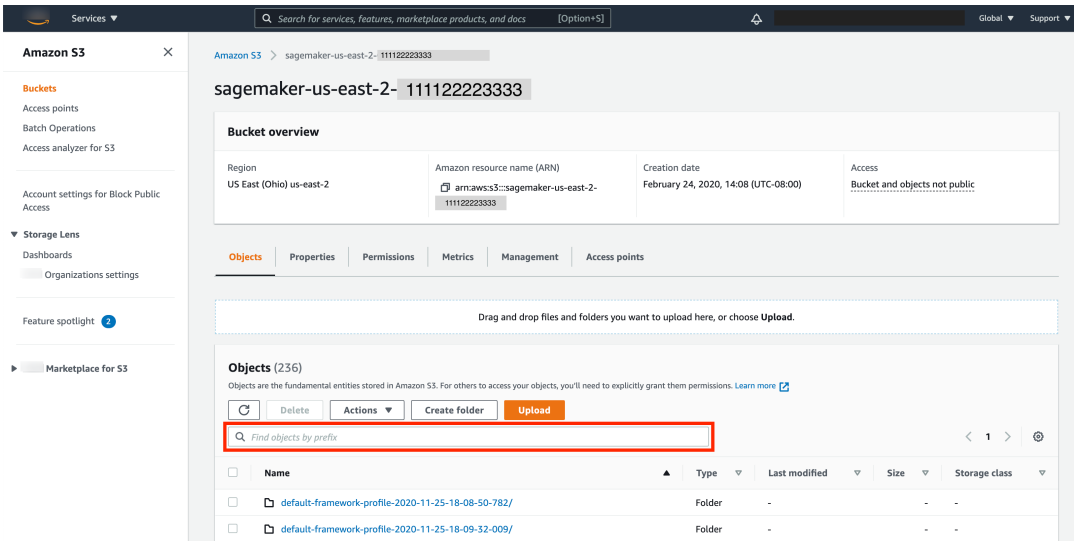
Download using the Amazon S3 console

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。

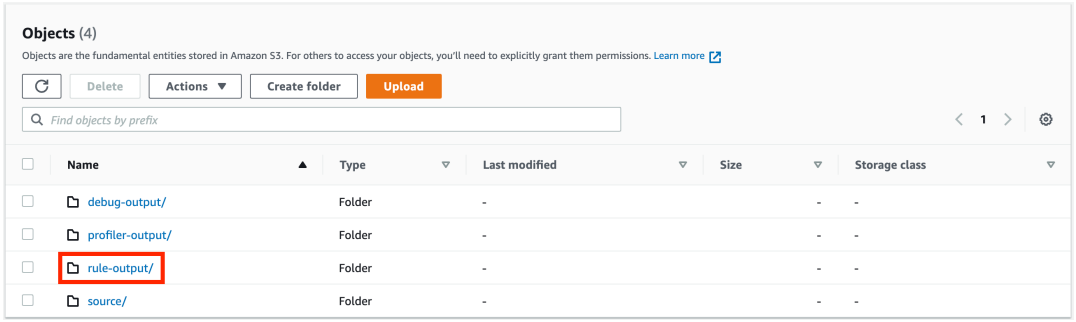
2. 搜索基本 S3 存储桶。例如，如果您尚未指定任何基本任务名称，则基本 S3 存储桶名称应采用以下格式：`sagemaker-<region>-111122223333`. 通过查找基本 S3 存储桶按名称查找存储桶字段中返回的子位置类型。



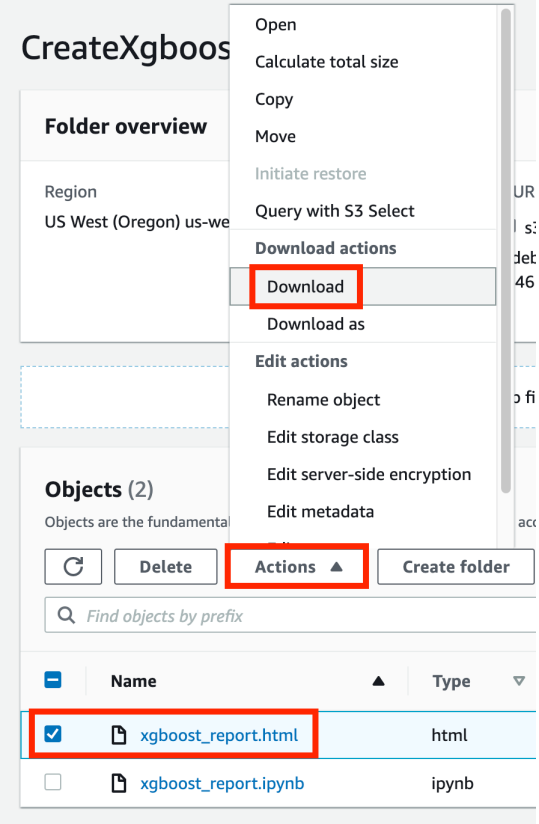
3. 在基本 S3 存储桶中，通过在中输入您的作业名称前缀来查找培训任务名称按前缀查找对象然后选择训练任务名称。



4. 在训练作业的 S3 存储桶中，选择规则输出/子文件夹。调试器收集的训练数据必须有三个子文件夹：调试输出/、剖析器输出/, 和规则输出/.



5. 在规则输出/文件夹中，选择CreateXGBoost 报告/folder。该文件夹包含xbgoost_report.html（html 中的自动生成的报告）和xbgoost_报告.ipynb（带有用于生成报告的脚本的 Jupyter 笔记本）。
6. 选择xbgoost_report.html文件，选择下载操作，然后选择下载.



7. 打开已下载的xbgoost_report.html在 Web 浏览器中使用文件。

调试器 xgBoost 培训报告演练

本节指导您完成调试器 xgBoost 培训报告。报告会根据输出张量正则表达式自动聚合，识别二进制分类、多类分类和回归中的训练作业类型。

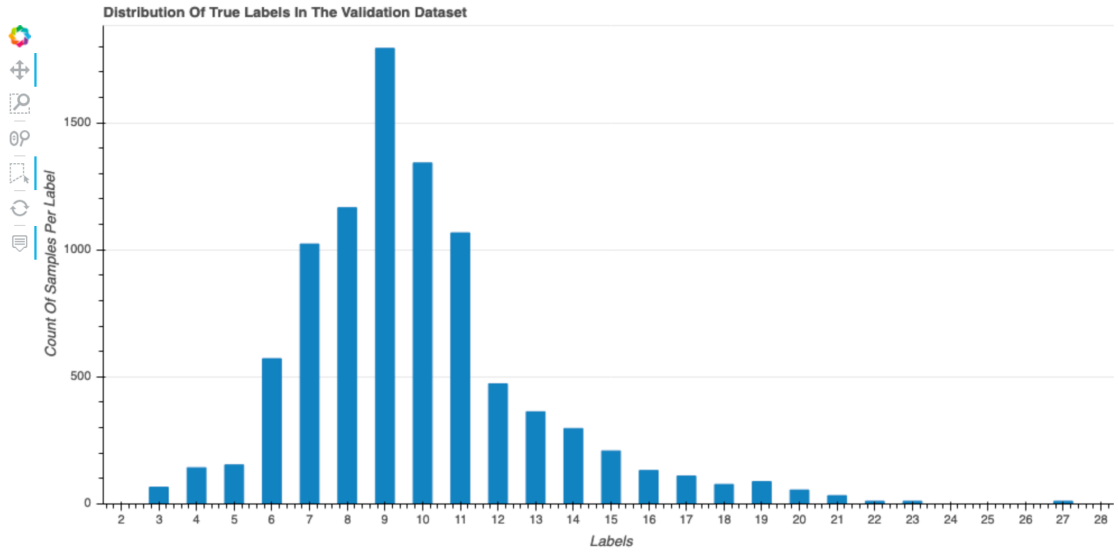
> **Important**
>
> 报告中提供了地块和建议供参考，但不是确定性的。您有责任对信息进行单独评估。

主题

- 数据集的真实标签的分布 (p. 1911)
- 亏损与步长图 (p. 1912)
- 功能重要性 (p. 1913)
- 混乱矩阵 (p. 1914)
- 混乱矩阵的评估 (p. 1915)
- 迭代中每个对角线元素的准确率 (p. 1916)
- 接收器运行特性曲线 (p. 1917)
- 最后保存步骤的残差分布 (p. 1918)
- 每个标签箱遍历迭代的绝对验证错误 (p. 1919)

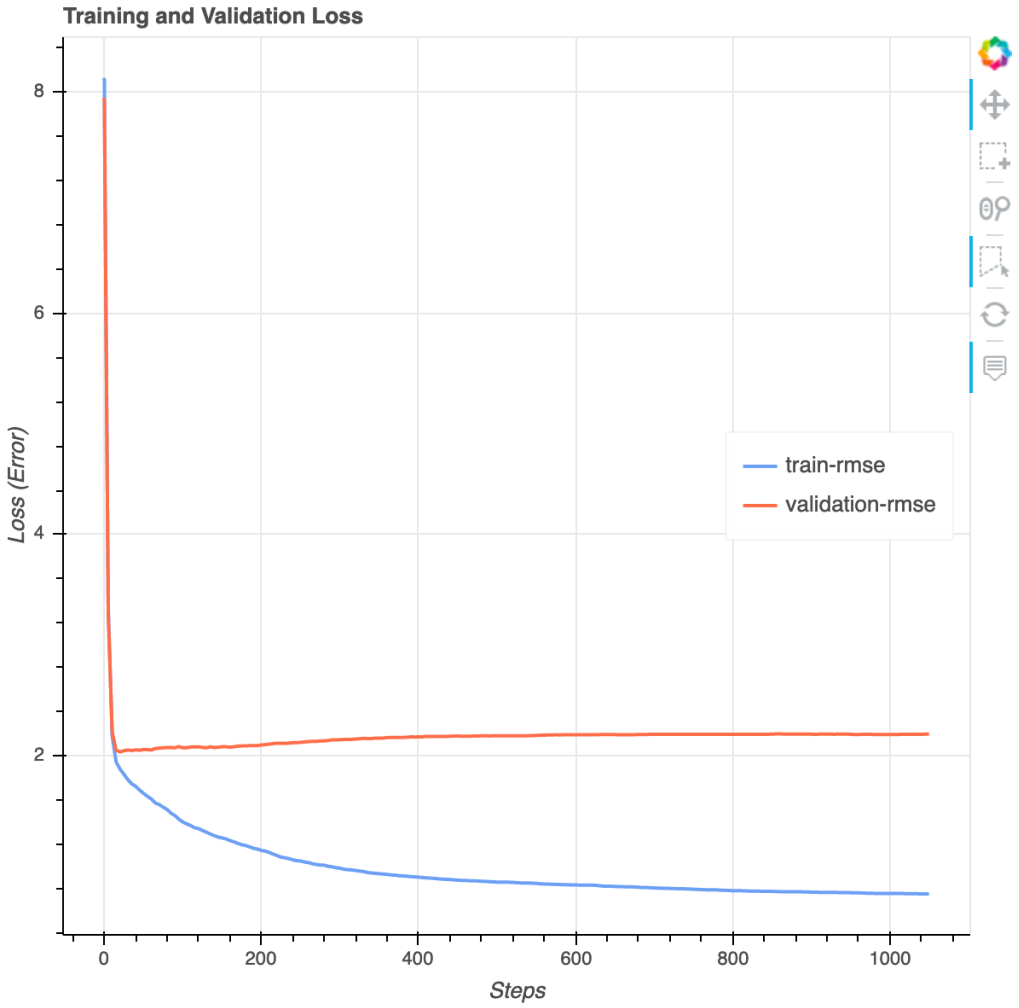
数据集的真实标签的分布

此直方图显示原始数据集中标记类（用于分类）或值（用于回归）的分布情况。数据集中的偏斜可能会导致错误。此可视化适用于以下模型类型：二进制分类、多分类和回归。



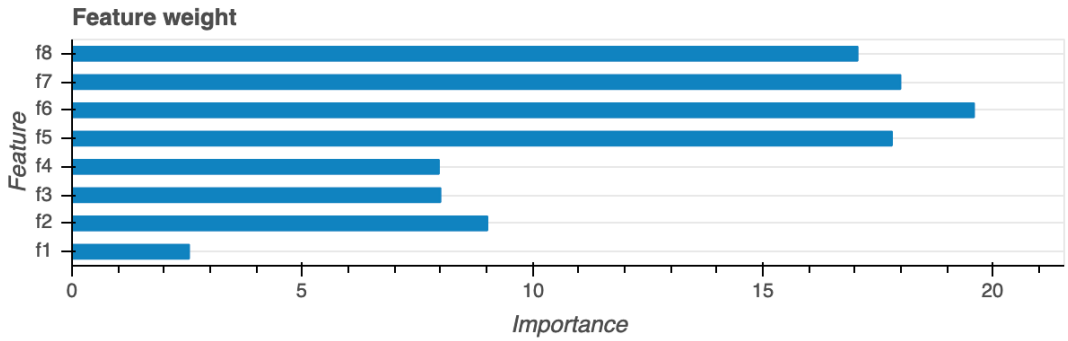
亏损与步长图

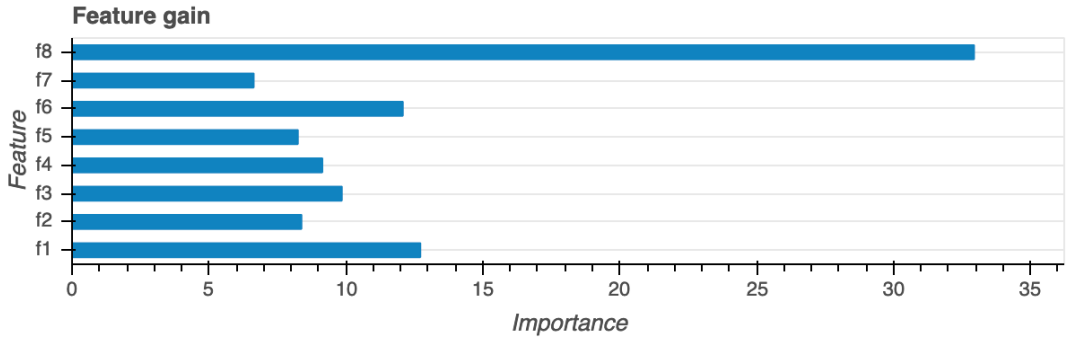
这是一个折线图，显示整个训练步骤中训练数据和验证数据丢失的进展。损失是你在目标函数中定义的，例如平方误差。你可以从这个图中衡量模型是过度拟合还是缺陷。此部分还提供见解你可以用它来确定如何解决过度合适和不足的问题。此可视化适用于以下模型类型：二进制分类、多分类和回归。

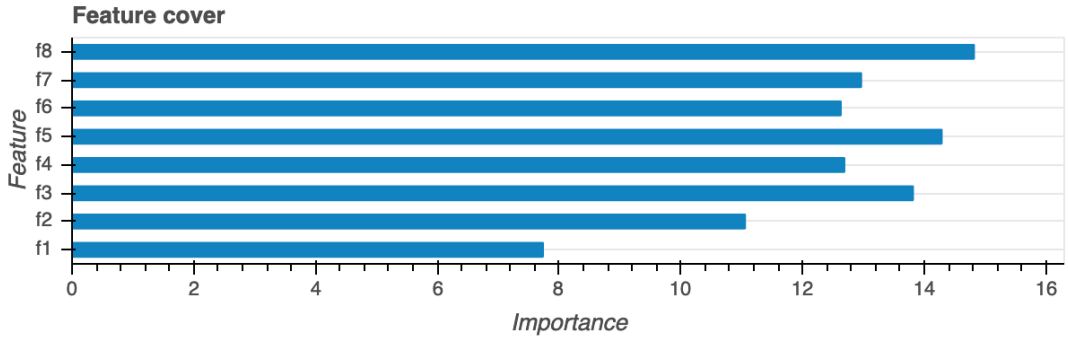


功能重要性

提供了三种不同类型的功能重要性可视化效果：体重、增益和覆盖范围。我们为报告中三者中的每一个提供了详细的定义。要素重要性可视化可帮助您了解训练数据集中的哪些要素对预测作出了贡献。功能重要性可视化可用于以下模型类型：二进制分类、多分类和回归。

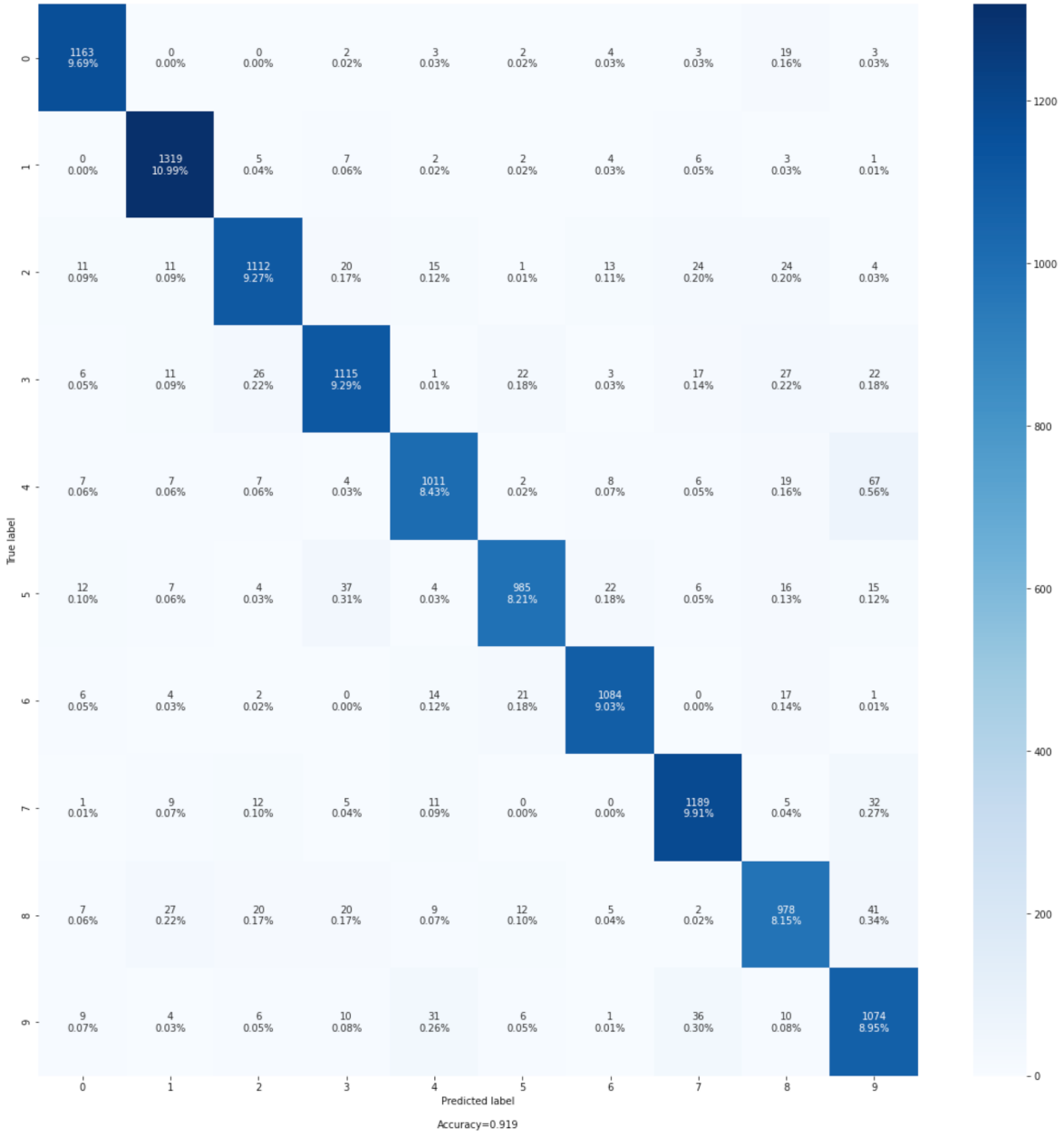






混乱矩阵

此可视化仅适用于二进制和多类分类模型。光靠准确度可能不足以评估模型性能。对于某些使用案例，例如医疗保健和欺诈检测，了解假阳性率和误报率也很重要。混淆矩阵为您提供了用于评估模型性能的额外维度。



混乱矩阵的评估

本节为您提供了有关模型精度、召回和 F1 分数的微观、宏观和加权指标的更多见解。

Overall Accuracy

```
    Overall Accuracy: 0.919
```

Micro Performance Metrics

```
    Performance metrics calculated globally by counting the total true positives, false negatives, and false positive
s.

    Micro Precision: 0.919
    Micro Recall: 0.919
    Micro F1-score: 0.919
```

Macro Performance Metrics

```
    Performance metrics calculated for each label, and find their unweighted mean.
    This does not take the class imbalance problem into account.

    Macro Precision: 0.919
    Macro Recall: 0.918
    Macro F1-score: 0.918
```

Weighted Performance Metrics

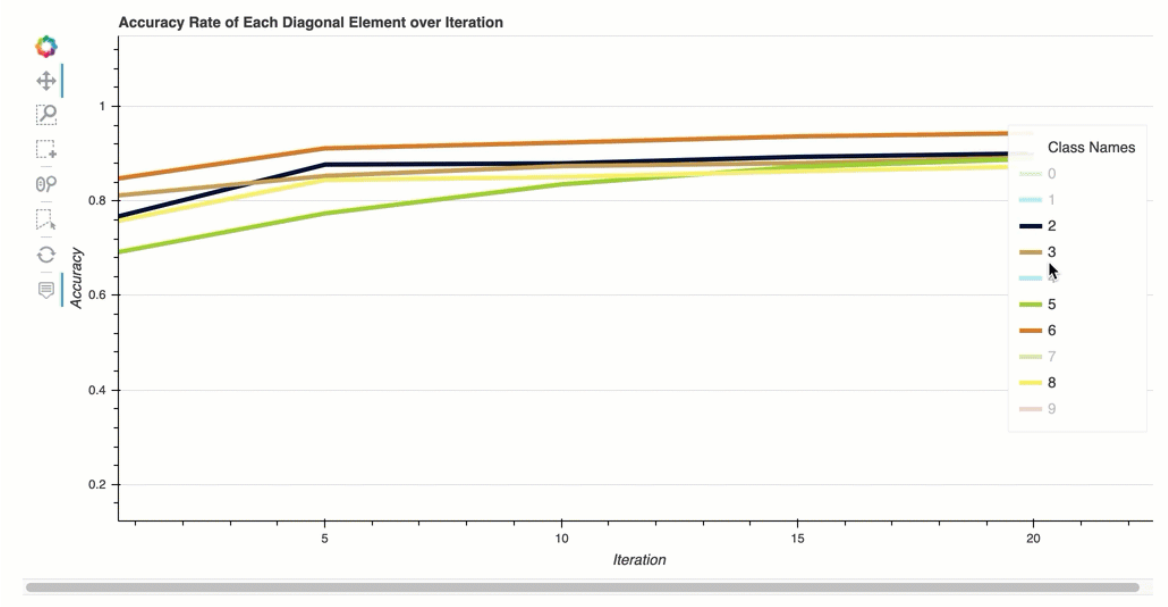
```
    Performance metrics calculated for each label and their average weighted by support
    (the number of true instances for each label).
    This extends the macro option to take the class imbalance into account.
    It might result in an F-score that is not between precision and recall.

    Weighted Precision: 0.92
    Weighted Recall: 0.919
    Weighted F1-score: 0.919
```

Classification Report

```
        The summary of the precision, recall, and F1-score for each class.

              precision    recall  f1-score   support

         0.0       0.95      0.97      0.96      1199
         1.0       0.94      0.98      0.96      1349
         2.0       0.93      0.90      0.92      1235
         3.0       0.91      0.89      0.90      1250
         4.0       0.92      0.89      0.90      1138
         5.0       0.94      0.89      0.91      1108
         6.0       0.95      0.94      0.95      1149
         7.0       0.92      0.94      0.93      1264
         8.0       0.87      0.87      0.87      1121
         9.0       0.85      0.90      0.88      1187

    accuracy                           0.92     12000
   macro avg       0.92      0.92      0.92     12000
weighted avg       0.92      0.92      0.92     12000
```

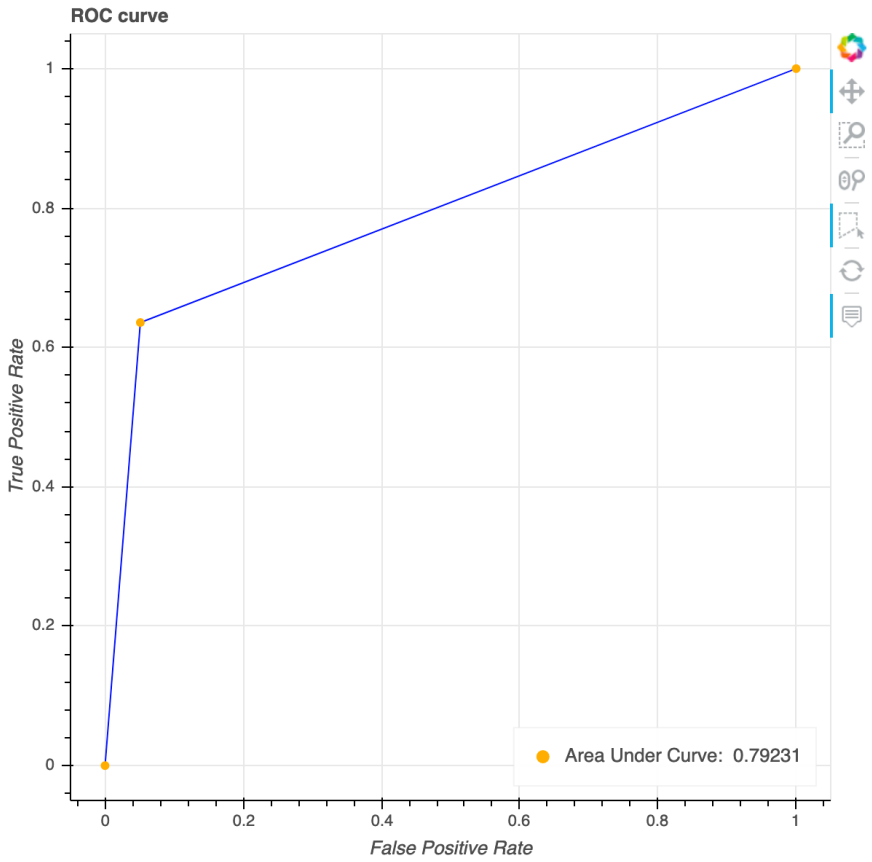
迭代中每个对角线元素的准确率

此可视化仅适用于二进制分类和多类分类模型。这是一个折线图，在每个课程的整个训练步骤中绘制混淆矩阵中的对角线值。此图显示了每个课程的准确性在整个训练步骤中的进展情况。你可以从这个情节中识别表现不佳的课程。



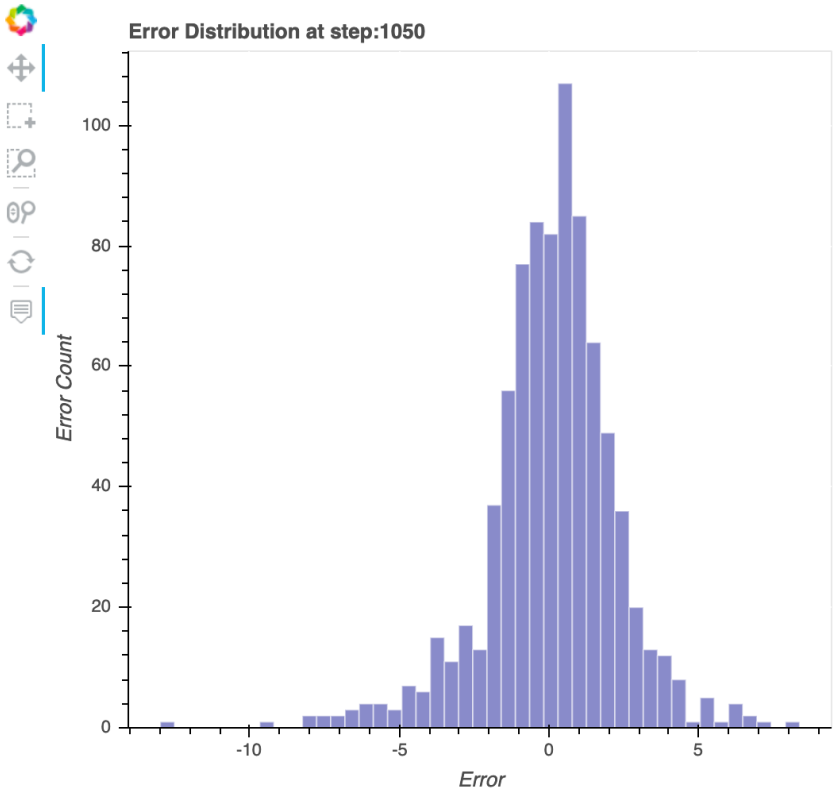
接收器运行特性曲线

此可视化仅适用于二进制分类模型。Receiver 操作特征曲线通常用于评估二进制分类模型的性能。曲线的 y 轴为真正率 (TPF)，x 轴为假阳性率 (FPR)。该图还显示曲线下面积 (AUC) 的值。AUC 值越高，分类器的预测性就越高。您还可以使用 ROC 曲线了解 TPR 和 FPR 之间的权衡，并确定用例的最佳分类阈值。可以调整分类阈值以调整模型的行为，以减少多种或另一种类型的错误 (FP/FN)。



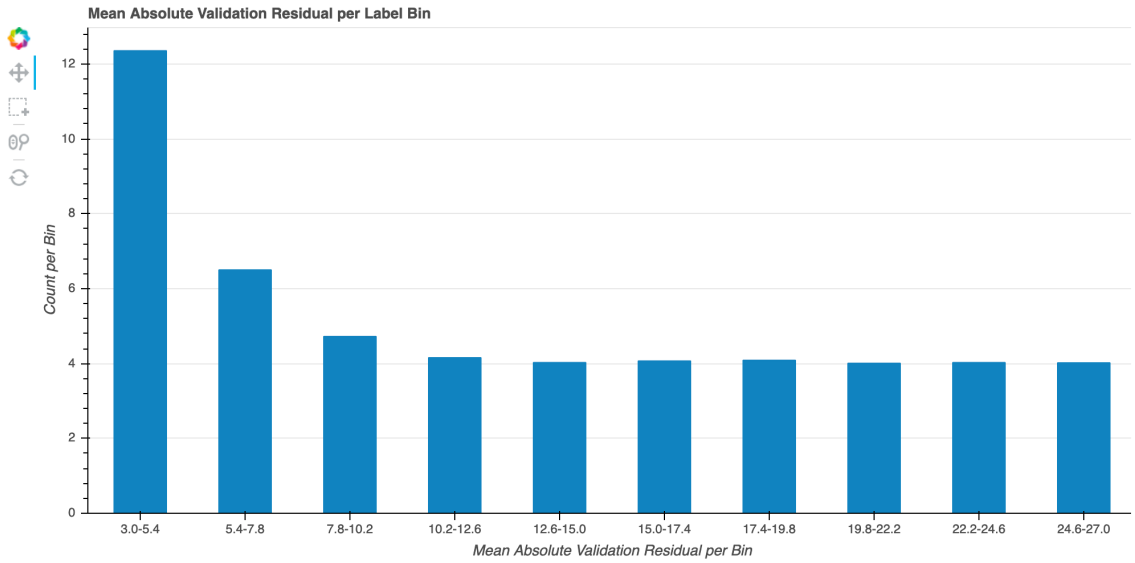
最后保存步骤的残差分布

此可视化是一个柱状图，显示调试器捕获的最后一步中的残余分布。在此可视化中，您可以检查残差分布是否接近以零为中心的正态分布。如果残差偏斜，则您的要素可能不足以预测标注。



每个标签箱遍历迭代的绝对验证错误

此可视化仅适用于回归模型。实际目标值分为 10 个间隔。此可视化显示了线图中的整个训练步骤中每个时间间隔的验证错误的进展情况。绝对验证误差是验证期间预测与实际差异的绝对值。您可以从此可视化中识别绩效不佳的时间间隔。



使用 smDebug 客户端库分析数据

在训练作业正在运行或完成训练之后，您可以使用Amazon SageMaker Python 开发工具包和smdebug 客户端库. SMDebug 库提供了分析和可视化工具，使您能够深入了解训练工作数据。

安装库并使用 SMDebug 分析工具（在 JupyterLab 笔记本电脑或 iPython 内核中）

```
! pip install -U smdebug
```

以下主题将向您介绍如何使用 SMDebug 工具可视化和分析调试器收集的训练数据。

分析系统和框架指标

- 访问监控和分析数据 (p. 1920)
- 绘制系统指标和框架指标数据 (p. 1921)
- 使用 Pandas 数据解析工具访问分析数据 (p. 1922)
- 访问 Python 分析统计数据 (p. 1922)
- 合并不同性能分析跟踪文件的时间表 (p. 1924)
- 数据加载程序 (p. 1926)

访问监控和分析数据

smDebug`TrainingJob`类从保存系统和框架指标的 S3 存储桶读取数据。

要设置**`TrainingJob`**对象并检索训练作业的分析事件文件

```
from smdebug.profiler.analysis.notebook_utils.training_job import TrainingJob
tj = TrainingJob(training_job_name, region)
```

Tip

您需要指定`training_job_name`和`region`要记录到训练任务的参数。可以通过两种方法指定培训作业信息：

- 在估算器仍附加到培训作业时，请使用 SageMaker Python SDK。

```
import sagemaker
training_job_name=estimator.latest_training_job.job_name
region=sagemaker.Session().boto_region_name
```

- 直接传递字符串。

```
training_job_name="your-training-job-name-YYYY-MM-DD-HH-MM-SS-SSS"
region="us-west-2"
```

Note

默认情况下，SageMaker Debug 会收集系统指标以监控硬件资源利用率和系统瓶颈。运行以下函数，您可能会收到有关框架指标不可用的错误消息。要检索框架分析数据并深入了解框架操作，必须启用框架分析。

- 如果你使用 SageMaker Python SDK 来操作你的训练工作请求，请通过`framework_profile_params`到`profiler_config`你的估计器的论点。要了解更多信息，请参阅配置 SageMaker 调试器框架分析.

- 如果你使用 Studio，请使用分析调试器见解仪表板中的切换按钮。要了解更多信息，请参阅SageMaker 调试器洞察面板控制器.

检索培训作业描述和保存指标数据的 S3 存储桶 URI 的描述

```
tj.describe_training_job()
tj.get_config_and_profiler_s3_output_path()
```

检查 S3 URI 中是否可以使用系统和框架指标

```
tj.wait_for_sys_profiling_data_to_be_available()
tj.wait_for_framework_profiling_data_to_be_available()
```

在指标数据可用后创建系统和框架阅读器对象

```
system_metrics_reader = tj.get_systems_metrics_reader()
framework_metrics_reader = tj.get_framework_metrics_reader()
```

刷新和检索最新的培训活动文件

读者对象有一个扩展的方法，`refresh_event_file_list()`，以检索最新的培训活动文件。

```
system_metrics_reader.refresh_event_file_list()
framework_metrics_reader.refresh_event_file_list()
```

绘制系统指标和框架指标数据

您可以将系统和算法指标对象用于以下可视化类来绘制时间线图和直方图。

> Note
>
> 要在以下可视化对象图方法中使用缩小量度来显示数据，请指定`select_dimensions`和`select_events`参数。例如，如果指定`select_dimensions=["GPU"]`，绘图方法会过滤包含 “GPU” 关键字的指标。如果你指定`select_events=["total"]`，绘图方法会过滤指标名称末尾包含 “总” 事件标签的指标。如果启用这些参数并提供关键字字符串，则可视化类将返回带有过滤量度的图表。

- 这些区域有：`MetricsHistogram`班级

```
from smdebug.profiler.analysis.notebook_utils.metrics_histogram import MetricsHistogram

metrics_histogram = MetricsHistogram(system_metrics_reader)
metrics_histogram.plot(
    starttime=0,
    endtime=system_metrics_reader.get_timestamp_of_latest_available_file(),
    select_dimensions=["CPU", "GPU", "I/O"], # optional
    select_events=["total"]                  # optional
)
```

- 这些区域有：`StepTimelineChart`班级

```
from smdebug.profiler.analysis.notebook_utils.step_timeline_chart import
 StepTimelineChart

view_step_timeline_chart = StepTimelineChart(framework_metrics_reader)
```

- 这些区域有：`StepHistogram`班级

```
from smdebug.profiler.analysis.notebook_utils.step_histogram import StepHistogram

step_histogram = StepHistogram(framework_metrics_reader)
step_histogram.plot(
    starttime=step_histogram.last_timestamp - 5 * 1000 * 1000,
    endtime=step_histogram.last_timestamp,
    show_workers=True
)
```

- 这些区域有：`TimelineCharts`班级

```
from smdebug.profiler.analysis.notebook_utils.timeline_charts import TimelineCharts

view_timeline_charts = TimelineCharts(
    system_metrics_reader,
    framework_metrics_reader,
    select_dimensions=["CPU", "GPU", "I/O"], # optional
    select_events=["total"]                  # optional
)

view_timeline_charts.plot_detailed_profiler_data([700,710])
```

- 这些区域有：`Heatmap`班级

```
from smdebug.profiler.analysis.notebook_utils.heatmap import Heatmap

view_heatmap = Heatmap(
    system_metrics_reader,
    framework_metrics_reader,
    select_dimensions=["CPU", "GPU", "I/O"], # optional
    select_events=["total"],                 # optional
    plot_height=450
)
```

使用 Pandas 数据解析工具访问分析数据

以下`PandasFrame`类提供了将收集的分析数据转换为 Pandas 数据框的工具。

```
from smdebug.profiler.analysis.utils.profiler_data_to_pandas import PandasFrame
```

这些区域有：`PandasFrame`类`tj`对象的 S3 存储桶输出路径及其方法`get_all_system_metrics()` `get_all_framework_metrics()`返回 Pandas 数据格式的系统指标和框架指标。

```
pf = PandasFrame(tj.profiler_s3_output_path)
system_metrics_df = pf.get_all_system_metrics()
framework_metrics_df = pf.get_all_framework_metrics(
    selected_framework_metrics=[
        'Step:ModeKeys.TRAIN',
        'Step:ModeKeys.GLOBAL'
    ]
)
```

访问 Python 分析统计数据

Python 分析提供了与训练脚本和 SageMaker 深度学习框架中的 Python 函数和运算符相关的框架指标。

Python 分析的训练模式和阶段

为了在训练期间分析特定的时间间隔以分区每个时间间隔的统计信息，调试器提供了设置模式和阶段的工具。

对于训练模式，请使用以下内容`PythonProfileModes`类:

```
from smdebug.profiler.python_profile_utils import PythonProfileModes
```

此类提供了以下选项：

- `PythonProfileModes.TRAIN`— 如果要分析训练阶段的目标步骤，请使用。此模式选项仅适用于 TensorFlow。
- `PythonProfileModes.EVAL`— 如果要分析评估阶段的目标步骤，请使用。此模式选项仅适用于 TensorFlow。
- `PythonProfileModes.PREDICT`— 如果您想分析预测阶段的目标步骤，请使用。此模式选项仅适用于 TensorFlow。
- `PythonProfileModes.GLOBAL`— 如果要分析全局阶段（包括前三个阶段）中的目标步骤，请使用。此模式选项仅适用于 PyTorch。
- `PythonProfileModes.PRE_STEP_ZERO`— 如果要在第一个时代的第一个训练步骤开始之前分析初始化阶段的目标步骤，请使用。此阶段包括初始作业提交、将训练脚本上传到 EC2 实例、准备 EC2 实例以及下载输入数据。此模式选项适用于 TensorFlow 和 PyTorch。
- `PythonProfileModes.POST_HOOK_CLOSE`— 如果要在训练任务完成并关闭调试器挂钩后分析最终阶段的目标步骤，请使用此选项。此阶段包括在培训作业完成和完成期间的分析数据。此模式选项适用于 TensorFlow 和 PyTorch。

对于培训阶段，请使用以下内容`StepPhase`类:

```
from smdebug.profiler.analysis.utils.python_profile_analysis_utils import StepPhase
```

此类提供了以下选项：

- `StepPhase.START`— 用于指定初始化阶段的起点。
- `StepPhase.STEP_START`— 用于指定训练阶段的开始步骤。
- `StepPhase.FORWARD_PASS_END`— 用于指定向前传递结束的步骤。该选项仅适用于 PyTorch。
- `StepPhase.STEP_END`— 用于指定训练阶段的结束步骤。该选项仅适用于 TensorFlow。
- `StepPhase.END`— 用于指定最终定稿（挂钩关闭后）阶段的结束点。如果回调钩子没有关闭，则不会进行定稿阶段分析。

Python 分析分析工具

调试器使用两个分析工具支持 Python 分析：

- cProfile — 标准 python 分析器。cProfile 为启用分析时调用的每个函数收集有关 CPU 时间的框架指标。
- Py仪器 — 这是一个低开销的 Python 分析器每毫秒对性能分析事件进行采样。

要了解有关 Python 分析选项和收集内容的更多信息，请参阅使用默认系统监控和使用不同的分析选项的自定义框架分析开始训练 Job (p. 1802).

的以下方法`PythonProfileAnalysis`、`cProfileAnalysis`、`PyinstrumentAnalysis`提供了类来获取和分析 Python 分析数据。每个函数都从默认 S3 URI 加载最新数据。

```
from smdebug.profiler.analysis.python_profile_analysis import PythonProfileAnalysis,
 cProfileAnalysis, PyinstrumentAnalysis
```

要设置 Python 分析对象以供分析，请使用 CPfileAnalysis 或 Python Analysis 类，如以下示例代码所示。它显示如何设置`cProfileAnalysis`对象，如果你想使用`PyinstrumentAnalysis`，替换类名称。

```
python_analysis = cProfileAnalysis(
    local_profile_dir=tf_python_stats_dir,
    s3_path=tj.profiler_s3_output_path
)
```

以下方法可用于`cProfileAnalysis`和`PyinstrumentAnalysis`用于获取 Python 分析统计数据的类：

- `python_analysis.fetch_python_profile_stats_by_time(start_time_since_epoch_in_secs, end_time_since_epoch_in_secs)`— 获取开始时间和结束时间，并返回开始时间或结束时间与提供的时间间隔重叠的步骤统计信息的函数统计信息。
- `python_analysis.fetch_python_profile_stats_by_step(start_step, end_step, mode, start_phase, end_phase)`— 进入起始步骤和结束步骤并返回所有分析的步骤统计信息的功能统计信息`step`满足`start_step <= step < end_step`.
  - `start_step`和`end_step`(str) — 指定开始步骤和结束步骤以获取 Python 分析统计数据。
  - `mode`(str) — 使用`PythonProfileModes`枚举器类。默认为 `PythonProfileModes.TRAIN`。提供了可用选项，请参阅Python 分析的训练模式和阶段 (p. 1923)部分。
  - `start_phase`(str) — 使用`StepPhase`枚举器类。此参数允许在训练的不同阶段之间进行分析。默认为 `StepPhase.STEP_START`。提供了可用选项，请参阅Python 分析的训练模式和阶段 (p. 1923)部分。
  - `end_phase`(str) — 使用`StepPhase`枚举器类。此参数设置训练的结束阶段。可用的选项与`start_phase`参数。默认为 `StepPhase.STEP_END`。提供了可用选项，请参阅Python 分析的训练模式和阶段 (p. 1923)部分。
- `python_analysis.fetch_profile_stats_between_modes(start_mode, end_mode)`— 从开始模式和结束模式之间的 Python 分析中获取统计信息。
- `python_analysis.fetch_pre_step_zero_profile_stats()`— 从 Python 分析中获取统计数据，直到步骤 0。
- `python_analysis.fetch_post_hook_close_profile_stats()`— 挂钩关闭后从 Python 分析中获取统计信息。
- `python_analysis.list_profile_stats()`— 返回 Python 分析统计信息的 DataFrame。每行都保存每个分析实例的元数据和相应的统计数据文件（每步一个）。
- `python_analysis.list_available_node_ids()`— 返回 Python 分析统计信息的可用节点 ID 列表。

这些区域有：`cProfileAnalysis`类特定的方法：

- `fetch_profile_stats_by_training_phase()`— 获取和聚合每种可能的开始和结束模式组合的 Python 分析统计信息。例如，如果在启用详细分析的情况下完成了培训和验证阶段，则组合为`(PRE_STEP_ZERO, TRAIN)`、`(TRAIN, TRAIN)`、`(TRAIN, EVAL)`、`(EVAL, EVAL)`, 和`(EVAL, POST_HOOK_CLOSE)`. 这些组合中每个组合中的所有统计文件都被汇总。
- `fetch_profile_stats_by_job_phase()`— 按工作阶段获取和聚合 Python 分析统计信息。任务阶段包括`initialization`（分析直到步骤 0），`training_loop`（培训和验证），以及`finalization`（挂钩关闭后进行分析）。

合并不同性能分析跟踪文件的时间表

SMDebug 客户端库提供了分析分析和可视化工具，用于合并调试器收集的系统指标、框架指标和 Python 分析数据的时间表。

Tip

在继续之前，您需要设置一个 TrainJob 对象，该对象将在本页面的整个示例中使用。有关设置 TrainJob 对象的更多信息，请参阅访问监控和分析数据 (p. 1920).

这些区域有：`MergedTimeline`类提供了在单个时间轴中集成和关联不同的分析信息的工具。调试器从训练作业的不同阶段捕获分析数据和注释之后，默认情况下保存跟踪事件的 JSON 文件`tracefolder`目录。

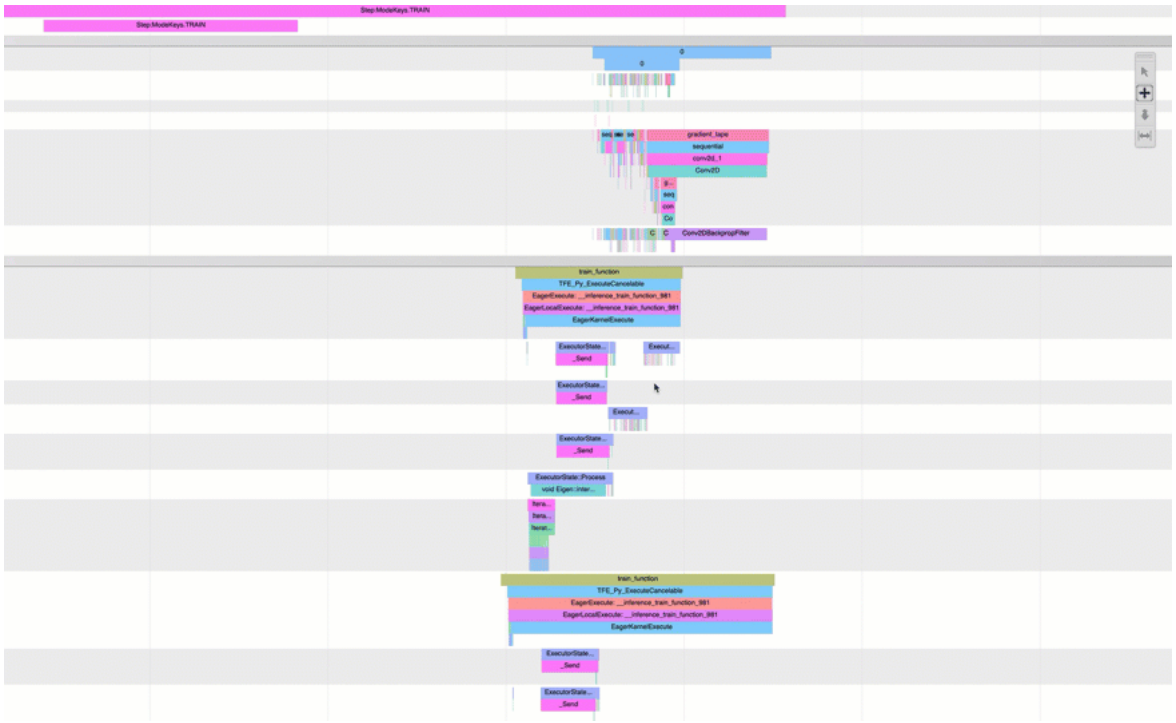
- 对于 Python 图层中的注释，跟踪文件保存在`*pythontimeline.json`.
- 对于 TensorFlow C++ 图层中的注释，跟踪文件保存在`*model_timeline.json`.
- Tensorflow 分析器将事件保存在`*trace.json.gz`文件。

Tip

如果要列出所有 JSON 跟踪文件，请使用以下命令Amazon CLI命令：

```
! aws s3 ls {tj.profiler_s3_output_path} --recursive | grep '\.json$'
```

如以下动画屏幕截图所示，将从不同分析源捕获的跟踪事件放在一个图中并对齐可以概述在训练作业的不同阶段发生的整个事件。



Tip

要使用键盘在跟踪应用程序上与合并的时间轴进行交互，请使用`W`放大的关键，`A`向左移动的关键，`S`缩小的关键，以及`D`向右移动的关键。

可以使用以下方法将多个事件跟踪 JSON 文件合并到一个跟踪事件 JSON 文件中：`MergedTimeline`来自的 API 操作和类方法`smdebug.profiler.analysis.utils.merge_timelines`模块。

```
from smdebug.profiler.analysis.utils.merge_timelines import MergedTimeline

combined_timeline = MergedTimeline(path, file_suffix_filter, output_directory)
```

```
combined_timeline.merge_timeline(start, end, unit)
```

这些区域有：MergedTimelineAPI 操作传递以下参数：

- path(str) — 指定根文件夹 (/profiler-output)，其中包含系统和框架分析跟踪文件。您可以在profiler-output使用 SageMaker 估算器类方法或 TrainEngine Job 对象。例如，estimator.latest_job_profiler_artifacts_path() 或 tj.profiler_s3_output_path。
- file_suffix_filter(list) — 指定用于合并时间轴的文件后缀筛选器列表。可用的 suffiex 过滤器包括["model_timeline.json", "pythontimeline.json", "trace.json.gz"].如果未手动指定此参数，则默认情况下将合并所有跟踪文件。
- output_directory(str) — 指定用于保存合并的时间轴 JSON 文件的路径。默认设置为为path参数。

这些区域有：merge_timeline()classmethod 传递以下参数来执行合并过程：

- start(int) — 指定开始时间（以微秒为单位和 Unix 时间格式）或合并时间轴的起始步骤。
- end(int) — 指定结束时间（以微秒为单位和 Unix 时间格式）或结束步骤以合并时间轴。
- unit(str) — 选择"time"和"step". 默认为 "time"。

使用以下示例代码，执行merge_timeline()方法然后下载合并的 JSON 文件。

- 将时间轴与"time"单位选项。以下示例代码合并了 Unix 开始时间（绝对零 Unix 时间）和当前 Unix 时间之间的所有可用跟踪文件，这意味着您可以合并整个训练期间的时间表。

```
import time
from smdebug.profiler.analysis.utils.merge_timelines import MergedTimeline
from smdebug.profiler.profiler_constants import CONVERT_TO_MICROSECS

combined_timeline = MergedTimeline(tj.profiler_s3_output_path, output_directory="./")
combined_timeline.merge_timeline(0, int(time.time() * CONVERT_TO_MICROSECS))
```

- 将时间轴与"step"单位选项。以下示例代码合并了步骤 3 和步骤 9 之间的所有可用时间表。

```
from smdebug.profiler.analysis.utils.merge_timelines import MergedTimeline

combined_timeline = MergedTimeline(tj.profiler_s3_output_path, output_directory="./")
combined_timeline.merge_timeline(3, 9, unit="step")
```

在打开 Chrome 追踪应用程序chrome://tracing在 Chrome 浏览器上，然后打开 JSON 文件。您可以浏览输出来绘制合并的时间轴。

数据加载程序

在 PyTorch 中，数据加载器迭代器，例如SingleProcessingDataLoaderIter和MultiProcessingDataLoaderIter，在对数据集的每次迭代开始时启动。在初始化阶段，PyTorch 根据配置的工作器数量启用工作进程，建立数据队列以获取数据，以及pin_memory线程。

要使用 PyTorch 数据加载器分析分析工具，请导入以下内容PT_dataloader_analysis类:

```
from smdebug.profiler.analysis.utils.pytorch_dataloader_analysis import
 PT_dataloader_analysis
```

将检索到的分析数据作为 Pandas 帧数据对象传递到使用 Pandas 数据解析工具访问分析数据 (p. 1922)部分:

```
pt_analysis = PT_dataloader_analysis(pf)
```

以下功能可用于pt_analysis对象:

smDebugS3SystemMetricsReader类从指定的 S3 存储桶读取系统指标到s3_trial_path参数。

- pt_analysis.analyze_dataloaderIter_initialization()

  分析输出这些初始化的中位数和最大持续时间。如果存在异常值（即持续时间大于 2 * 中位数），函数会打印这些持续时间的开始和结束时间。这些可用于在这段时间间隔内检查系统指标。

  下面的列表显示了此类方法可用的分析：

  - 初始化了哪种类型的数据加载器迭代器。
  - 每个迭代器的工作人员数。
  - 检查迭代器是否使用 pin_memory 进行初始化。
  - 训练期间迭代器初始化的次数。
- pt_analysis.analyze_dataloaderWorkers()

  下面的列表显示了此类方法可用的分析：

  - 在整个培训期间分散的工作人员进程的数量。
  - 工作进程的中位数和最长持续时间。
  - 异常值的工作进程的开始时间和结束时间。
- pt_analysis.analyze_dataloader_getnext()

  下面的列表显示了此类方法可用的分析：

  - 培训期间拨打的 getNext 电话数。
  - getNext 通话的中值和最长持续时间（以微秒为单位）。
  - 异常值 getNext 呼叫持续时间的开始时间、结束时间、持续时间和工作人员 ID。
- pt_analysis.analyze_batchtime(start_timestamp, end_timestamp, select_events=[".*"], select_dimensions=[".*"])

  调试程序收集所有 getNext 调用的开始时间和结束时间。您可以找到训练脚本在一批数据上花费的时间。在指定的时间范围内，您可以识别不直接参与培训的呼叫。这些调用可以来自以下操作：计算准确性、为调试或记录目的增加损失以及打印调试信息。这样的操作可能是计算密集型或耗时的。我们可以通过关联 Python 分析器、系统指标和框架指标来识别此类操作。

  下面的列表显示了此类方法可用的分析：

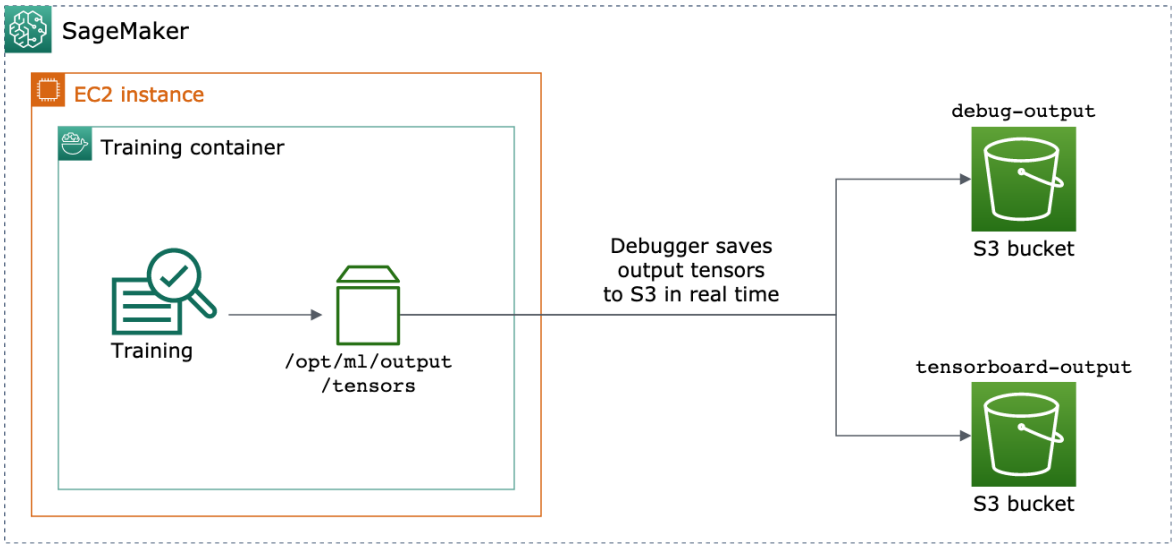
  - 分析每个数据批处理花费的时间，BatchTime_in_seconds，通过找出当前和后续 getNext 调用的开始时间之间的差异。
  - 在查找异常值BatchTime_in_seconds以及这些异常值的开始时间和结束时间。
  - 在这些期间获取系统和框架指标BatchTime_in_seconds时间戳。这表明了时间花在哪里。
- pt_analysis.plot_the_window()

  在开始时间戳和结束时间戳之间绘制时间轴图表。

在 TensorBoard 中可视化 Amazon SageMaker 调试器输出张量

使用 SageMaker 调试器创建与 TensorBoard 兼容的输出张量文件。加载文件以在 TensorBoard 中可视化，然后分析您的 SageMaker 培训作业。调试器会自动生成与 TensorBoard 兼容的输出张量文件。对于为保存

输出张量而自定义的任何挂钩配置，调试器可以灵活地创建可以导入到 TensorBoard 的标量摘要、分布和直方图。



你可以通过传递`DebuggerHookConfig`和`TensorBoardOutputConfig`对象`estimator`.

以下过程解释了如何将标量、权重和偏差保存为可以使用 TensorBoard 可视化的完整张量、直方图和分布。调试器将它们保存到训练容器的本地路径（默认路径为`/opt/ml/output/tensors`）并同步到通过调试器输出配置对象传递的 Amazon S3 位置。

使用调试器保存 TensorBoard 兼容的输出张量文件

1. 设置`tensorboard_output_config`使用调试器保存 TensorBoard 输出的配置对象`TensorBoardOutputConfig`类。对于`s3_output_path`参数中，指定当前 SageMaker 会话的默认 S3 存储桶或首选 S3 存储桶。该示例不添加`container_local_output_path`参数；相反，它被设置为默认的本地路径`/opt/ml/output/tensors`.

```
import sagemaker
from sagemaker.debugger import TensorBoardOutputConfig

bucket = sagemaker.Session().default_bucket()
tensorboard_output_config = TensorBoardOutputConfig(
    s3_output_path='s3://{}'.format(bucket)
)
```

   有关更多信息，请参阅调试程序。`TensorBoardOutputConfig`中的 APIAmazon SageMaker Python 开发工具包.

2. 配置调试器挂接并自定义挂接参数值。例如，以下代码配置一个调试器挂接，以便在训练阶段中每 100 个步骤和验证阶段中的 10 个步骤保存所有标量输出，`weights`每 500 个步骤参数（默认值）`save_interval`用于保存张量集合的值为 500），`bias`参数每 10 个全局步长，直到全局步长达到 500 个。

```
from sagemaker.debugger import CollectionConfig, DebuggerHookConfig

hook_config = DebuggerHookConfig(
    hook_parameters={
        "train.save_interval": "100",
        "eval.save_interval": "10"
    },
    collection_configs=[
        CollectionConfig("weights"),
```

```
        CollectionConfig(
            name="biases",
            parameters={
                "save_interval": "10",
                "end_step": "500",
                "save_histogram": "True"
            }
        ),
    ]
)
```

有关调试程序配置 API 的更多信息，请参阅调试程序。`CollectionConfig`和`DebuggerHookConfig`中的 APIAmazon SageMaker Python 开发工具包.

3. 使用调试器参数传递配置对象来构建 SageMaker 估计器。以下示例模板说明了如何创建通用 SageMaker 估计程序。你可以更换`estimator`和`Estimator`与其他 SageMaker 框架的估计器父类和估计器类一起使用。可用于此功能的 SageMaker 框架估算器包括`TensorFlow`、`PyTorch`, 和`MXNet`.

```
from sagemaker.estimator import Estimator

estimator = Estimator(
    ...
    # Debugger parameters
    debugger_hook_config=hook_config,
    tensorboard_output_config=tensorboard_output_config
)
estimator.fit()
```

这些区域有：`estimator.fit()`方法开始训练任务，调试器将输出张量文件实时写入调试器 S3 输出路径和 TensorBoard S3 输出路径。要检索输出路径，请使用以下估计器方法：

- 对于调试器 S3 输出路径，请使用`estimator.latest_job_debugger_artifacts_path()`.
- 对于 TensorBoard S3 输出路径，请使用`estimator.latest_job_tensorboard_artifacts_path()`.

4. 培训完成后，检查保存的输出张量的名称：

```
from smdebug.trials import create_trial
trial = create_trial(estimator.latest_job_debugger_artifacts_path())
trial.tensor_names()
```

5. 查看 Amazon S3 中的 TensorBoard 输出数据：

```
tensorboard_output_path=estimator.latest_job_tensorboard_artifacts_path()
print(tensorboard_output_path)
!aws s3 ls {tensorboard_output_path}/
```

6. 将 TensorBoard 输出数据下载到您的笔记本实例。例如，以下内容：Amazon CLI命令将 TensorBoard 文件下载到`/logs/fit`在您的笔记本实例的当前工作目录下。

```
!aws s3 cp --recursive {tensorboard_output_path} ./logs/fit
```

7. 将文件目录压缩为 TAR 文件以下载到本地计算机。

```
!tar -cf logs.tar logs
```

8. 将 Tensorboard TAR 文件下载并解压到设备上的目录中，启动 Jupyter 笔记本服务器，打开新笔记本电脑，然后运行 TensorBoard 应用程序。

```
!tar -xf logs.tar
```

```
%load_ext tensorboard
%tensorboard --logdir logs/fit
```

Amazon SageMaker 调试器的最佳实践

使用调试器运行训练作业时，请使用以下指导方针。

主题

- 选择 Machine Learning 框架 (p. 1930)
- 使用 Studio 调试器见解仪表板 (p. 1930)
- 下载调试器报告并获取更多见解 (p. 1930)
- 从训练 Job 中捕获数据并将数据保存到 Amazon S3 (p. 1930)
- 使用一组调试器内置规则分析数据 (p. 1931)
- 根据内置规则状态采取操作 (p. 1931)
- 使用 smDebug 客户端库深入了解数据 (p. 1931)
- 监控和分析训练 Job 指标 (p. 1931)
- 监控系统利用率和检测瓶颈 (p. 1931)
- 分析框架操作 (p. 1932)
- 调试模型输出张量 (p. 1932)

选择 Machine Learning 框架

您可以选择机器学习框架并使用 SageMaker 预构建的培训容器或自己的容器。使用调试器检测训练和性能问题，并分析 SageMaker 中训练作业的训练进度。SageMaker 为您提供了使用为多种机器学习框架环境准备的预构建容器的选项，以便在 Amazon EC2 上训练模型。任何训练作业都可以调整为运行AmazonDeep Learning Containers、SageMaker 培训容器和自定义容器。要了解更多信息，请参阅使用亚马逊配置调试器SageMakerPython 开发工具包 (p. 1796)和将调试器用于自定义训练容器 (p. 1868)。

使用 Studio 调试器见解仪表板

借助 Studio Debug 见解仪表板，您可以控制自己的培训工作。使用 Studio 调试器仪表板可以控制和优化 Amazon EC2 实例上的模型性能。对于在 Amazon EC2 实例上运行的任何 SageMaker 培训作业，调试器会监控资源利用率和基本模型输出数据（丢失和准确度值）。通过 Studio Debug 仪表板，深入了解您的训练工作并提高模型训练绩效。要了解更多信息，请参阅 Amazon SageMaker Studio 中的 Amazon SageMaker 调试器 (p. 1881)。

下载调试器报告并获取更多见解

您可以在调试器报告中查看聚合结果并获得见解。调试器将从内置规则分析中收集的训练和性能分析结果汇总到每个训练作业的报告中。您可以通过调试器报告找到有关训练结果的更多详细信息。要了解更多信息，请参阅 SageMaker 调试器交互式报告 (p. 1896)。

从训练 Job 中捕获数据并将数据保存到 Amazon S3

你可以使用调试器钩子来保存输出张量。在选择适合训练脚本的容器和框架后，使用调试器挂钩配置要保存的张量以及将其保存到的目录，例如 Amazon S3 存储桶。调试器挂钩可以帮助您构建配置并将其保留在您的账户中，以便在以后的分析中使用，在这些分析中，该配置可以安全地用于对隐私最敏感的应用程序。要了解更多信息，请参阅 配置调试程序挂钩以保存张量 (p. 1804)。

使用一组调试器内置规则分析数据

您可以使用调试器内置规则并行检查张量和训练作业。为了分析训练绩效数据，调试器提供了监视异常训练过程行为的内置规则。例如，在训练过程发生系统瓶颈问题或训练问题时，调试器规则将检测到这些问题，例如梯度消失、张量爆炸、过度拟合或过度训练。如有必要，您还可以通过使用自己的标准创建规则定义来定义培训问题来构建自定义规则。要了解有关调试器规则的更多信息，请参阅配置调试程序内置规则 (p. 1810)了解有关使用Amazon SageMaker Python 开发工具包. 有关调试器内置规则的完整列表，请参阅调试程序内置规则列表 (p. 1826). 如果您要创建自定义规则，请参阅为训练 Job 分析创建调试器自定义规则 (p. 1866).

根据内置规则状态采取操作

您可以将调试器与 Amazon CloudWatch Events 结合使用，Amazon Lambda. 您可以根据规则状态自动执行操作，例如提前停止培训作业和通过电子邮件或短信设置通知。当调试器规则检测到问题并触发`"IssuesFound"`评估状态，CloudWatch Event 会检测规则状态更改并调用 Lambda 函数采取行动。要针对训练问题配置自动操作，请参阅使用 Amazon CloudWatch 创建规则操作和Amazon Lambda (p. 1876).

使用 smDebug 客户端库深入了解数据

您可以使用 SMDebug 工具访问和分析调试器收集的训练数据。这些区域有：`TrainingJob`和`create_trial`类加载调试器保存的指标和张量。这些课程提供了扩展的课程方法，用于实时分析数据或在训练结束后分析数据。SMDebug 库还提供了可视化工具：合并框架指标的时间表以聚合不同的分析、折线图和热图以跟踪系统利用率，以及直方图以查找步长时间异常值。要了解有关 SMDebug 库工具的更多信息，请参阅使用 smDebug 客户端库分析数据 (p. 1920).

监控和分析训练 Job 指标

Amazon CloudWatch 支持高精度定制指标，其最好的分辨率是 1 秒钟。但是，分辨率越好，CloudWatch 指标的使用寿命就越短。对于 1 秒的频率分辨率，CloudWatch 指标可用期为 3 小时。有关 CloudWatch 指标的更多信息，请参阅。GetMetricStatistics中的Amazon CloudWatch API 参考.

如果您想以更精细的分辨率分析训练工作，最小至 100 毫秒（0.1 秒）的粒度，并随时将训练指标无限期存储在 Amazon S3 中以进行自定义分析，请考虑使用Amazon SageMaker 调试器. SageMaker Debug 提供了自动检测常见训练问题的内置规则；它可以检测硬件资源利用率问题（例如 CPU、GPU 和 I/O 瓶颈）和非融合模型问题（例如过度拟合、消失渐变和爆炸张量）。

SageMaker 调试器还通过 Studio 及其性能分析报告提供可视化效果。与 CloudWatch 指标不同，该指标累积 CPU 和 GPU 核心的资源利用率，并在多个实例中实现平均数，与此不同，调试器跟踪每个核心的利用率。这使您能够在向上扩展到更大的计算群集时识别硬件资源的不平衡使用情况。要探索调试器的可视化效果，请参阅SageMaker 调试器见解仪表板演练、调试器性能分析报告演练, 和使用 smDebug 客户端库分析数据.

监控系统利用率和检测瓶颈

借助 Amazon SageMaker 调试器监控，您可以衡量 Amazon EC2 实例的硬件系统资源利用率。监控适用于使用 SageMaker 框架估算器（TensorFlow、PyTorch 和 MxNet）和通用 SageMaker 估算器（SageMaker 内置算法和您自己的自定义容器）构建的任何 SageMaker 培训作业。调试器内置的监控规则可检测系统瓶颈问题，并在检测到瓶颈问题时通知您。

要了解如何启用调试器系统监控，请参阅使用亚马逊配置调试器SageMakerPython 开发工具包 (p. 1796)然后配置调试器监视硬件系统资源利用率 (p. 1800).

有关可用内置监控规则的完整列表，请参阅。调试器用于监视硬件系统资源利用率的内置规则（系统指标） (p. 1826).

分析框架操作

借助 Amazon SageMaker 调试器配置文件，您可以分析深度学习框架操作。您可以使用 SageMaker TensorFlow 训练容器、SageMaker PyTorch 框架容器和自己的训练容器来分析模型训练。使用调试器的分析功能，您可以深入了解为执行训练作业而执行的 Python 运算符和函数。调试器支持详细的分析、Python 分析、数据加载器分析和 Horovod 分布式训练分析。您可以合并分析的时间表，以便与系统瓶颈关联起来。调试器用于分析监视框架操作相关问题的内置规则，包括训练开始前下载数据导致的训练初始化时间过长以及训练循环中的步长期异常值。

要了解如何配置调试器以进行框架分析，请参阅使用亚马逊配置调试器SageMakerPython 开发工具包 (p. 1796)然后配置调试器框架分析 (p. 1801).

有关分析的可用内置规则的完整列表，请参阅。用于分析框架指标的调试器内置规则 (p. 1827).

调试模型输出张量

调试可用于深度学习框架AmazonDeep Learning Containers 和 SageMaker 培训容器。有关完全支持的框架版本（请参阅支持的框架和算法 (p. 1785)），调试器会自动注册钩子以收集输出张量，并且您可以直接运行训练脚本。对于带有一个星号的版本，你需要手动注册钩子才能收集张量。调试器提供了预配置的张量集合，其中包含广义名称，您可以在不同框架中使用这些名称 如果你想自定义输出张量配置，你也可以使用 CollectionConfig 和 debugGerHookConfig API 操作和Amazon SageMaker Python 开发工具包来配置你自己的张量集合。调试器内置的调试规则分析输出张量并识别阻止模型最大限度地减少损失函数的模型优化问题。例如，这些规则确定了过度拟合、过度训练、损失不减少、张量爆炸和消失的渐变。

要了解如何配置调试器以调试输出张量，请参阅使用亚马逊配置调试器SageMakerPython 开发工具包 (p. 1796)然后配置调试程序挂钩以保存张量 (p. 1804).

有关可用内置调试规则的完整列表，请参阅。调试器用于调试模型训练数据（输出张量）的内置规则 (p. 1827).

Amazon SageMaker 调试器高级主题和参考文档

以下各节包含高级主题、API 操作的参考文档、例外情况以及调试器的已知限制。

主题

- Amazon SageMaker 调试器 API 操作 (p. 1932)
- 将调试程序 Docker 映像用于内置规则或自定义规则 (p. 1933)
- Amazon SageMaker 调试程序异常 (p. 1935)
- Amazon SageMaker 调试程序的注意事项 (p. 1936)
- Amazon SageMaker 调试器使用统计信息 (p. 1938)

Amazon SageMaker 调试器 API 操作

Amazon SageMaker Debugger 在多个位置都有 API 操作，这些操作用于实施模型训练的监控和分析。

Amazon SageMaker 调试器还提供了开源`SMDebug`Python 库awslabs/sagemaker-调试器，这些步骤用于配置内置规则、定义自定义规则和注册钩子，从训练作业中收集输出张量数据。

这些区域有：Amazon SageMaker Python 开发工具包是一个专注于机器学习实验的高级开发工具包。此开发工具包可用于部署通过`SMDebug`使用 SageMaker 估计器监控和分析这些张量的 Python 库。

调试程序向 Amazon SageMaker API 添加了一些操作和类型，这些类型可让平台在训练模型时使用调试程序，并可让平台管理输入和输出的配置。

- `CreateTrainingJob`和`UpdateTrainingJob`使用以下调试器 API 来配置张量集合、规则、规则图像和性能分析选项：
  - `CollectionConfiguration`
  - `DebugHookConfig`
  - `DebugRuleConfiguration`
  - `TensorBoardOutputConfig`
  - `ProfilerConfig`
  - `ProfilerRuleConfiguration`
- `DescribeTrainingJob`提供培训作业的完整描述，包括以下调试器配置和规则评估状态：
  - `DebugHookConfig`
  - `DebugRuleConfiguration`
  - `DebugRuleEvaluationStatus`
  - `ProfilerConfig`
  - `ProfilerRuleConfiguration`
  - `ProfilerRuleEvaluationStatus`

分析模型训练时，规则配置 API 操作使用 SageMaker 处理功能。有关 SageMaker 处理的更多信息，请参阅处理数据 (p. 728).

将调试程序 Docker 映像用于内置规则或自定义规则

Amazon SageMaker 提供了两组适用于规则的 Docker 映像：一组用于评估由 SageMaker 提供的规则（内置规则），另一组用于评估 Python 源文件中提供的自定义规则。

如果您将Amazon SageMaker Python 开发工具包，您只需将 SageMaker 高级调试器 API 操作与 SageMaker Estumator API 操作一起使用，而无需手动检索调试器 Docker 映像并配置`ConfigureTrainingJob`API。

如果您没有使用 SageMaker Python 开发工具包，则必须为调试程序规则检索相关的预构建容器基础映像。Amazon SageMaker 调试程序为内置规则和自定义规则提供了预构建的 Docker 映像，映像存储在 Amazon Elastic Container Registry (Amazon ECR) 中。要从 Amazon ECR 存储库中提取映像（或向一个存储库推送映像），请使用映像的完整名称注册表 URL 使用`CreateTrainingJob`API。SageMaker 使用以下 URL 模式作为调试器规则容器映像注册表地址。

```
<account_id>.dkr.ecr.<Region>.amazonaws.com/<ECR repository name>:<tag>
```

对于每个中的账户 IDAmazon区域、Amazon ECR 存储库名称和标签值，请参阅以下主题。

主题

- 内置规则评估程序的 Amazon SageMaker 调试程序的注册表 URL (p. 1933)
- Amazon SageMaker 调试程序的自定义规则评估程序的注册表 URL (p. 1934)

内置规则评估程序的 Amazon SageMaker 调试程序的注册表 URL

可以将以下值用于为 Amazon SageMaker 调试程序提供内置规则的映像的注册表 URL 的组成部分。有关账户 ID，请参阅下表。

ECR 存储库名称：sagemaker-debugger-rules

标签：最新

完整注册表 URL 示例：

```
904829902805.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-debugger-rules:latest
```

内置规则容器映像的账户 IDAmazon区域

区域	account_id
af-south-1	314341159256
ap-east-1	199566480951
ap-northeast-1	430734990657
ap-northeast-2	578805364391
ap-south-1	904829902805
ap-southeast-1	972752614525
ap-southeast-2	184798709955
ca-central-1	519511493484
cn-north-1	618459771430
cn-northwest-1	658757709296
eu-central-1	482524230118
eu-north-1	314864569078
eu-south-1	563282790590
eu-west-1	929884845733
eu-west-2	250201462417
eu-west-3	447278800020
me-south-1	986000313247
sa-east-1	818342061345
us-east-1	503895931360
us-east-2	915447279597
us-west-1	685455198987
us-west-2	895741380848
us-gov-west-1	515509971035

Amazon SageMaker 调试程序的自定义规则评估程序的注册表 URL

可以将以下值用于为 Amazon SageMaker 调试程序提供自定义规则评估程序的映像的注册表 URL 的组成部分。有关账户 ID，请参阅下表。

ECR 存储库名称：sagemaker-debugger-rule-evaluator

标签：最新

完整注册表 URL 示例：

```
552407032007.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-debugger-rule-
evaluator:latest
```

自定义规则容器映像的账户 IDAmazon区域

区域	account_id
af-south-1	515950693465
ap-east-1	645844755771
ap-northeast-1	670969264625
ap-northeast-2	326368420253
ap-south-1	552407032007
ap-southeast-1	631532610101
ap-southeast-2	445670767460
ca-central-1	105842248657
cn-north-1	617202126805
cn-northwest-1	658559488188
eu-central-1	691764027602
eu-north-1	091235270104
eu-south-1	335033873580
eu-west-1	606966180310
eu-west-2	074613877050
eu-west-3	224335253976
me-south-1	050406412588
sa-east-1	466516958431
us-east-1	864354269164
us-east-2	840043622174
us-west-1	952348334681
us-west-2	759209512951
us-gov-west-1	515361955729

Amazon SageMaker 调试程序异常

Amazon SageMaker 调试程序旨在注意执行规则所需的张量可能并非在每个步骤中都可用。因此，它会引发几个异常，这将使您能够控制张量丢失时发生的情况。在 smdebug.exceptions 模块中提供了这些异常。可按如下方式导入它们：

```
from smdebug.exceptions import *
```

提供了以下异常：

- `TensorUnavailableForStep`— 请求的张量对步骤不可用。这可能意味着此步骤可能不会由挂钩保存，或者此步骤可能已保存一些张量，但请求的张量不在其中。请注意，当您看到该异常时，这意味着此张量将来绝不可用于此步骤。如果张量已为此步骤保存减少量，则它会告知您可以查询这些减少量。
- `TensorUnavailable`— 此张量未保存或尚未由`smdebug`API。这意味着，该张量永远对 `smdebug` 中的任何步骤不可见。
- `StepUnavailable`— 步骤未保存，并且调试程序没有来自步骤的数据。
- `StepNotYetAvailable`— 步骤对尚不可见`smdebug`. 如果训练仍在进行中，将来可能会提供该步骤。在该步骤变得可用时，调试程序自动加载新数据。
- `NoMoreData`— 在训练结束时引发。一旦您看到此项，便知道没有其他需要保存的步骤和张量。
- `IndexReaderException`— 索引读取器无效。
- `InvalidWorker`— 调用了无效的工作线程。
- `RuleEvaluationConditionMet`— 在步骤中评估规则导致满足条件。
- `InsufficientInformationForRuleInvocation`— 提供的信息不足，无法调用规则。

Amazon SageMaker 调试程序的注意事项

在使用 Amazon SageMaker 调试程序时，请注意以下事项。

分布式训练注意事项

以下列表显示了在具有深度学习框架和各种分布式培训选项的训练作业中使用调试器的有效性范围和注意事项。

- Horovod

  使用调试器进行 Horovod 培训工作的有效性范围

深度学习框架	Apache MXNet	TensorFlow 1.x	TensorFlow 2.x	TensorFlow 2.x 与克拉斯	PyTorch
监控系统瓶颈	是	是	是	是	是
分析框架操作	否	否	否	是	是
调试模型输出张量	是	是	是	是	是

- SageMaker 并行分布式数据

  使用调试器进行 SageMaker 分布式数据并行训练作业的有效性范围

深度学习框架	TensorFlow 2.x	TensorFlow 2.x 与克拉斯	PyTorch
监控系统瓶颈	是	是	是
分析框架操作	没有 *	没有 **	是
调试模型输出张量	是	是	是

  * 调试器不支持 TensorFlow 2.x 的框架分析。

  ** SageMaker 分布式数据并行不支持使用 Keras 实现的 TensorFlow 2.x。

- SageMaker 分布式模型并行— 调试器不支持 SageMaker 分布式模型并行训练。
- 使用 SageMaker 检查点进行分散— 启用分布式训练选项和 SageMaker 检查点时，调试器不能用于培训作业。您可能会看到如下所示的错误：

```
SMDebug Does Not Currently Support Distributed Training Jobs With Checkpointing Enabled
```

  要将调试器用于具有分布式训练选项的训练作业，您需要禁用 SageMaker 检查点并将手动检查点功能添加到训练脚本中。有关在分布式训练选项和检查点中使用 Debugger 的更多信息，请参阅使用SageMaker与亚马逊并行分布式数据SageMaker调试程序和检查点 (p. 1996)和保存检查点 (p. 2041).
- 参数服务器— 调试器不支持基于参数服务器的分布式训练。
- 分析分布式培训框架操作，例如AllReducedSageMaker 分布式数据并行运行Horovod 操作，不可用。

监控系统瓶颈和分析框架操作的注意事项

- 适用于AmazonTensorFlow，无法使用默认值收集数据加载器指标local_path的设置FrameworkProfile类。必须手动配置路径并结束"/". 例如：

```
FrameworkProfile(local_path="/opt/ml/output/profiler/")
```

- 适用于AmazonTensorFlow，在训练作业运行时，无法更新数据加载器性能分析配置。
- 适用于AmazonTensorFlow，NoneType在 TensorFlow 2.3 训练作业和详细分析选项中使用分析工具和笔记本示例时，可能会发生错误。
- 只有 Keras API 才支持 Python 分析和详细分析。
- 要访问 TensorFlow 和 PyTorch 的深度分析功能，目前您必须指定最新的Amazon使用 CUDA 11 进行深度学习容器映像。例如，您必须在 TensorFlow 和 PyTorch 估计器中指定特定的图像 URI，如下所示：
  - TensorFlow

```
image_uri = f"763104351884.dkr.ecr.{region}.amazonaws.com/tensorflow-training:2.3.1-gpu-py37-cu110-ubuntu18.04"
```

  - PyTorch

```
image_uri = f"763104351884.dkr.ecr.{region}.amazonaws.com/pytorch-training:1.6.0-gpu-py36-cu110-ubuntu18.04"
```

调试模型输出张量的注意事项

- 避免使用功能性 API 操作。调试器无法从由功能性 API 操作组成的 PyTorch 和 MxNet 训练脚本中收集模型输出张量。
  - 调试器无法从torch.nn.functionalAPI 操作。当你编写 PyTorch 训练脚本时，建议使用torch.nn而是模块。
  - 调试器无法从混合模块中的 MxNet 功能对象收集模型输出张量。例如，ReLU 激活 (F.relu) 无法从以下示例中收集输出mxnet.gluon.HybridBlock和F中的hybrid_forwardfunction.

```
import mxnet as mx
from mxnet.gluon import HybridBlock, nn

class Model(HybridBlock):
    def __init__(self, **kwargs):
        super(Model, self).__init__(**kwargs)
        # use name_scope to give child Blocks appropriate names.
        with self.name_scope():
```

```
            self.dense0 = nn.Dense(20)
            self.dense1 = nn.Dense(20)

    def hybrid_forward(self, F, x):
        x = F.relu(self.dense0(x))
        return F.relu(self.dense1(x))

model = Model()
model.initialize(ctx=mx.cpu(0))
model.hybridize()
model(mx.nd.zeros((10, 10), ctx=mx.cpu(0)))
```

Amazon SageMaker 调试器使用统计信息

使用 Amazon SageMaker 调试器自动生成的报告时，请注意以下因素。

调试器分析报告使用情况

对于所有 SageMaker 培训作业，Amazon SageMaker 调试器会运行剖析器报告 (p. 1828)规则并自动生成SageMaker 调试器性能分析报告 (p. 1897). 这些区域有：`ProfilerReport`规则提供了一个 Jupyter 笔记本文件（`profiler-report.ipynb`）生成相应的 HTML 文件（`profiler-report.html`)。

调试器通过将代码包含在 Jupyter 笔记本中收集唯一`ProfilerReport`如果用户打开最终的规则的处理作业 ARN`profiler-report.html`文件。

调试器只收集有关用户是否打开最终 HTML 报告的信息。它没有从培训作业、培训数据、培训脚本、处理作业、日志或性能分析报告本身的内容中收集任何信息。

使用下列选项之一，您可以选择退出使用情况统计数据收集。

（推荐）选项 1：在运行培训 Job 之前选择退出

要选择退出，你需要添加以下调试器`ProfilerReport`规则配置到你的培训作业请求中。

SageMaker Python SDK

```
estimator=sagemaker.estimator.Estimator(
    ...

    rules=ProfilerRule.sagemaker(
        base_config=rule_configs.ProfilerReport()
        rule_parameters={"opt_out_telemetry": "True"}
    )
)
```

Amazon CLI

```
"ProfilerRuleConfigurations": [
    {
        "RuleConfigurationName": "ProfilerReport-1234567890",
        "RuleEvaluatorImage": "895741380848.dkr.ecr.us-west-2.amazonaws.com/sagemaker-
debugger-rules:latest",
        "RuleParameters": {
            "rule_to_invoke": "ProfilerReport",
            "opt_out_telemetry": "True"
        }
    }
]
```

Amazon SDK for Python (Boto3)

```
ProfilerRuleConfigurations=[
    {
        'RuleConfigurationName': 'ProfilerReport-1234567890',
        'RuleEvaluatorImage': '895741380848.dkr.ecr.us-west-2.amazonaws.com/sagemaker-
debugger-rules:latest',
        'RuleParameters': {
            'rule_to_invoke': 'ProfilerReport',
            'opt_out_telemetry': 'True'
        }
    }
]
```

选项 2：培训 Job 完成后选择退出

要在培训完成后选择退出，你需要修改`profiler-report.ipynb`文件。

> **Note**
>
> HTML 报告自动生成没有选项 1已添加到您的培训工作请求中，即使您选择退出使用情况后，仍会报告使用统计选项 2.

1. 按照中的说明下载调试器性能分析报告文件下载调试器性能分析报告 (p. 1897)页.
2. 在`/ProfilerReport-1234567890/profiler-output`目录，打开`profiler-report.ipynb`.
3. Add**`opt_out=True`**到`setup_profiler_report()`函数在第五个代码单元中，如下例代码所示：

```
setup_profiler_report(processing_job_arn, opt_out=True)
```

4. 运行代码单元以完成选项。

使用 SageMaker 执行自动模型优化

亚马逊 SageMaker 自动模型优化也称为超参数优化，它使用您指定的算法和超参数范围，在您的数据集上运行多个训练作业来查找模型的最佳版本。然后，它会选择超参数值来生成性能最佳的模型（按所选指标衡量）。

例如，假设您希望解决营销数据集上的二进制分类问题。您的目标是最大化曲线下面的区域（AUC）通过训练算法的指标XGBoost 算法 (p. 1731)模型。您不知道应该使用 `eta`、`alpha`、`min_child_weight` 和 `max_depth` 超参数的哪些值来训练最佳模型。要查找这些超参数的最佳值，可以指定以下值的范围：SageMaker 超参数优化搜索以查找导致训练作业绩最出色的值组合，并根据您选择的目标指标进行评估。超参数优化会启动训练作业，使用您指定范围中的超参数值，然后返回具有最高 auc 的训练作业。

您可以使用 SageMaker 使用内置算法、自定义算法和自动模型优化 SageMaker 用于机器学习框架的预构建容器。

亚马逊 SageMaker 在运行培训作业时，自动模型调整可以使用 Amazon EC2 竞价型实例优化成本。有关托管现场训练的更多信息，请参阅。Amazon SageMaker 中的托管的 Spot (p. 2085).

在开始使用超参数优化之前，您应该已经具有明确定义的机器学习问题，包括下列内容：

- 一个数据集
- 了解需要训练的算法的类型
- 明确了解如何衡量成功

您还应准备数据集和算法，以便它们在其中运行。 SageMaker 并至少成功运行训练作业一次。有关设置和运行训练作业的信息，请参阅开始使用 Amazon SageMaker (p. 30)。

主题

- 超参数优化的工作方式 (p. 1940)
- 定义指标 (p. 1941)
- 定义超参数范围 (p. 1942)
- 通过超参数优化优化优化多个算法以找到最佳模型 (p. 1943)
- 例如：超参数优化 Job (p. 1947)
- 提前停止训练作业 (p. 1957)
- 运行热启动超参数优化作业 (p. 1959)
- 自动模型优化的资源限制 (p. 1962)
- 超参数优化的最佳实践 (p. 1963)

超参数优化的工作方式

随机搜索

在随机搜索中，超参数优化从为其启动的每个训练作业的超参数指定的范围内选择一个随机值组合。由于超参数值的选择不依赖于以前训练作业的结果，因此您可以运行最大数量的并发训练作业，而不影响搜索性能。

有关使用随机搜索的示例笔记本，请访问 https://github.com/awslabs/amazon-sagemaker-examples/blob/master/hyperparameter_tuning/xgboost_random_log/hpo_xgboost_random_log.ipynb。

贝叶斯搜索

贝叶斯搜索将超参数优化视为 [回归] 问题。给定一组输入特征（超参数），超参数优化会针对您选择的指标来优化模型。为解决递归问题，超参数优化会猜测哪些超参数组合有可能获得最佳结果，然后运行训练作业来测试这些值。在测试了第一组超参数值之后，超参数优化使用回归来选择要测试的下一组超参数值。

超参数调整使用亚马逊 SageMaker 实施贝叶斯优化。

在选择下一个训练作业的最佳超参数时，超参数优化考虑迄今为止所了解到的有关此问题的全部信息。有时，它会选择与导致先前最佳训练作业的组合接近的超参数值组合，以逐步提高性能。这使得超参数优化可以利用最佳的已知结果。其他时候，它会选择一组远离已尝试过的值的超参数值。这使得它可以探索超参数值的范围，以尝试找到新的尚未了解的领域。探索/利用的权衡在许多机器学习问题中都很常见。

有关贝叶斯优化的更多信息，请参阅以下内容：

关于贝叶斯优化的基本主题

- 高成本函数贝叶斯优化及其在主动用户建模和分层强化学习中的应用的教程
- 机器学习算法的实用贝叶斯优化
- 使人类脱离循环：贝叶斯优化回顾

加快贝叶斯优化

- 超频段：基于 Bandit 的超参数优化方法
- Google Vizier：黑盒优化服务
- 使用贝叶斯神经网络进行学习曲线预测

- 利用学习曲线加快深度神经网络的自动超参数优化

高级建模和迁移学习

- 可扩展的超参数迁移学习
- 具有树结构化依赖项的贝叶斯优化
- 具有强健的贝叶斯神经网络的贝叶斯优化
- 使用深度神经网络的可扩展贝叶斯优化
- 非固定函数贝叶斯优化的输入变形

> **Note**
>
> 超参数优化可能不会改进您的模型。这是一种用于生成机器解决方案的高级工具，因此，在科学开发流程中应该予以考虑。
> 在您生成复杂机器学习系统时，例如深度学习神经网络，探讨所有可能的组合不切实际。超参数优化通过尝试模型的不同变体，重点关注您指定范围内最有可能的超参数值组合，从而提高您的工作效率。要获得良好的结果，您需要选择合适的探索范围。由于算法本身是推测性的，超参数优化模型可能无法收敛到最佳答案，即使可能的最佳值组合位于您选择的范围内。

定义指标

当您使用亚马逊之一时 SageMaker 内置算法，您无需定义指标。内置算法自动将指标发送到超参数优化。您需要选择内置算法发出的指标之一作为优化作业的目标指标。有关内置算法发出的指标的列表，请参阅中列出的相应算法的模型调整部分使用 Amazon SageMaker 内置算法或预训练模型 (p. 787).

要针对机器学习模型优化超参数，优化作业使用训练算法写入日志的指标来评估所启动的训练作业。亚马逊 SageMaker 超参数调整解析算法的`stdout`和`stderr`用于查找算法指标（例如丢失或验证准确度）的流，这些指标显示了模型在数据集上的性能

> **Note**
>
> 这些是相同的指标 SageMaker 发送到 CloudWatch 日志。有关更多信息，请参阅 使用 Amazon CloudWatch 记录 Amazon SageMaker 事件 (p. 2820)。

如果使用自己的算法进行超参数优化，请确保您的算法通过将评估数据写入 `stderr` 或 `stdout`，至少发出一个指标。

> **Note**
>
> 超参数优化将一个额外的超参数 `_tuning_objective_metric` 发送到训练算法。此超参数指定要用于超参数优化作业的目标指标，以便在训练期间您的算法可以使用该信息。

您可为优化作业定义最多 20 个要监控的指标。您选择这些指标之一作为目标指标，超参数优化使用该指标来评估训练作业。训练作业在得到了目标指标最佳值时，超参数优化作业返回该作业作为最佳训练作业。

您通过为优化作业所监控的各个指标指定名称和正则表达式，定义优化作业的指标。设计正则表达式以捕获您的算法发出的指标值。您将指标传递到 `CreateHyperParameterTuningJob` 操作中的 `TrainingJobDefinition` 参数，作为 `AlgorithmSpecification` 字段的 `MetricDefinitions` 字段。

下面的示例定义了 4 个指标：

```
=[
    {
        "Name": "loss",
```

```
        "Regex": "Loss = (.*?);",
    },
    {
        "Name": "ganloss",
        "Regex": "GAN_loss=(.*?);",
    },
    {
        "Name": "discloss",
        "Regex": "disc_train_loss=(.*?);",
    },
    {
        "Name": "disc-combined",
        "Regex": "disc-combined=(.*?);",
    },
]
```

下面是该算法写入的日志的示例：

```
GAN_loss=0.138318;  Scaled_reg=2.654134; disc:[-0.017371,0.102429] real 93.3% gen 0.0%
 disc-combined=0.000000; disc_train_loss=1.374587;  Loss = 16.020744;  Iteration 0 took
 0.704s;  Elapsed=0s
```

使用正则表达式 (regex) 匹配算法的日志输出并捕获指标的数值。例如，在上面定义的 `loss` 指标的正则表达式中，正则表达式的第一部分查找完全匹配的文本“Loss = ”，表达式 `(.*?);` 捕获零个或多个任意字符，直到遇到第一个分号字符为止。在此表达式中，括号告诉正则表达式捕获括号内的内容，`.` 表示任何字符，`*` 表示零或更多个字符，`?` 表示仅捕获到遇到 `;` 字符的第一个实例为止。

选择您定义的指标之一作为优化作业的目标指标。如果您在使用 API，则在发送到 `CreateHyperParameterTuningJob` 操作的 `HyperParameterTuningJobConfig` 参数的 `HyperParameterTuningJobObjective` 字段中指定 `name` 键的值。

定义超参数范围

超参数优化通过搜索您为每个可优化的超参数指定的值集，查找针对您模型的最佳超参数值。超参数和范围的选择会对优化作业的性能产生巨大影响。有关如何选择超参数和范围的指南，请参阅超参数优化的最佳实践 (p. 1963)。

要使用低级 API 定义超参数范围，您需要指定超参数名称，并在传递到 `CreateHyperParameterTuningJob` 操作的 `HyperParameterTuningJobConfig` 参数的 `ParameterRanges` 字段中指定值的范围。`ParameterRanges` 字段有三个子字段，分别用于分类、整数和连续超参数范围。您可以定义搜索最多 20 个超参数。每个分类超参数范围的值都计为限制内的一个超参数。超参数范围具有以下结构：

```
"ParameterRanges": {
      "CategoricalParameterRanges": [
        {
          "Name": "tree_method",
          "Values": ["auto", "exact", "approx", "hist"]
        }
      ],
      "ContinuousParameterRanges": [
        {
          "Name": "eta",
          "MaxValue" : "0.5",
          "MinValue": "0",
          "ScalingType": "Auto"
        }
      ],
      "IntegerParameterRanges": [
        {
```

```
        "Name": "max_depth",
        "MaxValue": "10",
        "MinValue": "1",
        "ScalingType": "Auto"
      }
    ]
}
```

超参数标度

对于整数和连续超参数范围，您可以通过为超参数范围的 `ScalingType` 字段指定值，选择超参数优化要用于搜索值范围的标度。您可从以下标度类型中进行选择：

自动

SageMaker 超参数优化为超参数选择最佳标度。

线性

超参数优化使用线性标度在超参数范围中搜索值。通常，如果从最低到最高的所有值的范围相对较小（在一个数量级内），则选择此选项，因为从该范围均匀搜索值将使您合理地探索整个范围。

对数

超参数优化使用对数标度在超参数范围中搜索值。

对数标度仅适用于其值大于 0 的范围。

当您要搜索跨多个数量级的范围时，请选择对数标度。例如，如果您要调优化调整线性学习者模型 (p. 1657)模型，并为 `learning_rate` 超参数指定值介于 .0001 和 1.0 之间的范围，则与按线性标度搜索相比，按对数标度统一搜索会使您更好地对整个范围进行采样，因为平均而言按线性标度搜索对于仅在 .1 和 1.0 之间的值投入 90% 的训练预算，而对于 .0001 和 .1 之间的值仅剩下 10% 的训练预算。

反向对数

超参数优化使用反向对数标度搜索超参数范围中的值。仅连续超参数范围支持反向对数标度。整数超参数范围不支持它。

反向对数标度仅适用于完全位于 0 <= x < 1.0 内的范围。

当您搜索对非常接近 1 的微小变化高度敏感的范围时，请使用反向对数标度。

有关使用超参数标度的示例笔记本，请访问 https://github.com/awslabs/amazon-sagemaker-examples/blob/master/hyperparameter_tuning/xgboost_random_log/hpo_xgboost_random_log.ipynb。

通过超参数优化优化优化多个算法以找到最佳模型

使用亚马逊创建新的超参数优化 (HPO) 作业 SageMaker 调整多个算法，必须提供适用于所有待测试算法的作业设置以及每种算法的训练定义。您还必须指定要用于优化作业的资源。

- 这些区域有：任务设置配置包括热启动、提早停止和调整策略。仅当优化单个算法时热启动和提前停止才可用。
- 这些区域有：训练作业定义可在需要时指定名称、算法来源、目标指标和值范围，以配置每个训练作业的超参数值集。它为每个训练作业配置数据输入、数据输出位置和任何检查点存储位置的通道。该定义还配置要为每个培训作业部署的资源，包括实例类型和计数、托管现场培训和停止条件。
- 这些区域有：调整作业资源：部署，包括超参数优化作业可并发运行的并发训练作业的最大数目以及超参数优化作业可运行的最大训练作业数。

入门

您可以从控制台创建新的超参数优化作业、克隆作业、向作业添加或编辑标签。您还可以使用搜索功能按名称、创建时间或状态查找作业。或者，您也可以使用 SageMaker API 进行超参数调整作业。

- 在控制台中：要创建新任务，请打开亚马逊 SageMaker 控制台在https://console.aws.amazon.com/sagemaker/，选择超参数优化作业来自 的训练，菜单，然后选择创建超参数优化作业. 然后按照配置步骤为要使用的每个算法创建训练作业。这些步骤在为一个或多个算法创建超参数优化调整 Job（控制台）(p. 1944)主题。

  Note

  请注意，当您开始配置步骤时，热启动和提前停止功能不适用于多算法 HPO。如果要使用这两种功能，则一次只能优化单个算法。
- 使用 API：有关如何使用 SageMaker 要创建超参数优化作业的 API，请参阅。示例：超参数优化 Job. 当你打电话`CreateHyperParameterTuningJob`要优化多个算法，必须使用提供训练定义列表。`TrainingJobDefinitions`而不是指定单个`TrainingJobDefinition`. 根据调整的算法数量，您必须只选择其中一种定义类型。

主题

- 为一个或多个算法创建超参数优化调整 Job（控制台）(p. 1944)
- 管理超参数调整和训练作业 (p. 1947)

为一个或多个算法创建超参数优化调整 Job（控制台）

要为一个或多个算法创建新的超参数优化 (HPO) 调整作业，您需要定义调整作业的设置，为每个正在调整的算法创建训练作业定义，并为调整作业配置资源。

主题

- 定义作业设置 (p. 1944)
- 创建培训 Job 定义 (p. 1945)
- 配置优化 Job 资源 (p. 1947)
- 查看并创建 HPO 调整 Job (p. 1947)

定义作业设置

您的优化作业设置将应用于 HPO 优化作业中的所有算法。仅当优化单个算法时热启动和提前停止才可用。定义作业设置后，您将为要优化的每个算法或变体创建单独的训练定义。

热启动

如果克隆了此作业，则可以选择使用之前优化作业的结果来提高此新优化作业的性能。这是热启动功能，只有在优化单个算法时才可用。 选择此选项后，最多可以选择要使用的五个超参数优化作业。 或者，您可以使用迁移学习向父优化作业添加额外数据。选择此选项时，您可以选择以前的一个优化作业作为父作业。

Note

热启动只有与 2018 年 10 月 1 日之后创建的优化作业兼容。有关更多信息，请参阅运行热启动作业。

提前停止

要减少计算时间并避免过度拟合模型，训练作业不太可能改善超参数优化作业的当前最佳目标指标，则可以提前停止训练作业。与热启动一样，此功能仅在优化单个算法时才可用。这是一个没有配置选项的自动功

能，默认情况下它处于禁用状态。 有关提前停止的工作原理、支持早期停止的算法以及如何在自己的算法中使用早期停止的更多信息，请参阅提前停止训练作业.

优化策略

优化策略可以是随机的，也可以是贝叶斯。它指定自动优化如何在指定的超参数范围内进行搜索。您可以在后面的步骤中指定范围。随机搜索从指定范围中选择值的随机组合，并且可以同时运行。贝叶斯搜索根据以前选择的历史已知的内容可能获得最佳结果来选择值。有关更多信息，请参阅。超参数优化的工作方式.

标签

您输入标签作为键值对，以将元数据分配给优化作业以帮助您管理它们。值不是必需的。您可以只使用键。要查看与作业关联的键，请选择标签用于调整作业的详细信息页面上的标签。有关使用优化作业标签的更多信息，请参阅管理超参数调整和训练作业 (p. 1947)

创建培训 Job 定义

要创建训练作业定义，您需要配置算法和参数、定义数据输入和输出以及配置资源。您必须提供至少一个`TrainingJobDefinition`针对每个 HPO 优化作业。每个训练定义都指定算法的配置。要为您的训练作业创建多个定义，您可以克隆作业定义。克隆作业可以节省时间，因为它会复制所有作业设置，包括数据通道和输出构件的 S3 存储位置。然后，您可以仅针对配置算法选项所需的更改来编辑克隆的作业。

主题

- 配置算法和参数 (p. 1945)
- 定义数据输入和输出 (p. 1946)
- 配置培训 Job 资源 (p. 1946)
- 添加或克隆训练 Job (p. 1947)

配置算法和参数

调整作业的每个训练作业定义都需要一个名称、访问服务的权限、算法选项的说明、客观指标和值范围（如果需要），才能为每个训练作业配置超参数值集。

名称

为训练定义提供唯一名称。

权限

亚马逊 SageMaker 需要有权代表您调用其他服务。选择 IAM 角色或者让Amazon创建具有`AmazonSageMakerFullAccess`附加 IAM 策略。

可选的安全设置

网络隔离设置阻止容器进行任何出站网络调用。以下内容是必填项的AmazonMarketplace 机器学习产品。

您还可以选择使用私有 VPC。

> Note
>
> 仅当从 API 创建作业定义时，容器间加密才可用。

算法选项

您可以选择其中一种内置算法、您自己的算法、您自己的带算法的容器，也可以从AmazonMarketplace。

- 如果您选择内置算法，则该算法预填充了 ECR 映像信息。

- 如果您选择自己的容器，则必须指定 ECR 映像信息。 您可以选择算法的输入模式作为文件或管道。
- 如果您计划使用 Amazon S3 中的 .CSV 文件提供数据，则应选择该文件。

指标

选择内置算法时，系统会为您提供指标。如果您选择自己的算法，则需要定义指标。您可为优化作业定义最多 20 个要监控的指标，其中一个指标必须选择作为目标指标。有关如何为优化作业定义指标的更多信息，请参阅定义指标 (p. 1941).

目标指标

要找到最佳的训练作业，请设置目标指标以及是最大化还是最小化。训练作业完成后，您可以查看优化作业详细信息页面，了解使用此目标指标找到的最佳训练作业的摘要。

超参数配置

选择内置算法时，会使用针对优化算法优化的范围为您设置其超参数的默认值。您可以根据您认为合适的情况更改这些值。例如，您可以通过将参数的类型设置为，从而不是范围，为超参数设置固定值，而不是范围。静态的. 每个算法都有不同的必需参数和可选参数。有关更多信息，请参阅 。超参数优化的最佳实践和定义超参数范围.

定义数据输入和输出

调整作业的每个训练作业定义都必须为每个训练作业配置数据输入、数据输出位置以及可选的任何检查点存储位置的渠道。

输入数据配置

输入数据由通道定义，每个通道都有自己的源位置（Amazon S3 或 Amazon Elastic File System）、压缩和格式选项。您最多可以定义 20 个输入源通道。如果您选择的算法支持多个输入通道，您也可以指定这些通道。例如，在使用 XGBoost 流失预测笔记本时，您可以添加两个通道：训练和验证。

检查点配置

在训练期间定期生成检查点。您必须选择要保存的检查点的 Amazon S3 位置。检查点用于指标报告，还用于恢复托管型 Spot 训练作业。有关更多信息，请参阅 在 Amazon SageMaker 中使用检查点 (p. 2087)。

输出数据配置

您必须为要存储的训练作业构件定义 Amazon S3 位置。您可以选择使用向输出添加加加密。AmazonKey Management Service (AmazonKMS) 密钥。

配置培训 Job 资源

调整作业的每个培训作业定义都必须配置要部署的资源，包括实例类型和计数、托管现场训练和停止条件。

资源配置

每个训练定义都可以具有不同的资源配置。您可以选择实例类型和节点数。

托管的现场训练

如果您在开始和结束时间具有灵活性，则可以通过允许为工作节省计算机成本 SageMaker 以使用备用容量运行作业。有关更多信息，请参阅 Amazon SageMaker 中的托管的 Spot (p. 2085)。

停止条件

停止条件指定了每个训练作业允许的最长持续时间。

添加或克隆训练 Job

为调整作业创建了培训作业定义后，您将返回到培训 Job 定义面板，您可以在其中创建其他培训作业定义来训练其他算法。你可以选择添加训练作业定义然后通过步骤再次确定培训工作或选择Clone来自 的操作菜单来复制现有的训练作业定义并对其进行编辑以适用于新算法。克隆选项可以节省时间，因为它会复制作业的所有设置，包括数据通道和 S3 存储位置。有关克隆的更多信息，请参阅。管理超参数调整和训练作业 (p. 1947)

配置优化 Job 资源

资源限制

您可以指定超参数调整作业可以同时运行的最大并发训练作业数（最多 10 个）和超参数调整作业可以运行的最大训练作业数（最多 500 个）。并 parallel 作业的数量不应超过您请求的节点数涵盖你的所有训练定义。作业总数不能超过您的定义预期运行的作业数。

查看并创建 HPO 调整 Job

查看职位设置、培训作业定义和资源限制。然后选择创建超参数优化作业.

管理超参数调整和训练作业

调整作业可以包含许多培训工作，创建和管理这些作业，它们的定义可能会成为一项复杂而繁重的任务。SageMaker 提供了帮助便于管理这些工作的工具。您运行的调整作业可以从亚马逊访问 SageMaker 控制台在https://console.aws.amazon.com/sagemaker/. Select超参数调整作业来自 的训练菜单来查看列表。此页面也是您开始创建新调整作业的过程的地方，方法是选择创建超参数优化作业.

要查看训练作业运行调整作业的一部分，请从列表中选择其中一个超参数调整作业。调整作业页面上的选项卡允许您检查训练作业、它们的定义、用于调整作业的标签和配置以及调整期间找到的最佳训练作业。您可以选择最佳培训作业或属于调整作业的任何其他培训作业，以查看他们的所有设置。在这里，您可以通过选择以下方法创建一个使用跟踪作业找到的超参数值的模型。创建模型或者你可以通过选择克隆训练作业Clone.

克隆

您可以通过克隆属于超参数调整作业的训练作业来节省时间。克隆会复制作业的所有设置，包括数据通道、S3 存储位置以用于输出构件。如上所述，您可以对已从调整作业页面运行的训练作业执行此操作，也可以在创建超参数调整作业的同时创建其他训练作业定义时执行此操作，如中所述添加或克隆训练 Job (p. 1947)该程序的步骤。

Tagging

自动模型调整在单个家长调整作业中启动多个训练作业，以发现模型超参数的理想权重。可以将标签添加到父调整作业中，如定义作业设置 (p. 1944)然后将这些标签传播到下面的各个培训工作。客户可以将这些标签用于成本分配或访问控制等目的。使用添加标签 SageMaker SDK，使用`AddTags`API。有关将标记用于的更多信息。Amazon请参阅资源标记Amazon资源.

例如：超参数优化 Job

此示例显示了如何创建新的笔记本用于配置和启动超级参数优化作业。优化作业使用 XGBoost 算法 (p. 1731) 来训练模型，预测在通过电话联系客户之后，该客户是否会在银行注册定期存款。

你使用低级别Amazon SDK for Python (Boto)配置和启动超参数优化作业，Amazon Web Services Management Console以监控超参数调整作业的状态。您也可以使用亚马逊 SageMaker 高级别亚马逊 SageMaker Python 开发工具包配置、运行、监控和分析超参数优化作业。有关更多信息，请参阅 https://github.com/aws/sagemaker-python-sdk。

先决条件

要运行此示例中的代码，您需要

- Amazon 账户和管理员用户 (p. 31)
- Amazon S3 存储桶，用于存储您的训练数据集以及在训练期间创建的模型构件
- 正在运行 SageMaker 笔记本实例 (p. 59)

主题

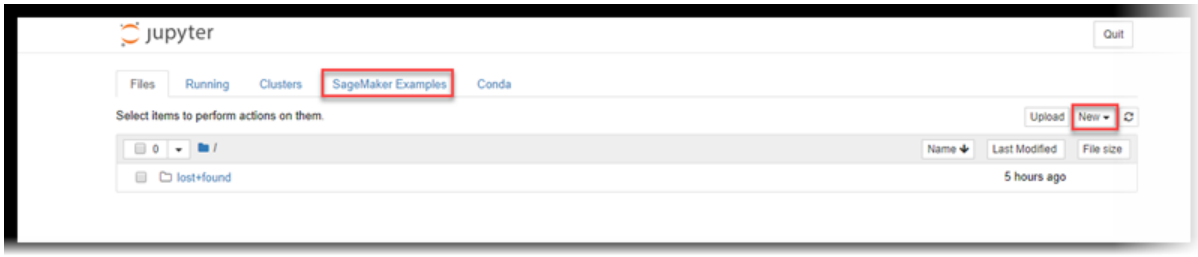
- 创建笔记本 (p. 1948)
- 获取 Amazon SageMaker Boto 3 客户 (p. 1948)
- Get SageMaker 执行角色 (p. 1949)
- 指定 S3 存储桶以上传训练数据集和存储输出数据 (p. 1949)
- 下载、准备和上传训练数据 (p. 1950)
- 配置并启动超参数优化作业 (p. 1950)
- 监控超参数优化作业的进度 (p. 1954)
- 清除 (p. 1957)

创建笔记本

创建一个包含预安装使用环境默认的 Anaconda 安装和 Python 3.

创建 Jupyter 笔记本

1. 打开 Amazon SageMaker 控制台在https://console.aws.amazon.com/sagemaker/.
2. 要打开一个运行的笔记本实例，请选择其名称旁的 Open (打开)。此时显示 Jupyter 笔记本服务器页面：



3. 要创建笔记本，请选择 Files (文件)、New (新建) 和 conda_python3。
4. 命名笔记本。

下一个步骤

获取 Amazon SageMaker Boto 3 客户 (p. 1948)

获取 Amazon SageMaker Boto 3 客户

导入亚马逊 SageMaker Python 开发工具包，Amazon SDK for Python (Boto3)，以及其他 Python 库。在新的 Jupyter 笔记本中，将以下代码粘贴到第一个单元格中：

```
import sagemaker
import boto3
```

```
import numpy as np                                # For performing matrix operations and
 numerical processing
import pandas as pd                               # For manipulating tabular data
from time import gmtime, strftime
import os

region = boto3.Session().region_name
smclient = boto3.Session().client('sagemaker')
```

前面的代码单元格定义了`region`和`smclient`你将用来调用内置 xgBoost 算法并设置 SageMaker 超参数优化作业。

下一个步骤

Get SageMaker 执行角色 (p. 1949)

Get SageMaker 执行角色

获取笔记本实例的执行角色。这是您为笔记本实例创建的 IAM 角色。您传递该角色到优化作业。

```
from sagemaker import get_execution_role

role = get_execution_role()
print(role)
```

下一个步骤

指定 S3 存储桶以上传训练数据集和存储输出数据 (p. 1949)

指定 S3 存储桶以上传训练数据集和存储输出数据

设置 S3 存储桶以上传训练数据集并保存训练输出数据。

使用默认 S3 存储桶

使用以下代码指定分配给您的默认 S3 存储桶 SageMaker 会话。`prefix`是存储桶中的路径 SageMaker 存储当前培训作业的数据。

```
sess = sagemaker.Session()
bucket = sess.default_bucket()                     # Set a default S3 bucket
prefix = 'DEMO-automatic-model-tuning-xgboost-dm'
```

(可选) 使用特定 S3 存储桶

如果您想使用特定的 S3 存储桶，请使用以下代码并将字符串替换为 S3 存储桶的确切名称。存储桶的名称必须包含 **`sagemaker`**，并且全局唯一。存储桶必须在同一位置Amazon区域作为此示例使用的笔记本实例。

```
bucket = "sagemaker-your-preferred-s3-bucket"
```

> **Note**
>
> 存储桶的名称不需要包含**`sagemaker`**如果您用于运行超参数优化作业的 IAM 角色具有向`S3FullAccess`权限。

下一个步骤

下载、准备和上传训练数据 (p. 1950)

下载、准备和上传训练数据

在本示例中，您使用有关银行客户信息的训练数据集，其中包含客户的工作、婚姻状况以及在银行的直接营销市场活动中通过何种方式联系他们。要将数据集用于超参数优化作业，请下载该数据集，转换数据，然后将其上载到 Amazon S3 存储桶。

有关数据集和示例执行的数据转换的详细信息，请参阅HPO_XGBOOST_Direct_Marketing_SageMaker_API中的笔记本超参数优化的 部分SageMaker 示例选项卡在笔记本实例中。

下载和探索训练数据集

要下载和探索数据集，请在笔记本中运行以下代码：

```
!wget -N https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-
additional.zip
!unzip -o bank-additional.zip
data = pd.read_csv('./bank-additional/bank-additional-full.csv', sep=';')
pd.set_option('display.max_columns', 500)     # Make sure we can see all of the columns
pd.set_option('display.max_rows', 5)         # Keep the output on one page
data
```

准备和上传数据

在创建超级参数优化作业之前，请准备数据并将其上传到超级参数优化作业可以访问的 S3 存储桶。

在笔记本中运行以下代码：

```
data['no_previous_contact'] = np.where(data['pdays'] == 999, 1, 0)
        # Indicator variable to capture when pdays takes a value of 999
data['not_working'] = np.where(np.in1d(data['job'], ['student', 'retired', 'unemployed']),
 1, 0)   # Indicator for individuals not actively employed
model_data = pd.get_dummies(data)
        # Convert categorical variables to sets of indicators
model_data
model_data = model_data.drop(['duration', 'emp.var.rate', 'cons.price.idx',
 'cons.conf.idx', 'euribor3m', 'nr.employed'], axis=1)

train_data, validation_data, test_data = np.split(model_data.sample(frac=1,
 random_state=1729), [int(0.7 * len(model_data)), int(0.9*len(model_data))])

pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'], axis=1)],
 axis=1).to_csv('train.csv', index=False, header=False)
pd.concat([validation_data['y_yes'], validation_data.drop(['y_no', 'y_yes'], axis=1)],
 axis=1).to_csv('validation.csv', index=False, header=False)
pd.concat([test_data['y_yes'], test_data.drop(['y_no', 'y_yes'], axis=1)],
 axis=1).to_csv('test.csv', index=False, header=False)

boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/
train.csv')).upload_file('train.csv')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'validation/
validation.csv')).upload_file('validation.csv')
```

下一个步骤

配置并启动超参数优化作业 (p. 1950)

配置并启动超参数优化作业

要配置和启动超级参数优化作业，请完成以下步骤。

主题

- 指定超参数优化作业设置 (p. 1951)
- 配置训练作业 (p. 1952)
- 命名并启动超参数优化作业 (p. 1954)
- 下一个步骤 (p. 1954)

指定超参数优化作业设置

要指定超参数优化作业的设置，您需要定义 JSON 对象。创建优化作业时，将对象作为 `HyperParameterTuningJobConfig` 参数的值传递给 `CreateHyperParameterTuningJob`。

在此 JSON 对象中，您将指定：

- 要优化的超参数的范围。有关更多信息，请参阅 。定义超参数范围 (p. 1942)
- 超参数优化作业可以使用的资源限制。
- 超参数优化作业的目标指标。目标指标是超级参数优化作业用于评估所启动训练作业的指标。

  Note

  要使用自己的算法来实现超参数优化，您需要为您的算法定义指标。有关信息，请参阅定义指标 (p. 1941)。

超级参数优化作业定义 XGBoost 算法 (p. 1731) 内置算法的 `eta`、`alpha`、`min_child_weight` 和 `max_depth` 超级参数的范围。超参数优化作业的目标指标最大化这`validation:auc`算法发送到的指标 CloudWatch 日志。

```
tuning_job_config = {
    "ParameterRanges": {
      "CategoricalParameterRanges": [],
      "ContinuousParameterRanges": [
        {
          "MaxValue": "1",
          "MinValue": "0",
          "Name": "eta"
        },
        {
          "MaxValue": "2",
          "MinValue": "0",
          "Name": "alpha"
        },
        {
          "MaxValue": "10",
          "MinValue": "1",
          "Name": "min_child_weight"
        }
      ],
      "IntegerParameterRanges": [
        {
          "MaxValue": "10",
          "MinValue": "1",
          "Name": "max_depth"
        }
      ]
    },
    "ResourceLimits": {
      "MaxNumberOfTrainingJobs": 20,
      "MaxParallelTrainingJobs": 3
    },
    "Strategy": "Bayesian",
```

```
  "HyperParameterTuningJobObjective": {
    "MetricName": "validation:auc",
    "Type": "Maximize"
  }
}
```

配置训练作业

要配置优化作业启动的训练作业，请定义作为 `CreateHyperParameterTuningJob` 调用的 `TrainingJobDefinition` 参数值传递的 JSON 对象。

在此 JSON 对象中，您将指定：

- 可选-训练作业发出的指标。

  Note

  只在使用自定义训练算法时定义指标。因为此示例使用内置算法，您无需指定指标。有关定义指标的信息，请参阅定义指标 (p. 1941)。
- 指定训练算法的容器镜像。
- 您的训练和测试数据的输入配置。
- 算法输出的存储位置。指定存储训练作业输出的 S3 存储桶。
- 未在优化作业中进行优化的算法超级参数的值。
- 用于训练作业的实例的类型。
- 训练作业的停止条件。这是各个训练作业的最长持续时间。

在本示例中，我们将设置 XGBoost 算法 (p. 1731) 内置算法的 `eval_metric`、`num_round`、`objective`、`rate_drop` 和 `tweedie_variance_power` 参数的静态值。

SageMaker Python SDK v1

```
from sagemaker.amazon.amazon_estimator import get_image_uri
training_image = get_image_uri(region, 'xgboost', repo_version='1.0-1')

s3_input_train = 's3://{}/{}/train'.format(bucket, prefix)
s3_input_validation ='s3://{}/{}/validation/'.format(bucket, prefix)

training_job_definition = {
    "AlgorithmSpecification": {
      "TrainingImage": training_image,
      "TrainingInputMode": "File"
    },
    "InputDataConfig": [
      {
        "ChannelName": "train",
        "CompressionType": "None",
        "ContentType": "csv",
        "DataSource": {
          "S3DataSource": {
            "S3DataDistributionType": "FullyReplicated",
            "S3DataType": "S3Prefix",
            "S3Uri": s3_input_train
          }
        }
      },
      {
        "ChannelName": "validation",
        "CompressionType": "None",
```

```
        "ContentType": "csv",
        "DataSource": {
          "S3DataSource": {
            "S3DataDistributionType": "FullyReplicated",
            "S3DataType": "S3Prefix",
            "S3Uri": s3_input_validation
          }
        }
      }
    ],
    "OutputDataConfig": {
      "S3OutputPath": "s3://{}/{}/output".format(bucket,prefix)
    },
    "ResourceConfig": {
      "InstanceCount": 2,
      "InstanceType": "ml.c4.2xlarge",
      "VolumeSizeInGB": 10
    },
    "RoleArn": role,
    "StaticHyperParameters": {
      "eval_metric": "auc",
      "num_round": "100",
      "objective": "binary:logistic",
      "rate_drop": "0.3",
      "tweedie_variance_power": "1.4"
    },
    "StoppingCondition": {
      "MaxRuntimeInSeconds": 43200
    }
}
```

SageMaker Python SDK v2

```
training_image = sagemaker.image_uris.retrieve('xgboost', region, '1.0-1')

s3_input_train = 's3://{}/{}/train'.format(bucket, prefix)
s3_input_validation ='s3://{}/{}/validation/'.format(bucket, prefix)

training_job_definition = {
    "AlgorithmSpecification": {
      "TrainingImage": training_image,
      "TrainingInputMode": "File"
    },
    "InputDataConfig": [
      {
        "ChannelName": "train",
        "CompressionType": "None",
        "ContentType": "csv",
        "DataSource": {
          "S3DataSource": {
            "S3DataDistributionType": "FullyReplicated",
            "S3DataType": "S3Prefix",
            "S3Uri": s3_input_train
          }
        }
      },
      {
        "ChannelName": "validation",
        "CompressionType": "None",
        "ContentType": "csv",
        "DataSource": {
          "S3DataSource": {
            "S3DataDistributionType": "FullyReplicated",
            "S3DataType": "S3Prefix",
            "S3Uri": s3_input_validation
```

```
            }
          }
        }
    ],
    "OutputDataConfig": {
      "S3OutputPath": "s3://{}/{}/output".format(bucket,prefix)
    },
    "ResourceConfig": {
      "InstanceCount": 2,
      "InstanceType": "ml.c4.2xlarge",
      "VolumeSizeInGB": 10
    },
    "RoleArn": role,
    "StaticHyperParameters": {
      "eval_metric": "auc",
      "num_round": "100",
      "objective": "binary:logistic",
      "rate_drop": "0.3",
      "tweedie_variance_power": "1.4"
    },
    "StoppingCondition": {
      "MaxRuntimeInSeconds": 43200
    }
}
```

命名并启动超参数优化作业

现在，您可为超参数优化作业提供名称，然后通过调用 `CreateHyperParameterTuningJob` API 来启动它。传递您在之前步骤中创建的 `tuning_job_config` 和 `training_job_definition` 作为参数的值。

```
tuning_job_name = "MyTuningJob"
smclient.create_hyper_parameter_tuning_job(HyperParameterTuningJobName = tuning_job_name,
                                           HyperParameterTuningJobConfig =
 tuning_job_config,
                                           TrainingJobDefinition = training_job_definition)
```

下一个步骤

监控超参数优化作业的进度 (p. 1954)

监控超参数优化作业的进度

要监控超参数优化作业的进度以及它启动的训练作业，请使用亚马逊 SageMaker 控制台。

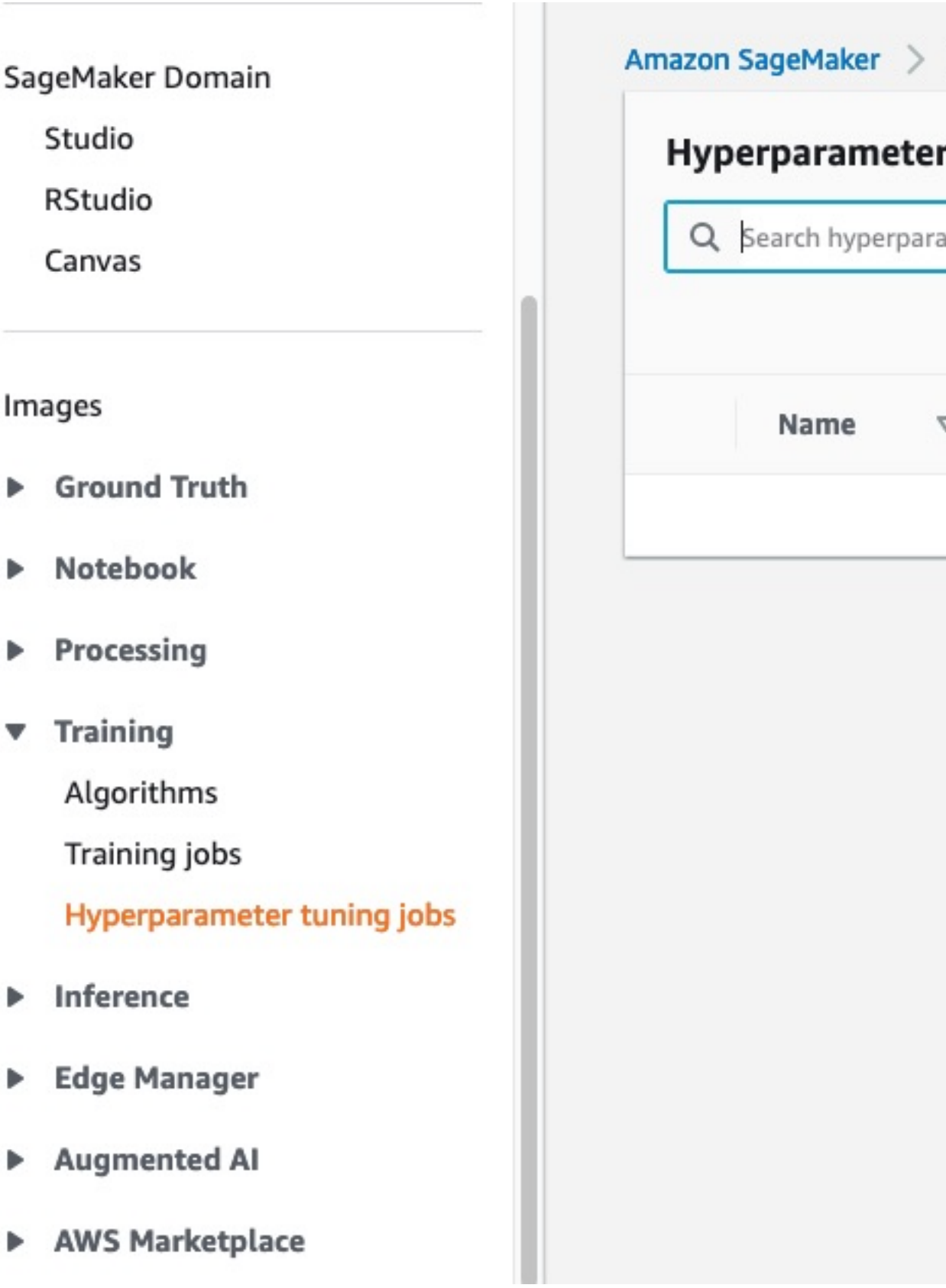
主题

- 查看超参数优化作业的状态 (p. 1954)
- 查看训练作业的状态 (p. 1956)
- 查看最佳训练作业 (p. 1956)

查看超参数优化作业的状态

查看超参数优化作业的状态

1. 打开 Amazon SageMaker 控制台在https://console.aws.amazon.com/sagemaker/.
2. 选择 Hyperparameter tuning jobs (超参数优化作业)。


SageMaker Domain
Studio
RStudio
Canvas
Images
Ground Truth
Notebook
Processing
Training
Algorithms
Training jobs
Hyperparameter tuning jobs
Inference
Edge Manager
Augmented AI
AWS Marketplace
Amazon SageMaker
Hyperparameter
Search hyperpara
Name

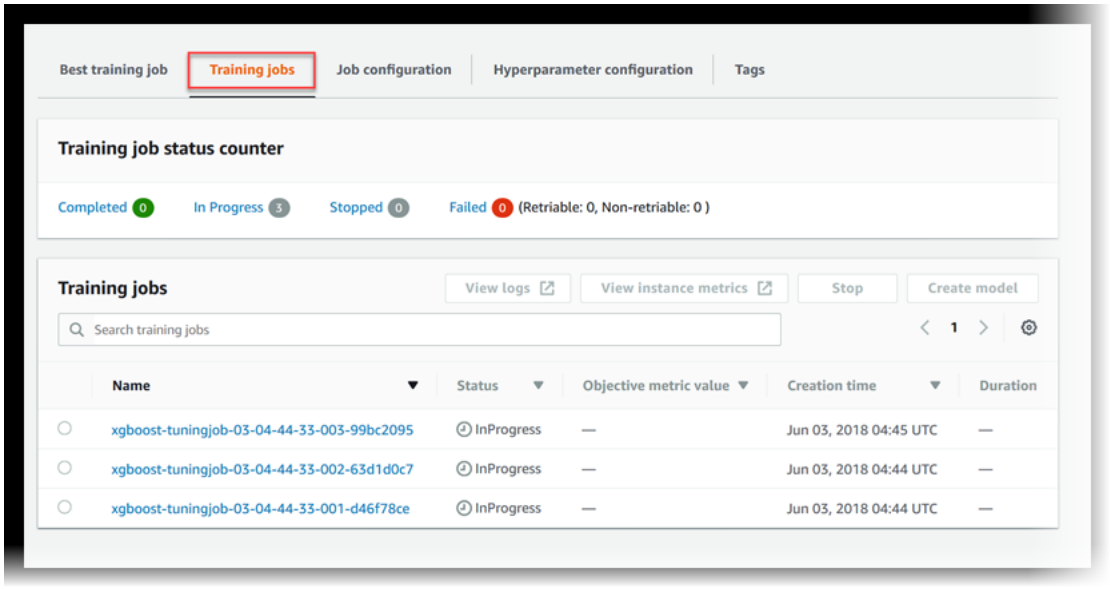
3. 在超参数优化作业列表中，检查您启动的超参数优化作业的状态。优化作业可以处于：

   - `Completed`超参数优化作业成功完成。
   - `InProgress`超参数优化作业正在进行。一个或多个训练作业仍在运行。
   - `Failed`超参数优化作业失败。
   - `Stopped`超参数优化作业在完成之前被手动停止。超参数优化作业启动的所有训练作业已停止。
   - `Stopping`超参数优化作业正在停止。

查看训练作业的状态

查看超参数优化作业启动的训练作业的状态

1. 在超参数优化作业列表中，选择您启动的作业。
2. 选择 Training jobs (训练作业)。



3. 查看各个训练作业的状态。要查看有关作业的详细信息，请在训练作业列表中选择该作业。要查看超参数优化作业启动的所有训练作业的状态摘要，请查看 Training job status counter (训练作业状态计数器)。

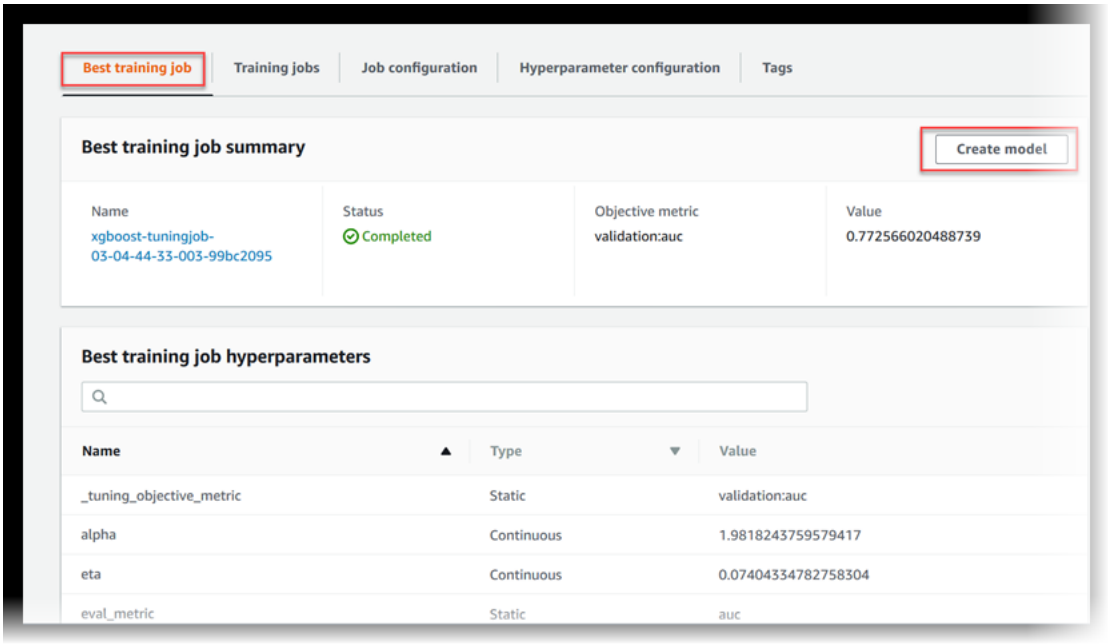
   训练作业的状态可以为：

   - `Completed`— 训练作业已成功完成。
   - `InProgress`— 训练作业正在进行。
   - `Stopped`训练作业在完成之前被手动停止。
   - `Failed (Retryable)`— 训练作业失败，但可以重试。只有在训练作业由于出现内部服务错误而失败时，才能重试该训练作业。
   - `Failed (Non-retryable)`— 训练作业失败，并且无法重试。在出现客户端错误时，无法重试失败的训练作业。

查看最佳训练作业

超参数优化作业使用各个训练作业返回的目标指标来评估训练作业。在超参数优化作业进行中时，最佳训练作业是迄今为止返回了最佳目标指标的作业。超参数优化作业完成后，最佳训练作业是返回了最佳目标指标的作业。

要查看最佳训练作业，请选择 Best training job (最佳训练作业)。



要将最佳训练作业部署为模型，您可以在 SageMaker 终端节点，选择创建模型.

下一个步骤

清除 (p. 1957)

清除

为避免产生不必要的费用，在完成示例之后，请使用 Amazon Web Services Management Console删除您为其创建的资源。

> Note
>
> 如果您打算探索其他示例，您可能需要保留其中一些资源，例如笔记本实例、S3 存储桶和 IAM 角色。

1. 打开 SageMaker 控制台在https://console.aws.amazon.com/sagemaker/然后删除笔记本实例。先停止实例，然后再将其删除。
2. 在以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/并删除您为存储模型构件和训练数据集创建的存储桶。
3. 在以下网址打开 IAM 控制台https://console.aws.amazon.com/iam/然后删除 IAM 角色。如果您已创建权限策略，也可以将其删除。
4. 打开 Amazon CloudWatch 控制台在https://console.aws.amazon.com/cloudwatch/并删除所有名称以开头的日志组`/aws/sagemaker/`.

提前停止训练作业

在按照目标指标进行衡量时，如果训练作业未能明显改进，则可提前停止超参数优化作业启动的训练作业。提前停止训练作业有助于减少计算时间，并帮助您避免过度拟合模型。要配置超参数调整作业以尽早停止训练作业，请执行以下操作之一：

- 如果您使用的是适用于 Python (Boto 3) 的 Amazon 开发工具包，请设置用于将调整作业配置为 `AUTO` 的 `HyperParameterTuningJobConfig` 对象的 `TrainingJobEarlyStoppingType` 字段。

- 如果您使用亚马逊 SageMaker Python 开发工具包中，设置`early_stopping_type`参数HyperParameterTuner反对`Auto`.
- 在 Amazon 中 SageMaker 在控制台中，创建超参数优化作业下工作流程，提前停止，选择汽车.

有关演示如何使用提前停止的示例笔记本，请参阅https://github.com/awslabs/amazon-sagemaker-examples/blob/master/hyperparameter_tuning/image_classification_early_stopping/hpo_image_classification_early_stopping.ipynb或者打开`hpo_image_classification_early_stopping.ipynb`中的笔记本超参数优化的 部分SageMaker 示例在笔记本实例中。有关在笔记本实例中使用示例笔记本的信息，请参阅示例笔记本 (p. 158)。

提前停止的工作原理

在您为超参数优化作业启用提前停止时， SageMaker 如下所示评估超参数优化作业启动的每个训练作业：

- 在每个训练周期结束后，获取对象指标的值。
- 计算直至当前周期的所有之前训练作业的目标指标运行平均值，然后计算所有运行平均值的中值。
- 如果当前训练作业的目标指标值（在最小化目标指标时更高，或者在最大化目标指标时更低）比直至同一时期之前训练作业的目标指标运行平均值的中值更糟糕， SageMaker 停止当前的训练作业。

支持提前停止的算法

要支持提前停止，算法必须为每个周期发出目标指标。内置以下内置 SageMaker 算法支持提前停止：

- 线性学习器算法 (p. 1657)— 仅在您使用时支持`objective_loss`作为目标指标。
- XGBoost 算法 (p. 1731)
- 图像分类算法 (p. 1620)
- 对象检测算法 (p. 1691)
- 序列到序列算法 (p. 1720)
- IP 见解 (p. 1630)

> **Note**
>
> 当前支持提前停止的内置算法列表截止到 2018 年 12 月 13 日。之后其他内置算法可能会支持提前停止。如果算法发出的指标可用作超参数优化作业的目标指标（最好是验证指标），则它支持提前停止。

要在您自己的算法中使用提前停止，您编写的算法必须在每个周期后发出目标指标的值。以下列表演示了如何在不同框架中实现这一点：

TensorFlow

使用 `tf.keras.callbacks.ProgbarLogger` 类。有关更多信息，请参阅tf.keras.callbacks.probarlogger API.

MXNet

使用 `mxnet.callback.LogValidationMetricsCallback`。有关更多信息，请参阅mxnet.callback API.

Chainer

使用 `extensions.Evaluator` 类扩展 chainer。有关更多信息，请参阅链人。培训。扩展。评估器 API.

PyTorch 和 Spark

没有高级支持。您必须明确编写训练代码，以便其计算目标指标并在每个周期之后写入日志。

运行热启动超参数优化作业

通过热启动，启动使用之前一个或多个优化作业作为起点的超参数优化作业。以前优化作业的结果用于告知在新优化作业中搜索哪些超参数组合。超参数优化使用贝叶斯或随机来从您指定的范围中选择超参数值组合。有关更多信息，请参阅 超参数优化的工作方式 (p. 1940)。使用来自以前超参数优化作业中的信息可以更高效地搜索最佳超参数组合，有助于提升新超参数优化作业的性能。

Note

热启动优化作业的启动时间通常要长于标准超参数优化作业，因为需要先加载来自父作业的结果，然后才能启动作业。增加的时间取决于父作业启动的训练作业总数。

考虑使用热启动的原因有以下几点：

- 您希望根据在每次迭代之后看到的结果，逐渐增加几个优化作业的训练作业数。
- 您获取了新数据，并希望使用新数据优化模型。
- 您希望更改在以前优化作业中使用的超参数范围，将静态超参数更改为可优化，或者将可优化超参数更改为静态值。
- 您提前停止了之前的超参数作业或者作业意外停止。

主题

- 热启动优化作业的类型 (p. 1959)
- 热启动优化的限制 (p. 1960)
- 热启动优化示例笔记本 (p. 1960)
- 创建热启动优化作业 (p. 1960)

热启动优化作业的类型

有两种不同类型的热启动优化作业：

`IDENTICAL_DATA_AND_ALGORITHM`

新超参数优化作业使用与父优化作业相同的输入数据和训练镜像。您可更改要搜索的超参数范围以及超参数优化作业启动的最大训练作业数。您还可以将超参数从可优化更改为静态值以及从静态值更改为可优化，但静态与可优化超参数的总数必须与所有父作业中的超参数总数保持相同。您不能使用新版本的训练算法，除非新版本中的更改不影响算法本身。例如，允许进行改进日志记录或添加对不同数据格式支持的更改。

在您使用与之前超参数优化作业中相同的训练数据时，使用相同的数据和算法，但您希望增加训练作业总数或者更改超参数的范围或值。

在您运行 `IDENTICAL_DATA_AND_ALGORITHM` 类型的热启动优化作业时，在对 `DescribeHyperParameterTuningJob` 的响应中有名为 `OverallBestTrainingJob` 的其他字段。在此优化作业启动的所有训练作业以及为热启动训练作业指定的所有父作业中，对于具有最佳目标指标值的训练作业，此字段的值为该训练作业的 TrainingJobSummary。

`TRANSFER_LEARNING`

新超参数优化作业可以包括输入数据、超参数范围、并发训练作业最大数以及与其父超参数优化作业中训练作业不同的训练作业最大数。您还可以将超参数从可优化更改为静态值以及从静态值更改为可优

化，但静态与可优化超参数的总数必须与所有父作业中的超参数总数保持相同。该训练算法镜像也可以使用与父超参数优化作业中所用版本不同的版本。在您使用迁移学习时，对数据集或算法进行的更改，如果会显著影响目标指标值，则可能会减少使用热启动优化的有效性。

热启动优化的限制

以下限制适用于所有热启动优化作业：

- 一个优化作业最多可以有 5 个父作业，并且所有父作业必须处于最终状态（`Completed`、`Stopped` 或 `Failed`）才能启动新优化作业。
- 新优化作业中使用的目标指标必须与在父作业中使用的目标指标相同。
- 静态与可优化超参数的总数必须与父作业以及新优化作业中的超参数总数保持相同。因此，如果您认为自己需要在未来的热启动优化作业中使用可优化的超参数，应在创建优化作业时将其作为静态超参数添加。
- 在父作业与新优化作业之间，不能更改各个超参数的类型（连续、整数、分类）。
- 在新优化作业中，将父作业中的可优化超参数更改为静态超参数的更改总数，加上静态超参数值的更改总数，不能超过 10 个。分类超参数中的每个值都计入这一限制中。例如，如果父作业有一个可优化分类超参数，其可能值为 `red` 和 `blue`，您在新优化作业中将该超参数更改为静态，则会在允许的 10 次更改总数中计入 2 次更改。如果同一个超参数在父作业中具有静态值 `red`，而您在新优化作业中将静态值更改为 `blue`，这也计入 2 次更改。
- 热启动优化不递归。如果，如果您创建 `MyTuningJob3` 作为热启动优化作业，其父作业为 `MyTuningJob2`，并且 `MyTuningJob2` 本身是具有父作业 `MyTuningJob1` 的热启动优化作业，则在运行 `MyTuningJob1` 时学到的信息不用于 `MyTuningJob3`。如果您希望使用来自 `MyTuningJob1` 的信息，则必须明确将其添加作为 `MyTuningJob3` 的父作业。
- 每个父作业在热启动优化作业中启动的训练作业，计入一个优化作业的最大 500 个训练作业总数中。
- 在 2018 年 10 月 1 日之前创建超参数优化作业不能用作热启动优化作业的父作业。

热启动优化示例笔记本

有关展示如何使用热启动优化的示例笔记本，请参阅 https://github.com/awslabs/amazon-sagemaker-examples/blob/master/hyperparameter_tuning/image_classification_warmstart/hpo_image_classification_warmstart.ipynb。有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅示例笔记本 (p. 158). 创建笔记本实例并将其打开后，选择SageMaker 示例选项卡以查看所有 SageMaker 示例。热启动优化示例笔记本位于 Hyperparameter tuning (超参数优化) 部分，名为 `hpo_image_classification_warmstart.ipynb`。要打开笔记本，请单击其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

创建热启动优化作业

您可以使用低级别Amazon适用于 Python 的开发工具包 (Boto 3) 或者高级 SageMaker Python 开发工具包来创建热启动优化作业。

主题

- 创建热启动优化 Job（低级） SageMaker 适用于 Python 的 API (Boto 3) (p. 1960)
- 创建热启动优化 Job（SageMaker Python 开发工具包） (p. 1961)

创建热启动优化 Job（低级） SageMaker 适用于 Python 的 API (Boto 3)

要使用热启动优化，您需要指定 `HyperParameterTuningJobWarmStartConfig` 对象的值，将其在对 `CreateHyperParameterTuningJob` 的调用中作为 `WarmStartConfig` 字段传递。

以下代码显示了如何创建`HyperParameterTuningJobWarmStartConfig`对象并将该项传递给`CreateHyperParameterTuningJob`通过使用低级别作业 SageMaker 适用于 Python 的 API (Boto 3)。

创建 HyperParameterTuningJobWarmStartConfig 对象：

```
warm_start_config = {
          "ParentHyperParameterTuningJobs" : [
          {"HyperParameterTuningJobName" : 'MyParentTuningJob'}
          ],
          "WarmStartType" : "IdenticalDataAndAlgorithm"
}
```

创建热启动优化作业：

```
smclient = boto3.Session().client('sagemaker')
smclient.create_hyper_parameter_tuning_job(HyperParameterTuningJobName =
 'MyWarmStartTuningJob',
   HyperParameterTuningJobConfig = tuning_job_config, # See notebook for tuning
 configuration
   TrainingJobDefinition = training_job_definition, # See notebook for job definition
   WarmStartConfig = warm_start_config)
```

创建热启动优化 Job（SageMaker Python 开发工具包）

使用亚马逊 SageMaker Python 开发工具包要运行热启动优化作业，您需要：

- 使用 WarmStartConfig 对象指定父作业和热启动类型。
- 传递WarmStartConfig对象的值warm_start_config参数HyperparameterTuner对象。
- 调用 HyperparameterTuner 对象的 fit 方法。

有关使用的更多信息亚马逊 SageMaker Python 开发工具包有关超参数调整，请参阅https://github.com/aws/sagemaker-python-sdk#sagemaker-automatic-model-tuning.

此示例使用利用 图像分类算法 (p. 1620) 算法进行训练的评估程序。以下代码设置热启动优化作业进行搜索的超参数范围，以查找值的最佳组合。有关设置超参数范围的信息，请参阅定义超参数范围 (p. 1942)。

```
hyperparameter_ranges = {'learning_rate': ContinuousParameter(0.0, 0.1),
                         'momentum': ContinuousParameter(0.0, 0.99)}
```

以下代码通过创建 WarmStartConfig 对象来配置热启动优化作业。

```
from sagemaker.tuner import WarmStartConfig,WarmStartTypes

parent_tuning_job_name = "MyParentTuningJob"
warm_start_config =
 WarmStartConfig(warm_start_type=WarmStartTypes.IDENTICAL_DATA_AND_ALGORITHM,
 parents={parent_tuning_job_name})
```

现在设置静态超参数的值，这是超参数为热启动优化作业所启动的每个训练作业保留的相同值。在以下代码中，imageclassification 是之前创建的评估程序。

```
imageclassification.set_hyperparameters(num_layers=18,
                                        image_shape='3,224,224',
                                        num_classes=257,
                                        num_training_samples=15420,
                                        mini_batch_size=128,
                                        epochs=30,
                                        optimizer='sgd',
```

```
                                        top_k='2',
                                        precision_dtype='float32',
                                        augmentation_type='crop')
```

现在创建 HyperparameterTuner 对象并将您之前创建的 WarmStartConfig 对象作为 warm_start_config 参数传递。

```
tuner_warm_start = HyperparameterTuner(imageclassification,
                            'validation:accuracy',
                            hyperparameter_ranges,
                            objective_type='Maximize',
                            max_jobs=10,
                            max_parallel_jobs=2,
                            base_tuning_job_name='warmstart',
                            warm_start_config=warm_start_config)
```

最后，调用 HyperparameterTuner 对象的 fit 方法来启动热启动优化作业。

```
tuner_warm_start.fit(
        {'train': s3_input_train, 'validation': s3_input_validation},
        include_cls_metadata=False)
```

自动模型优化的资源限制

SageMaker 为自动模型调整所使用的资源设置以下默认限制：

- 并 parallel（并发）超参数调整作业的数量：100
- 可搜索的超参数：20

  Note

  分类超参数中的每个可能值都计入这一限制中。
- 每个超级参数优化作业定义的指标数：20
- 每个超参数优化作业的并发（并发）训练作业数：10

  Note

  这可以增加到一百个。
- [贝叶斯搜索策略] 每个超参数优化作业的训练作业数：500
- [随机搜索策略] 每个超参数优化作业的训练作业数：500

  Note

  这可以增加到一万。
- 超参数优化作业的最长运行时间：30 天

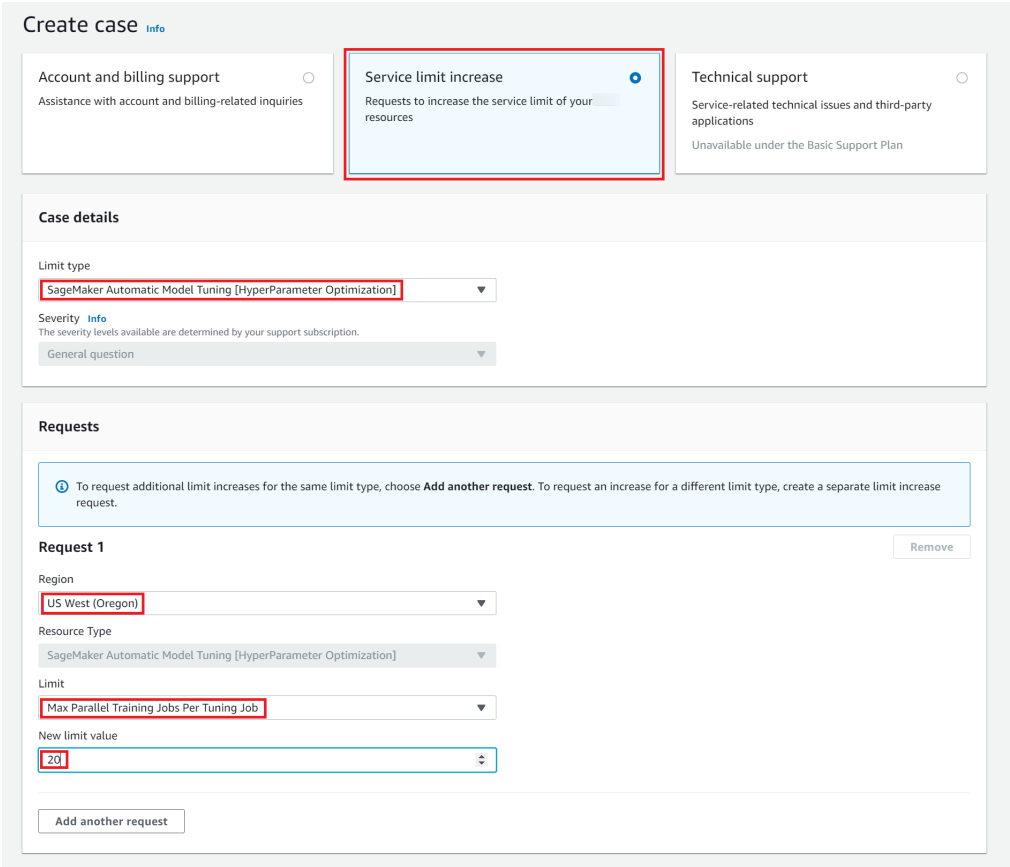
在计划超参数优化作业时，还必须考虑到训练资源限制。有关以下默认资源限制的信息： SageMaker 请参阅训练作业SageMaker 限制. 所有超参数优化作业在其上运行的每个并发训练实例都计入所允许的训练实例总数中。例如，如果您运行 10 个并发超参数优化作业，则每个超参数优化作业又各运行 100 个总训练作业和 20 个并发训练作业。每个训练作业在一个 ml.m4.xlarge 实例上运行。那么，下列限制适用：

- 并发超参数优化作业数：您不需要放宽限制，因为 10 个优化作业低于 100 这一限制。
- 每个超级参数优化作业的训练作业数：您不需要放宽限制，因为 100 个训练作业低于 500 这一限制。
- 每个超级参数优化作业的并发训练作业数：您需要请求将限制提高到 20，因为默认限制为 10。

- SageMaker 训练ml.m4.xlarge实例：您需要请求将限制放宽到 200，因为您有 10 个超参数优化作业，每个超参数优化作业又各运行 20 个并发训练作业。默认限制为 20 个实例。
- SageMaker 训练的实例总数：您需要请求将限制放宽到 200，因为您有 10 个超参数优化作业，每个超参数优化作业又各运行 20 个并发训练作业。默认限制为 20 个实例。

要请求提高配额：

1. 打开Amazon支持中心页面，登录（如有必要），然后选择创建案例.
2. 在 Create case (创建案例) 页面上，选择 Service limit increase (增加服务限制)。
3. 在存储库的案例细节面板中，选择SageMaker 自动模型优化 [超参数优化](对于)限制类型
4. 在存储库的请求面板请求 1中，选择区域，该资源限制要增加和新限制值你正在请求。Select添加另一个请求如果你有额外的增加配额请求。



5. 在案例描述面板中，请提供您的使用案例的描述。
6. 在联系选项面板中，选择您的首选联系方式(Web、聊天要么Phone) 然后选择Submit.

超参数优化的最佳实践

超参数优化不是完全自动化的过程。要改进优化，请在创建超参数时使用以下指南。

主题

- 选择超参数的数量 (p. 1964)
- 选择超参数范围 (p. 1964)
- 为超参数使用对数标度 (p. 1964)

- 选择最佳并发训练作业数 (p. 1964)
- 在多个实例上运行训练作业 (p. 1964)

选择超参数的数量

超参数优化作业的计算复杂性主要取决于超参数的数量，其值范围为 Amazon SageMaker 在优化过程中必须搜索。虽然您可以同时指定最多 20 个超参数以针对优化作业进行优化，但将搜索限制在较少的数量可能会得到更好的结果。

选择超参数范围

您选择要搜索的超参数值范围会显著影响超参数优化成功与否。虽然您可能需要指定非常大的范围来涵盖超参数的每个可能的值，但通过将搜索限制在小范围内，您会得到更好的结果。如果您知道您在可能范围的子集中获得了最佳指标值，请考虑将范围限制为该子集。

为超参数使用对数标度

在超参数调整期间， SageMaker 尝试确定您的超参数是对数标度还是线性标度。首先，它假定超参数为线性标度。如果它们实际上是日志缩放的，则可能需要一些时间 SageMaker 来发现这个事实。如果您知道超参数是对数缩放并可以自行转换，这样做可以改进超参数优化。

选择最佳并发训练作业数

设置资源限制时`MaxParallelTrainingJobs`要查看超参数优化作业可启动的并发训练作业的最大数目，请考虑以下权衡。同时运行多个超参数优化作业可以更快完成更多工作，但训练作业只有通过连续轮次的实验才能改进。通常，一次运行一个训练作业可以通过最少的计算时间得到最佳结果。

在多个实例上运行训练作业

当训练作业在多个实例上运行时，超参数优化使用来自该训练作业所有实例的最新报告的目标指标值作为该训练作业的目标指标的值。设计分布式训练作业，这样，所报告的目标指标就是所需的指标。

Amazon SageMaker 分布式培训库

SageMaker 提供了数据并行性和模型并行性的分布式培训库。这些库针对 SageMaker 培训环境进行了优化，有助于使您的分布式培训作业适应 SageMaker，并提高培训速度和吞吐量。

主题

- 开始使用分布式培训 (p. 1965)
- 基本分布式训练概念 (p. 1965)
- 高级概念 (p. 1966)
- 策略 (p. 1967)
- 优化分布式训练 (p. 1968)
- 方案 (p. 1969)
- SageMaker 内置分布式培训功能 (p. 1971)
- SageMaker 的分布式数据并行库 (p. 1971)
- SageMaker的分布式模型并行 (p. 1998)
- 亚马逊 SageMaker 分布式训练记本示例 (p. 2043)

开始使用分布式培训

如果你熟悉分布式培训，请按照指向首选策略或框架的链接之一开始使用。否则，请继续下一节以了解一些分布式培训概念。

SageMaker 分布式培训库：

- SageMaker 的分布式数据并行库 (p. 1971)
- SageMaker的分布式模型并行 (p. 1998)

> Note
>
> SageMaker 分布式培训库只能通过AmazonSageMaker 培训平台中的 TensorFlow、PyTorch 和 HuggingFace 框架的深度学习容器。要使用这些库，您必须使用 SageMaker Python SDK 或者通过适用 SDK for Python (Boto3) 使用 SageMaker API 或Amazon Command Line Interface. 在整个文档中，说明和示例都重点介绍了如何将分布式培训库与 SageMaker Python SDK 结合使用。

基本分布式训练概念

SageMaker 的分布式培训库使用以下分布式培训术语和功能。

批处理数据集和

- 训练数据集：训练模型时使用的所有数据。
- 全局批处理大小：在每次迭代中从训练数据集中选择的要发送到群集中 GPU 的记录数。这是每次迭代时计算渐变的记录数。如果使用数据并行机制，则等于模型副本总数乘以每副本批处理大小：`global batch size = (the number of model replicas) * (per-replica batch size)`. 全局批次大小的单个批次通常被称为小批在机器学习文献中。
- 每副本批处理大小：使用数据并行机制时，这是发送到每个模型副本的记录数。每个模型副本都使用此批次执行向前和向后传递以计算重量更新。在处理下一组每副本批处理之前，生成的权重更新将在所有副本之间同步（平均值）。
- 微批：小批处理的子集，或者，如果使用混合模型和数据并行机制，则它是每副本大小批次的子集。当您使用 SageMaker 的分布式模型并行度库时，每个微批量都会逐个输入到训练管道中并遵循执行时间表由图书馆的运行时间定义。

训练

- Epoch：对整个数据集进行一个训练周期。每个时代都有多次迭代是常见的。你在训练中使用的时代数量在你的模型和用例中是唯一的。
- 迭代: 使用全局批次大小批次（小批次）的训练数据执行单次向前和向后传递。训练期间执行的迭代次数取决于全局批次大小和用于训练的纪元数。例如，如果数据集包含 5,000 个样本，并且您使用 500 个全局批次大小，则需要 10 次迭代才能完成一个纪元。
- Learning rate (学习速率)：一个变量，它影响根据模型的计算误差而更改权重的量。学习率在模型的收敛能力以及收敛的速度和最佳性方面发挥着重要作用。

实例和 GPU

- 实例：网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon 机器学习计算实例. 这些也称为节点.
- 集群大小：使用 SageMaker 的分布式训练库时，这是每个实例中的实例数乘以 GPU 的数量。例如，如果您在训练作业中使用两个 ml.p3.8xlarge 实例（每个实例有 4 个 GPU），则集群大小为 8。虽然增加群集规模可以缩短训练时间，但实例之间的通信必须得到优化；否则，节点之间的通信可能会增加开销并导致

训练时间缩短。SageMaker 分布式培训库旨在优化之间的沟通Amazon机器学习计算实例，从而提高设备利用率和更快的训练时间。

分布式训练方案

- 数据并行：分布式训练中的一种策略，其中训练数据集被拆分到多个处理节点（例如AmazonML 实例），并且每个处理节点都包含复制品模型。每个节点接收不同批次的训练数据，执行向前和向后传递，并与其他节点共享权重更新以进行同步，然后再转到下一个批次，最终进入另一个时代。
- 模型并行：分布式训练中的一种策略，其中模型在多个处理节点之间进行分区（例如AmazonML 实例)。该模型可能很复杂，并且有大量隐藏的层和权重，因此无法适应单个节点的内存。每个节点都带有模型的一个子集，通过该子集共享和编译数据流和转换。在 GPU 利用率和训练时间方面，模型并行性的效率在很大程度上取决于模型的分区方式以及用于执行前向和向后传递的执行计划。
- 管道执行计划(管道): 管道执行计划决定了模型训练期间进行计算（微批量）和跨设备处理数据的顺序。Pipelining 是一种技术，可通过在不同的数据样本上同时计算 GPU 来实现模型并行性能，并克服由于顺序计算而导致的性能损失。要了解更多信息，请参阅管道执行计划.

示例

以下示例演示了如何使用这些术语来描述使用混合模型和数据并行性的培训作业：

启动了一项双向数据并行性和 4 向模型并行度训练作业，其全球批量为 64 个图像。此培训工作总共需要 8 个 GPU。两个模型副本中的每个模型副本都处理大小为 32 的每个副本批次。在向前和向后传递期间，每个副本批次将进一步划分为微批次。这些微批处理使用流水线的执行计划以交错方式进行处理。

训练数据集包括 640 张图像，因此一个时代需要 10 次迭代。

高级概念

Machine Learning (ML) 从业者在训练模型时通常面临两个扩展挑战：扩展模型大小和扩展训练数据。 虽然模型的大小和复杂性可以提高准确性，但是可以放入单个 CPU 或 GPU 的模型大小有限制。此外，扩展模型大小可能会导致更多的计算和更长的训练时间。

并非所有模型都处理训练数据的扩展能力都一样好，因为它们需要摄取所有训练数据记忆中的对于训练。它们只能垂直扩展，并扩展到更大和更大的实例类型。在大多数情况下，扩展训练数据会导致更长的训练时间。

深度学习 (DL) 是一个特定的机器学习算法系列，由多层人工神经网络组成。最常见的训练方法是使用小批处理随机梯度下降 (SGD)。在小批次 SGD 中，模型的训练方法是在减少误差的方向上对其系数进行小的迭代变化。这些迭代是在训练数据集的同等大小的子样本上进行的，称为小批. 对于每个小批次，模型都在小批次的每个记录中运行，测量的误差和估计误差的梯度。然后，在小批次的所有记录中测量平均梯度，并为每个模型系数提供更新方向。训练数据集的一次完整过程称为时代. 模型培训通常包括几十到数百个时代。Mini Batch SGD 有几个优势：首先，它的迭代设计使训练时间理论上与数据集大小是线性的。其次，在给定的小批次中，每条记录都由模型单独处理，而不需要记录间的通信，而不是最终梯度平均值。因此，小批量的加工特别适合并行化和分销。

通过将小批记录分发到不同的计算设备上并行进行 SGD 训练被称为数据并行分布式培训，是最常用的 DL 分布模式。数据并行训练是一种相关的分发策略，可以扩展小批量规模并更快地处理每个小批量。但是，数据并行训练带来了额外的复杂性，即必须计算小批次梯度平均值，使用来自所有员工的渐变，然后将其传达给所有员工，这是一个叫做的步骤所有减少随着培训群集的扩展，这可能意味着开销越来越大，如果实施不当或实施不当的硬件减少，也可能严重损害训练时间。

数据并行 SGD 仍然要求开发人员至少能够在计算设备（例如单个 CPU 或 GPU）中安装模型和单条记录。在训练非常大的模型时，例如自然语言处理 (NLP) 中的大型变压器，或者在高分辨率图像上进行分割模型时，可能会出现这种做法不可行的情况。分解工作负载的另一种方法是将模型分割在多个计算设备上，这种方法称为模型并行分布式培训.

策略

分布式训练通常分为两种方法：数据并行和模型并行。并行数据是分布式训练最常见的方法：你有很多数据，批量将数据块发送到多个 CPU 或 GPU（节点）以供神经网络或机器学习算法处理，然后合并结果。每个节点上的神经网络都是一样的。一个并行模型方法用于不能放在一个节点内存中的大型模型；它会分解模型并将不同的部分放在不同的节点上。在这种情况下，您需要将批量数据发送到每个节点，以便在模型的所有部分处理数据。

这些术语网络和模型经常可以互换使用：大型模型实际上是一个具有许多层和参数的大型网络。使用大型网络进行训练会生成一个大型模型，然后将模型加载到网络中，其中包含所有预训练的参数及其权重将大型模型加载到内存中。当你拆分模型以跨节点拆分模型时，你也会拆分底层网络。网络由层组成，为了分割网络，您可以在不同的计算设备上放置层。

在设备之间天真地分割图层的一个常见缺陷是 GPU 的严重利用不足。训练本质上无论是向后传递，都是顺序的，在给定时间，只有一个 GPU 可以主动计算，而其他 GPU 则等待激活发送。现代模型并行库通过使用管道执行计划来提高设备利用率来解决这个问题。但是，只有 Amazon SageMaker 的分布式模型并行库包含自动模型拆分。该库的两个核心功能：自动模型拆分和管道执行计划，它通过自动化决策提高设备利用率，简化了实施模型并行的过程。

使用数据并行和模型并行训练

如果您正在使用大型数据集进行训练，请从数据并行方法开始。如果在训练期间内存不足，可能需要切换到模型并行方法，或尝试混合模型和数据并行。您还可以尝试以下方法来通过数据并行提高性能：

- 更改模型的超参数。
- 减小批处理大小。
- 继续缩小批量大小，直到它适合。如果将批量大小减少到 1，但内存仍然不足，那么你应该尝试模型并行训练。

尝试渐变压缩（FP16、INT8）：

- 在配备 NVIDIA TensorCore 的硬件上，使用混合精度训练同时减少了加速和内存消耗。
- SageMaker 的分布式数据并行库支持开箱即用的自动混合精度 (AMP)。除了对训练脚本的框架级修改之外，无需执行额外操作即可启用 AMP。如果渐变位于 FP16 中，则 SageMaker 数据并行度库运行`AllReduce`在 FP16 中操作。有关将 AMP API 实施到训练脚本的更多信息，请参阅以下资源：
  - 框架-PyTorch中的NVIDIA 深度学习性能文档
  - 框架-TensorFlow中的NVIDIA 深度学习性能文档
  - 自动混合精度，用于深度学习中的NVIDIA 开发者文档
  - 引入原生 PyTorch 自动混合精度，以加快 NVIDIA GPU 上的训练速度中的PyTorch 博客
  - TensorFlow 混合精度 API中的TensorFlow 文档

尝试减小输入大小：

- 如果增加序列链接、需要调整批处理大小或向上调整 GPU 以分散批处理，则减少 NLP 序列长度。
- 降低图像分辨率。

检查是否使用批量标准化，因为这可能会影响收敛性。当你使用分布式训练时，你的批处理被拆分到 GPU 之间，而且批量大小更低的影响可能是更高的错误率，从而破坏模型的收敛。例如，如果您在批量大小为 64 的单个 GPU 上对网络进行原型设计，然后扩展到使用四个 p3dn.24xlarge，则现在有 32 个 GPU，每个 GPU 的批量大小从 64 个降至 2。这可能会打破你在单个节点上看到的收敛。

在以下情况下从模型并行训练开始：

- 你的型号不适合在单个设备上。

- 由于模型大小，您在选择较大批量时面临限制，例如模型权重占用了大部分 GPU 内存，而且您不得不选择较小、次优的批处理大小。

要了解有关 SageMaker 分布式库的更多信息，请参阅以下内容：

- SageMaker 的分布式数据并行库 (p. 1971)
- SageMaker的分布式模型并行 (p. 1998)

优化分布式训练

根据您的使用案例和数据自定义超参数，以获得最佳的扩展效率。在下面的讨论中，我们重点介绍了一些最具影响力的训练变量，并提供了最先进的实施方案的参考，以便您可以更多地了解自己的选择 此外，我们建议您参考首选框架的分布式培训文档。

- Apache MXNet 分布式训练
- PyTorch 分布式训练
- TensorFlow 分布式训练

Batch 大小

SageMaker 分发的工具包通常允许您进行更大批量的训练。例如，如果模型适合单个设备，但只能用较小批量进行训练，那么使用模型并行训练或数据并行训练使您能够尝试更大的批量大小。

请注意，批量大小通过控制每次迭代时模型更新中的噪声量直接影响模型准确性。增加批次大小可以减少梯度估计中的噪声量，这在从非常小的批量增加时会有所帮助，但随着批量大小增加到较大值，可能会导致模型准确性降低。

> Tip
>
> 调整超参数以确保模型在增加批次大小时能够达到令人满意的收敛性。

已经开发了许多技术，以便在批量增加时保持良好的模型收敛性。

迷你批量大小

在 SGD 中，迷你批量大小量量化了梯度估计中存在的噪声量。小批次会产生非常嘈杂的迷你批量渐变，这并不代表数据集的真正梯度。大型迷你批量生成的小批量渐变接近数据集的真实渐变，并且可能不够嘈杂，可能会保持锁定在无关紧要的最小值中。

要了解有关这些技术的更多信息，请参阅以下论文：

- 准确、大型迷你匹配 SGDD：在 1 小时内训练 ImageNet，戈雅等。
- PowerAI DDL，Cho 等。
- 向外扩展大型迷你匹配 SGD：在 Imagenet-1k 上进行残余网络训练，提高了准确性并缩短了训练时间，Codreanu 等人。
- 在几分钟内进行 ImageNet 培训，你等。
- 卷积网络的大 Batch 训练，你等。
- 深度学习的大 Batch 优化：76 分钟内训练 BERT，你等。
- 在 54 分钟内加快了 BERT 预训的大 Batch 优化，郑等。
- 深度梯度压缩，林等。

方案

以下各节介绍了您可能希望扩大培训的场景，以及如何使用这样做Amazon资源的费用。

从单个 GPU 扩展到多个 GPU

机器学习中使用的数据量或模型的大小可能会造成训练模型的时间比您愿意等待的时间更长的情况。有时，训练根本不起作用，因为模型或训练数据太大。一种解决方案是增加用于训练的 GPU 数量。 在具有多个 GPU 的实例上，例如`p3.16xlarge`有八个 GPU，数据和处理被拆分到八个 GPU 之间。当您使用分布式训练库时，这可能会导致训练模型所需的时间几乎线性加快。它需要的时间略超过 1/8`p3.2xlarge`只需一个 GPU。

实例类型	GPU
p3.2xlarge	1
p3.8xlarge	4
p3.16xlarge	8
p3dn.24xlarge	8

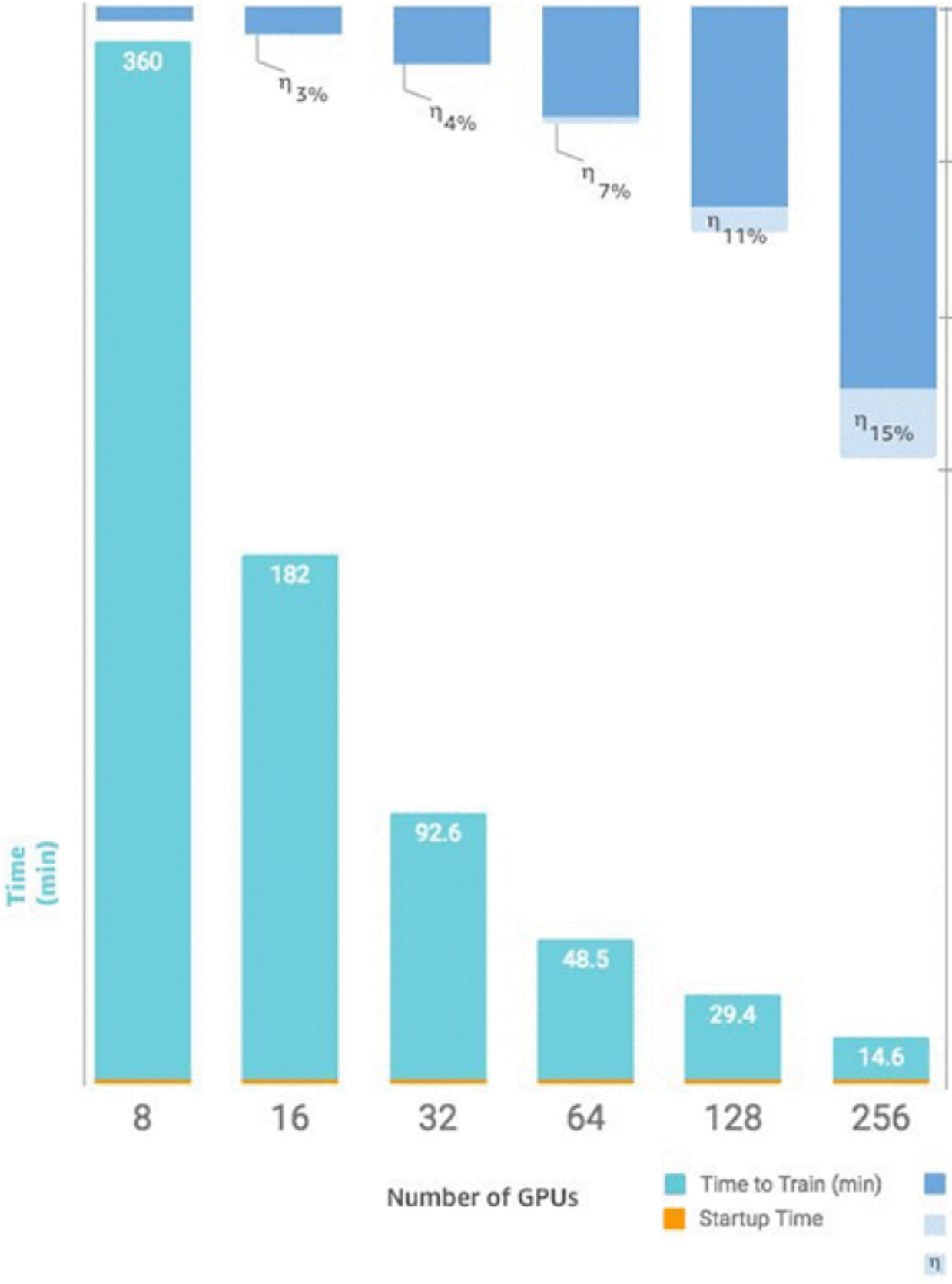
> **Note**
>
> SageMaker 培训使用的 ml 实例类型与相应的 p3 实例类型具有相同数量的 GPU。例如，`ml.p3.8xlarge`具有与相同数量的 GPU`p3.8xlarge`-4.

从单个实例扩展到多个实例

如果你想进一步扩展训练，你可以使用更多的实例。但是，在添加更多实例之前，您应该选择更大的实例类型。查看上表以了解每种 p3 实例类型中有多少 GPU。

如果你已经从一个 GPU 上跳出`p3.2xlarge`在一个上的四个 GPU`p3.8xlarge`，但是决定你需要更多的处理能力，如果选择`p3.16xlarge`在尝试增加实例数之前。根据您使用的库，当您将培训保留在单个实例上时，性能会更好，成本比使用多个实例的场景更低。

当你准备好扩展实例数量时，你可以使用 SageMaker Python SDK 来做到这一点`estimator`通过设置你的功能`instance_count`. 例如，您可以设置`instance_type = p3.16xlarge`和`instance_count = 2`. 而不是单个上的八个 GPU`p3.16xlarge`，你在两个相同的实例中有 16 个 GPU。下图显示了扩展和吞吐量从八个 GPU 开始在单个实例上，并增加到 64 个实例，总共 256 个 GPU。


360
η 3%
η 4%
η 7%
η 11%
η 15%
182
92.6
48.5
29.4
14.6
Time (min)
8
16
32
64
128
256
Number of GPUs
Time to Train (min)
Startup Time
η

可用区和网络背板

对于多个实例，重要的是了解连接实例的网络、实例如何读取训练数据以及它们之间如何共享信息（例如，在执行`AllReduce`操作)。

首先，您的实例必须位于同一区域和同一可用区。例如，us-west-2 中的实例必须全部位于 us-west-2a。当你使用 SageMaker Python SDK 时，这是为你处理的。如果您使用 Amazon EC2 并编排自己的培训集群，则需要注意这一点，否则您的训练速度会受到影响。

您的训练数据也应位于同一可用区。使用 SageMaker 估计器时，您将传入区域和 S3 存储桶，如果数据不在您设置的区域中，则会出现错误。

优化 GPU、网络和存储

这些区域有：`p3dn.24xlarge`实例类型专为快速本地存储和高达 100 GB 的快速网络背板而设计，我们强烈推荐它作为分布式培训的最高性能选项。SageMaker 支持 S3 中的流数据模式，称为`pipe`模式。对于 HPC 负载（如分布式培训），我们建议亚马逊 Fx用于文件存储。

自定义训练脚本

虽然 SageMaker 可以轻松部署和扩展实例和 GPU 的数量，具体取决于您选择的框架，但管理数据和结果可能非常具有挑战性，这就是经常使用外部支持库的原因。 这种最基本的分布式训练形式需要修改训练脚本以管理数据分发。

SageMaker 还支持 Horovod 以及每个主要深度学习框架原生的分布式培训的实施。如果你选择使用这些框架中的示例，你可以关注 SageMaker容器指南适用于 Deep Learning Containers，以及各种示例笔记本展示了实现。

SageMaker 内置分布式培训功能

这些区域有：SageMaker 内置算法库由 18 种流行的机器学习算法组成。其中许多都是从头开始重写的，以便可扩展并开箱即用分发。如果你想使用分布式深度学习培训代码，我们推荐 Amazon SageMaker 的分布式培训库。SageMaker 的分布式培训库使您可以更轻松地编写高度可扩展且经济高效的自定义数据并行并行深度学习培训作业模型。

SageMaker 分布式培训库提供数据并行和模型并行训练策略。它结合了软件和硬件技术，以改善 GPU 间和节点间的通信。它通过内置选项扩展了 SageMaker 的训练功能，这些选项只需对训练脚本进行少量的代码更改。

- SageMaker 的分布式数据并行库 (p. 1971)
- SageMaker的分布式模型并行 (p. 1998)

SageMaker 的分布式数据并行库

这些区域有： SageMaker 分布式数据 parallel 库扩展 SageMaker 在具有近线性扩展效率的深度学习模型上的培训功能，可快速 time-to-train 只需最少的代码更改。

在对大量数据进行模型训练时，机器学习从业人员经常转向分布式培训以缩短培训时间。在某些情况下，在时间至关重要的情况下，业务要求是尽快完成培训，或至少在有限的时间内完成培训。然后，分布式训练将扩展为使用多个节点组成的群集，不仅仅是计算实例中的多个 GPU，而是具有多个 GPU 的多个实例。随着群集规模的增加，性能的显著下降也是如此。性能下降的主要原因是群集中节点之间的通信开销。

为了解决这种开销问题， SageMaker 提供两种分布式培训选项： SageMaker parallel 模型和 SageMaker parallel 数据。本指南重点介绍如何使用 SageMaker 数据 parallel 库。

- 该图书馆优化你的训练工作Amazon网络基础设施和 Amazon EC2 实例拓扑。
- 该库利用渐变更新在具有自定义功能的节点之间进行通信`AllReduce`算法。

要跟踪库的最新更新，请参阅SageMaker 分布式数据并行发行说中的SageMaker Python SDK 文档.

有关使用并行模型策略训练的更多信息，请参阅SageMaker的分布式模型并行 (p. 1998).

主题

- SageMaker 的分布式数据并行库简介 (p. 1972)
- 支持的框架，Amazon Web Services 区域和实例类型 (p. 1974)
- 运行 SageMaker 具有数据并行性的分布式培训 Job (p. 1977)
- SageMaker 分布式数据并行配置提示和陷阱 (p. 1993)
- 亚马逊SageMaker数据并行库常见问题 (p. 1994)
- 数据并行排查 (p. 1996)

SageMaker 的分布式数据并行库简介

为什么使用 SageMaker 分布式数据并行库？

SageMaker 的分布式数据并行库以两种方式解决通信开销问题：

1. 库执行`AllReduce`，这是分布式培训期间的一项关键操作，它承担了很大一部分的通信开销。
2. 该库通过充分利用执行优化的节点到节点通信Amazon的网络基础设施和 Amazon EC2 实例拓扑。

使用此数据并行库可将培训模型（如 BERT）的速度提高 25%。虽然像 Horovod 这样的实现提供了大规模的亚线性性能，但该库提供了接近线性的大规模性能。这意味着您可以获得更快的训练时间和更低的训练模型的成本。

> **Note**
>
> SageMaker 分布式培训库只能通过AmazonSageMaker 培训平台中的 TensorFlow、PyTorch 和 HuggingFace 框架的深度学习容器。要使用这些库，您必须使用 SageMaker Python SDK 或者通过适用 SDK for Python (Boto3) 使用 SageMaker API 或Amazon Command Line Interface. 在整个文档中，说明和示例都重点介绍了如何将分布式培训库与 SageMaker Python SDK 结合使用。

训练基准

PyTorch 与 SageMaker 的数据并行库

Model	Setup	Throughput			Scaling Efficiency		
		PT-DDP	SageMaker	Speed up	PT-DDP	SageMaker	Improvement
BERT Large (seqs/sec)	2 node p3dn.24xl	1752	2479	41%	64%	90%	26%
	4 node p3dn.24xl	3017	4603	52%	55%	84%	29%
	8 node p3dn.24xl	7409	8551	15%	67%	78%	11%
MaskRCNN (samples/sec)	2 node p3dn.24xl	152	158	4%	82%	85%	3%
	4 node p3dn.24xl	258	307	19%	70%	83%	13%
	8 node p3dn.24xl	545	617	13%	74%	84%	10%

使用实例类型`p3dn.24xl`以及在 2、4 和 8 个节点群集上：

- 伯特：与 PyTorch 一起使用时，SageMaker 库比 Pytorch-DDP 快 41%、52% 和 13%。
- maskrCNN：与 PyTorch 一起使用时，SageMaker 库比 Pytorch-DDP 快 4%、19% 和 15%。

这些基准测试是在 PyTorch v1.6 上运行的`ml.p3dn.24xlarge`实例。你可以在SageMaker 示例网站. 示例网站还有这些模型的基准训练代码使用 TensorFlow 2.3.

平衡融合缓冲区实现最佳带宽使

SageMaker 的分布式数据并行库使用类似于参数服务器的通信模式来减少传输的数据量和平均来自多个 GPU 的渐变所涉及的步骤数。它还使用称为平衡融合缓冲区的新技术来最佳利用群集中所有节点上的可用带宽。

传统参数服务器的一个主要缺点是它们对可用网络带宽的利用不够最佳。参数服务器将变量视为原子单位并将每个变量放在一台服务器上。由于渐变在向后传递过程中依次可用，因此在任何给定的时刻，从不同服务器发送和接收的数据量都存在不平衡。有些服务器正在接收和发送更多数据，有些数据更少，有些没有。随着参数服务器数量的增加，这个问题变得更加严重。

图书馆通过介绍来解决这些问题平衡融合缓冲区. 平衡融合缓冲区是 GPU 中的缓冲区，它保存渐变，直到缓冲区的大小超过阈值。在使用 N 个参数服务器的设置中，当缓冲区超过阈值时，平衡融合缓冲区被复制到 CPU 内存中，分片成 N 个部分，第 i 个部分发送到第 i 个参数服务器。每台服务器从平衡融合缓冲区接收的字节数完全相同。第 i 个服务器从所有工作线程处接收平衡融合缓冲区的第 i 个分区，然后将结果汇总起来，然后将结果发送回所有工作线程。由于所有服务器平均参与平均每个平衡融合缓冲区，因此服务器带宽得到了高效利用

高效实现最佳 GPU 使用率`AllReduce`与向后通行证重叠

SageMaker 的分布式数据并行库实现了`AllReduce`通过向后操作，显著提高 GPU 利用率，并通过优化 CPU 和 GPU 之间的任务实现接近线性的扩展效率和更快的训练时间。库执行`AllReduce`同时 GPU 正在计算渐变而不占用额外的 GPU 周期，这使库更快。

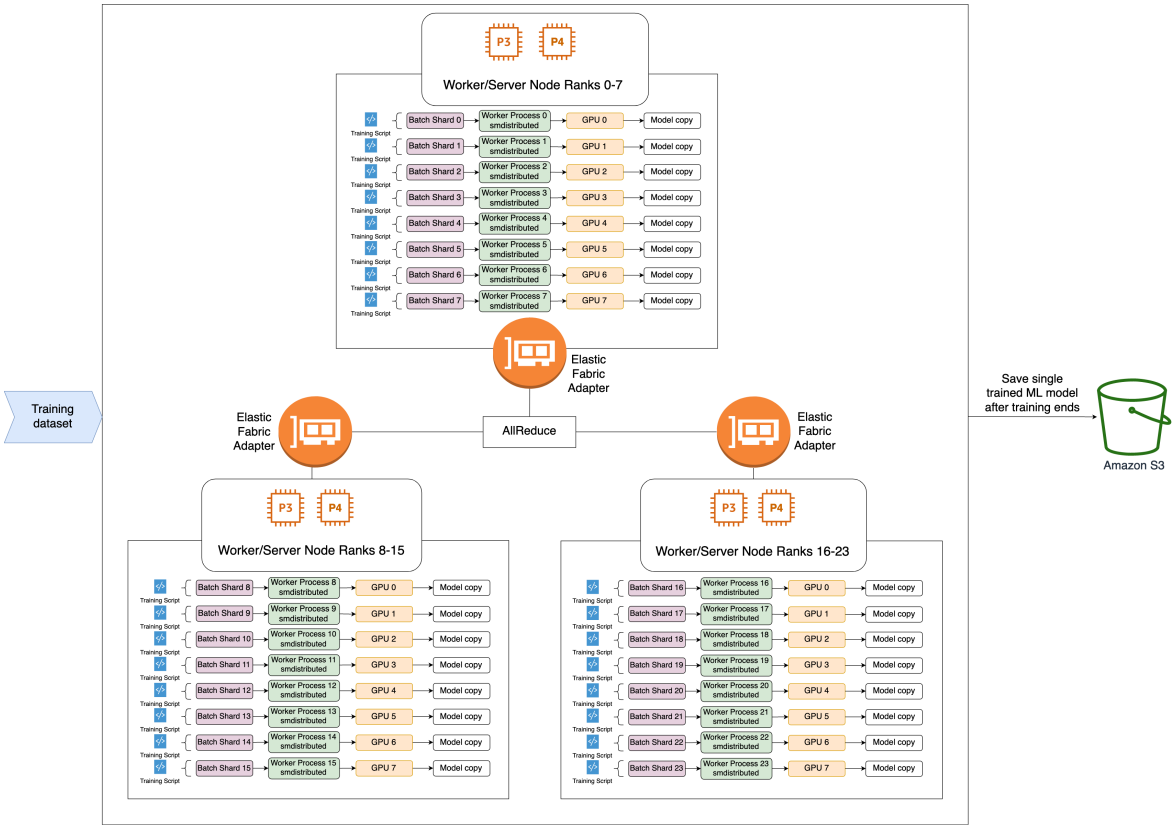
- 利用 CPU：该库使用 CPU 进行 allReduce 渐变，从 GPU 中卸载此任务。
- 改进 GPU 使用率：集群的 GPU 专注于计算梯度，在整个培训过程中提高它们的利用率。

SageMaker 分布式数据并行架构

该库支持大型计算实例，每个节点有 8 个 GPU：`ml.p3.16xlarge`、`ml.p3dn.24xlarge`, 和`ml.p4d.24xlarge`. SageMaker 分布式数据并行库的高级工作流程如下：

1. 图书馆将排名分配给 GPU（工作人员）。
2. 在每次迭代中，库将每个全局批次除以工作人员总数（世界大小），然后将小批量（批处理分片）分配给工作人员。
   - 全局批次的大小是`(number of nodes in a cluster) * (number of GPUs per node) * (per batch shard)`.
   - 批处理分片（小批量）是每次迭代分配给每个 GPU（worker）的数据集的子集。
3. 图书馆为每位工作人员启动训练脚本。
4. 该库在每次迭代结束时管理来自工作人员的模型权重和渐变的副本。
5. 该库同步工作人员之间的模型权重和梯度，以聚合单个训练过的模型。

下面的体系结构图显示了库如何为包含 3 个节点的群集设置数据并行性的示例。



要开始使用 SageMaker 分布式数据并行库，请参阅第 2 步：启动 SageMaker 使用分布式培训 Job SageMaker Python 开发工具包 (p. 1982)通过设置 SageMaker 估算器Amazon SageMaker Python 开发工具包, 和运行 SageMaker 具有数据并行性的分布式培训 Job (p. 1977)使用 SageMaker 分布式数据并行库调整训练脚本。

支持的框架，Amazon Web Services 区域和实例类型

在使用之前 SageMaker 数据 parallel 库，检查哪些支持的 ML 框架和实例类型以及您的中是否有足够的配额 Amazon账户和Amazon Web Services 区域.

支持的框架

这些区域有： SageMaker data parallel 库支持以下深度学习框架，可在AmazonDeep Learning Containers (DLC) 或可作为二进制文件下载。

PyTorch 版本支持 SageMaker 和 SageMaker 分布式数据 parallel 库

PyTorch 版本	SageMaker 分布式数据 parallel 库版本	**`smdistributed-dataparallel`**集成映像 URI	二进制文件的 URL **
v1.11.0	`smdistributed-dataparallel==v1.4.1`	763104351884.dkr.ecr.<region>.amazonaws.com/Pytorch-训练:1.11.0-gpu-py38-cu113-ubuntu20.04	https://smdataparallel.s3.amazonaws.com/binary/pytorch/1.11.0/cu113/2022-04-14/smdistributed_dataparallel-1.4.1-cp38-cp38-linux_x86_64.whl

PyTorch 版本	SageMaker 分布式数据 parallel 库版本	**smdistributed-dataparallel**集成映像 URI	二进制文件的 URL **
v1.10.2	`smdistributed-dataparallel==v1.4.0`	763104351884.dkr.ecr.<region>.amazonaws.com/pytorch 训练：1.10.2-gpu-py38-cu113-ubuntu20.04-sagemaker	https://smdataparallel.s3.amazonaws.com/binary/pytorch/1.10.2/cu113/2022-02-18/smdistributed_dataparallel-1.4.0-cp38-cp38-linux_x86_64.whl
v1.9.1	`smdistributed-dataparallel==v1.2.0`	763104351884.dkr.ecr.<region>.amazonaws.com/py38-cu1-ubuntu20.04	https://smdataparallel.s3.amazonaws.com/binary/pytorch/1.9.0/cu111/2021-08-13/smdistributed_dataparallel-1.2.0-cp38-cp38-linux_x86_64.whl
v1.8.1	`smdistributed-dataparallel==v1.2.3`	763104351884.dkr.ecr.<region>.amazonaws.com/py36-cu1-ubuntu18.04	https://smdataparallel.s3.amazonaws.com/binary/pytorch/1.8.1/cu111/2021-12-13/smdistributed_dataparallel-1.2.3-cp36-cp36m-linux_x86_64.whl
v1.7.1	`smdistributed-dataparallel==v1.0.0`	763104351884.dkr.ecr.<region>.amazonaws.com/py36-cu110-ubuntu18.04	https://smdataparallel.s3.amazonaws.com/binary/pytorch/1.7.1/cu110/2021-01-26/smdistributed_dataparallel-1.0.0-cp36-cp36m-linux_x86_64.whl
v1.6.0	`smdistributed-dataparallel==v1.0.0`	763104351884.dkr.ecr.<region>.amazonaws.com/Pytorch-训练:1.6.0-gpu-py36-cu110-ubuntu18.04	https://smdataparallel.s3.amazonaws.com/binary/pytorch/1.6.0/cu110/2021-01-14/smdistributed_dataparallel-1.0.0-cp36-cp36m-linux_x86_64.whl

* 该 SageMaker 分布式数据并行度库 v1.4.0 及更高版本可作为 PyTorch 分布式。根据变更，以下sm分布的API(对于) PyTorch 已弃用分布式软件包。

- `smdistributed.dataparallel.torch.distributed`已弃用。使用torch.分布式/分布而不是打包。
- `smdistributed.dataparallel.torch.parallel.DistributedDataParallel`已弃用。使用Torch.nn.Parallel.Pallel.Pallel.Pallell而不是 API。

  如果需要使用库的早期版本（v1.3.0 或早期版本），请参阅archived SageMaker 分布式数据 parallel 库文档中的SageMaker Python SDK 文档.

** 二进制文件的 URL 用于安装 SageMaker 自定义容器中的分布式数据并行库。有关更多信息，请参阅 使用创建自己的 Docker 容器 SageMaker 分布式数据并行库 (p. 1985)。

支持的 TensorFlow 版本 SageMaker 和 SageMaker 分布式数据 parallel 库

TensorFlow 版本	SageMaker 分布式数据 parallel 库版本	**smdistributed-dataparallel**集成映像 URI
2.8.0	smdistributed-dataparallel==v1.3.0	763104351884.dkr.ecr.*<region>*.amazonaws.c tensorflow: 2.8.0-gpu-py39-cu2-ubuntu20.04
2.7.1	smdistributed-dataparallel==v1.3.0	763104351884.dkr.ecr.*<region>*.amazonaws.c 张量流训练：2.7.1-gpu-py38-cu112-ubuntu20.04-sagemaker
2.6.2	smdistributed-dataparallel==v1.2.1	763104351884.dkr.ecr.*<region>*.amazonaws.c 张量流训练：2.6.2-gpu-py38-cu112-ubuntu20.04
2.5.1	smdistributed-dataparallel==v1.2.1	763104351884.dkr.ecr.*<region>*.amazonaws.c tensorflow 推理:2.5.1-gpu-py37-cu2-ubuntu18.04
2.4.1	smdistributed-dataparallel==v1.2.0	763104351884.dkr.ecr.*<region>*.amazonaws.c tensorflow: 2.4.1-gpu-py37-cu110-ubuntu18.04
2.3.2	smdistributed-dataparallel==v1.0.0	763104351884.dkr.ecr.*<region>*.amazonaws.c tensorflow: 2.3.2-gpu-py37-cu110-ubuntu18.04

拥抱 Face 变形金刚支持的版本 SageMaker 和 SageMaker 分布式数据 parallel 库

这些区域有：Amazon用于拥抱脸部的 Deep Learning Containers 使用 SageMaker 用于的训练容器 PyTorch 和 TensorFlow 作为他们的基本图片。查找拥抱 Face 变形金刚库版本并配对 PyTorch 和 TensorFlow 版本，请参阅最新拥抱脸容器和之前的拥抱 Face 容器版本.

Amazon Web Services 区域

这些区域有： SageMaker 数据 parallel 库在所有Amazon Web Services 区域其中AmazonSageMaker 的 Deep Learning Containers在服务中。有关更多信息，请参阅 。可用的 Deep Learning Containers 映像.

支持的 实例类型

这些区域有： SageMaker 数据 parallel 库需要以下机器学习实例类型之一。

实例类型
ml.p3.16xlarge
ml.p3dn.24xlarge
ml.p4d.24xlarge

有关实例类型的规格，请参阅加速计算部分中的Amazon EC2 实例类型. 有关实例定价的信息，请参阅亚马逊 SageMaker 定价.

如果您遇到类似于以下内容的错误消息，请按照请求提升服务配额 SageMaker 资源.

```
ResourceLimitExceeded: An error occurred (ResourceLimitExceeded) when calling
the CreateTrainingJob operation: The account-level service limit 'ml.p3dn.24xlarge
for training job usage' is 0 Instances, with current utilization of 0 Instances
and a request delta of 1 Instances.
Please contact AWS support to request an increase for this limit.
```

运行 SageMaker 具有数据并行性的分布式培训 Job

SageMaker 的分布式数据 parallel 库 API 旨在便于使用，并提供与现有分布式培训工具包的无缝集成。

- 使用库 API 的 SageMaker Python 开发工具包— 在大多数情况下，训练脚本中只需更改数据 parallel 库导入语句。将这些交换成 SageMaker 数据 parallel 库等价物。
- 无需基础架构管理，专注于模型培训— 在 SageMaker 上使用库训练深度学习模型时，您可以专注于编写训练脚本和模型训练。你可以使用由 SageMaker Python 开发工具包。估计器类有助于准备 ML 实例、从指定的数据资源加载数据集、使用训练脚本提交训练作业，以及在训练作业完成后关闭实例。

要开始操作，您需要适应 TensorFlow 要么 PyTorch 训练脚本以使用库。以下主题提供了有关如何修改训练脚本的说明。

主题

- 第 1 步：修改您自己的训练脚本 (p. 1977)
- 第 2 步：启动 SageMaker 使用分布式培训 Job SageMaker Python 开发工具包 (p. 1982)

第 1 步：修改您自己的训练脚本

使用本节可以了解如何自定义训练脚本以使用亚马逊的核心功能 SageMaker 分布式数据 parallel 库。要使用库特定的 API 函数和参数，我们建议您将本文档与SageMaker 数据 parallel 库 API中的SageMaker Python 开发工具包文.

这些部分中提供的训练脚本示例经过简化，旨在突出显示使用库所需的更改。有关演示如何使用的端到端、可运行的笔记本示例 TensorFlow 要么 PyTorch 使用 SageMaker 分布式数据 parallel 库，请参阅亚马逊 SageMaker 分布式训练记本示例 (p. 2043).

主题

- 修改 TensorFlow 训练脚本 (p. 1977)
- 修改 PyTorch 训练脚本 (p. 1979)

修改 TensorFlow 训练脚本

以下步骤向您显示如何修改 TensorFlow 用于利用 SageMaker 的分布式数据 parallel 库的训练脚本。

库 API 的设计与霍罗沃德 API 类似。有关库为 TensorFlow 提供的每个 API 的其他详细信息，请参阅SageMaker parallel 分布式数据 TensorFlow API 文档.

> **Note**
>
> SageMaker 分布式数据 parallel 可适应 TensorFlow 训练脚本由以下内容`tf`核心模块除外`tf.keras`模块。 SageMaker 不支持分布式数据 parallel TensorFlow 与 Keras 实施结合使用。

> **Note**
>
> SageMaker 的分布式数据并行库支持开箱即用的自动混合精度 (AMP)。除了对训练脚本的框架级修改之外，无需执行额外操作即可启用 AMP。如果渐变位于 FP16 中，则 SageMaker 数据并行度库运行它`AllReduce`在 FP16 中操作。有关在训练脚本中实施 AMP API 的更多信息，请参阅以下资源：

- TensorFlow中的NVIDIA 深度学习性能文档
- 深度学习自动混合精度中的NVIDIA 开发人员文档
- TensorFlow 混合精度 API中的TensorFlow 文档

1. 导入图书馆的 TensorFlow 客户端然后初始化它。

```
import smdistributed.dataparallel.tensorflow as sdp
sdp.init()
```

2. 将每个 GPU 固定到单个 GPUsmdistributed.dataparallel使用处理local_rank— 这是指给定节点内进程的相对排名。这些区域有：sdp.tensorflow.local_rank()API 为您提供设备的本地排名。领导节点为等级 0，工作人员节点为等级 1、2、3 等。这在以下代码块中被调用为sdp.local_rank().set_memory_growth与之无直接关系 SageMaker 分布式，但必须设置为使用 TensorFlow 进行分布式训练。

```
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
if gpus:
    tf.config.experimental.set_visible_devices(gpus[sdp.local_rank()], 'GPU')
```

3. 按工作人员数量扩展学习率。这些区域有：sdp.tensorflow.size()API 为您提供集群中的工作人员数量。这在以下代码块中被调用为sdp.size().

```
learning_rate = learning_rate * sdp.size()
```

4. 使用图书馆的DistributedGradientTape优化AllReduce训练期间的操作。这个包装tf.GradientTape.

```
with tf.GradientTape() as tape:
      output = model(input)
      loss_value = loss(label, output)

# SageMaker data parallel: Wrap tf.GradientTape with the library's
 DistributedGradientTape
tape = sdp.DistributedGradientTape(tape)
```

5. 将领导节点（等级 0）的初始模型变量广播到所有工作节点（等级 1 到 n）。这对于确保所有员工队伍的一致初始化是必要的。使用sdp.tensorflow.broadcast_variables初始化模型和优化程序变量之后的 API。 这在以下代码块中被调用为sdp.broadcast_variables().

```
sdp.broadcast_variables(model.variables, root_rank=0)
sdp.broadcast_variables(opt.variables(), root_rank=0)
```

6. 最后，修改脚本以仅在领导节点上保存检查点。领导节点具有同步模型。这还可以避免工作人员节点覆盖检查点并可能破坏检查点。

```
if sdp.rank() == 0:
    checkpoint.save(checkpoint_dir)
```

以下是示例： TensorFlow 使用图书馆进行分布式训练的训练脚本。

```
import tensorflow as tf

# SageMaker data parallel: Import the library TF API
import smdistributed.dataparallel.tensorflow as sdp
```

```
# SageMaker data parallel: Initialize the library
sdp.init()

gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
if gpus:
    # SageMaker data parallel: Pin GPUs to a single library process
    tf.config.experimental.set_visible_devices(gpus[sdp.local_rank()], 'GPU')

# Prepare Dataset
dataset = tf.data.Dataset.from_tensor_slices(...)

# Define Model
mnist_model = tf.keras.Sequential(...)
loss = tf.losses.SparseCategoricalCrossentropy()

# SageMaker data parallel: Scale Learning Rate
# LR for 8 node run : 0.000125
# LR for single node run : 0.001
opt = tf.optimizers.Adam(0.000125 * sdp.size())

@tf.function
def training_step(images, labels, first_batch):
    with tf.GradientTape() as tape:
        probs = mnist_model(images, training=True)
        loss_value = loss(labels, probs)

    # SageMaker data parallel: Wrap tf.GradientTape with the library's
 DistributedGradientTape
    tape = sdp.DistributedGradientTape(tape)

    grads = tape.gradient(loss_value, mnist_model.trainable_variables)
    opt.apply_gradients(zip(grads, mnist_model.trainable_variables))

    if first_batch:
       # SageMaker data parallel: Broadcast model and optimizer variables
       sdp.broadcast_variables(mnist_model.variables, root_rank=0)
       sdp.broadcast_variables(opt.variables(), root_rank=0)

    return loss_value

...

# SageMaker data parallel: Save checkpoints only from master node.
if sdp.rank() == 0:
    checkpoint.save(checkpoint_dir)
```

完成训练脚本的调整后，继续前往第 2 步：启动 SageMaker 使用分布式培训 Job SageMaker Python 开发工具包 (p. 1982).

修改 PyTorch 训练脚本

在 SageMaker 数据 parallel 库 v1.4.0 及更高版本，该库可作为后端选项PyTorch 分布式程序. 只需在训练脚本顶部导入库一次，然后将其设置为 PyTorch 初始化期间分布式后端。使用单行后端规范，您可以保留 PyTorch 训练脚本不变并直接使用 PyTorch 分布式模块。要查找该库的最新 API 文档，请参阅SageMaker 为 PyTorch 分布式数据 parallel API中的SageMaker Python 开发工具包文. 了解有关的更多信息 PyTorch 分布式包和后端选项，请参阅分布式通信包-torch.分布式/分布.

> **Important**
>
> 由于 SageMaker 分布式数据并行度库 v1.4.0 及更高版本可作为 PyTorch 已分发，以下sm分布式 API(对于) PyTorch 已弃用分布式软件包。

- `smdistributed.dataparallel.torch.distributed`已弃用。使用torch.分布式/分布而不是打包。
- `smdistributed.dataparallel.torch.parallel.DistributedDataParallel`已弃用。使用torch.nn.Pallel.Pallel.Pallel.Pallel.Pallel.B而不是 API。

如果需要使用库的早期版本（v1.3.0 或更早版本），请参阅archived SageMaker 分布式数据 parallel 库文档中的SageMaker Python 开发工具包文.

使用 SageMaker 作为后端的分布式数据并行库`torch.distributed`

使用 SageMaker 分布式数据 parallel 库，您唯一需要做的是导入 SageMaker 分布式数据 parallel 库 PyTorch 客户端 (`smdistributed.dataparallel.torch.torch_smddp`)。客户注册`smddp`作为 PyTorch 的后端。当你初始化 PyTorch 分布式进程组使用`torch.distributed.init_process_group`API，请务必指定`'smddp'`到`backend`参数。

```
import smdistributed.dataparallel.torch.torch_smddp
import torch.distributed as dist

dist.init_process_group(backend='smddp')
```

Note

这些区域有：`smddp`后端目前不支持使用`torch.distributed.new_group()`API。您无法使用`smddp`后端与其他进程组后端（例如 NCCL 和 Gloo）同时进行。

如果您已有工作方式 PyTorch 脚本，只需添加后端规范，就可以继续使用 SageMaker 框架估算器对于 PyTorch TensorFlow (p. 1983)中的第 2 步：启动 SageMaker 使用分布式培训 Job SageMaker Python 开发工具包 (p. 1982)主题。

如果您仍需修改训练脚本才能正确使用 PyTorch 分布式软件包，请按照本页面上的其余步骤进行操作。

准备 PyTorch 分布式训练的训练脚本

以下步骤提供了有关如何准备训练脚本以使用 PyTorch 成功运行分布式训练作业的其他提示。

Note

在 v1.4.0 中， SageMaker 分布式数据 parallel 库支持以下集体原始数据类型torch.分布式/分布接口：`all_reduce`、`broadcast`、`reduce`、`all_gather`, 和`barrier`.

1. 导入 PyTorch 分布式模块。

   ```
   import torch
   import torch.distributed as dist
   from torch.nn.parallel import DistributedDataParallel as DDP
   ```

2. 在解析参数并定义批处理大小参数之后（例如，`batch_size=args.batch_size`) 中，添加两行代码以调整每个工作人员的批量大小 (GPU)。PyTorch 的 DataLoader 操作不会自动处理分布式训练的批量调整大小。

   ```
   batch_size //= dist.get_world_size()
   batch_size = max(batch_size, 1)
   ```

3. 将每个 GPU 固定到单个 GPU SageMaker 数据 parallel 库进程`local_rank`— 这是指给定节点内进程的相对排名。

   您可以从`LOCAL_RANK`环境变量。

   ```
   import os
   local_rank = os.environ["LOCAL_RANK"]
   ```

```
torch.cuda.set_device(local_rank)
```

4. 定义模型后，用 PyTorch 包装模型`DistributedDataParallel`API。

```
model = ...

# Wrap the model with the PyTorch DistributedDataParallel API
model = DDP(model)
```

5. 当你打电话给`torch.utils.data.distributed.DistributedSampler`API 中，指定在集群中所有节点参与训练的进程 (GPU) 总数。这叫`world_size`，你可以从`torch.distributed.get_world_size()`API。此外，还可以使用`torch.distributed.get_rank()`API。

```
from torch.utils.data.distributed import DistributedSampler

train_sampler = DistributedSampler(
  train_dataset,
  num_replicas = dist.get_world_size(),
  rank = dist.get_rank()
)
```

6. 修改脚本，以便仅在领导进程中保存检查点（等级 0）。领导者进程具有同步的模型。这也避免了其他过程覆盖检查点，并可能会损坏检查点。

```
if dist.get_rank() == 0:
  torch.save(...)
```

以下示例代码显示了 PyTorch 训练脚本`smddp`作为后端。

```
import os
import torch

# SageMaker data parallel: Import the library PyTorch API
import smdistributed.dataparallel.torch.torch_smddp

# SageMaker data parallel: Import PyTorch's distributed API
import torch.distributed as dist
from torch.nn.parallel import DistributedDataParallel as DDP

# SageMaker data parallel: Initialize the process group
dist.init_process_group(backend='smddp')

class Net(nn.Module):
    ...
    # Define model

def train(...):
    ...
    # Model training

def test(...):
    ...
    # Model evaluation

def main():

    # SageMaker data parallel: Scale batch size by world size
    batch_size //= dist.get_world_size()
    batch_size = max(batch_size, 1)
```

```
    # Prepare dataset
    train_dataset = torchvision.datasets.MNIST(...)

    # SageMaker data parallel: Set num_replicas and rank in DistributedSampler
    train_sampler = torch.utils.data.distributed.DistributedSampler(
            train_dataset,
            num_replicas=dist.get_world_size(),
            rank=dist.get_rank())

    train_loader = torch.utils.data.DataLoader(..)

    # SageMaker data parallel: Wrap the PyTorch model with the library's DDP
    model = DDP(Net().to(device))

    # SageMaker data parallel: Pin each GPU to a single library process.
    local_rank = os.environ["LOCAL_RANK"]
    torch.cuda.set_device(local_rank)
    model.cuda(local_rank)

    # Train
    optimizer = optim.Adadelta(...)
    scheduler = StepLR(...)
    for epoch in range(1, args.epochs + 1):
        train(...)
        if rank == 0:
            test(...)
        scheduler.step()

    # SageMaker data parallel: Save model on master node.
    if dist.get_rank() == 0:
        torch.save(...)

if __name__ == '__main__':
    main()
```

完成训练脚本调整后，请继续第 2 步：启动 SageMaker 使用分布式培训 Job SageMaker Python 开发工具包 (p. 1982).

第 2 步：启动 SageMaker 使用分布式培训 Job SageMaker Python 开发工具包

在中运行你的改编脚本第 1 步：修改您自己的训练脚本 (p. 1977)，首先使用准备好的训练脚本和分布式训练配置参数创建 SageMaker 框架或通用估计器对象。您可以在任何类型的中使用库 SageMaker 环境和 Web IDE，例如SageMaker 笔记本实例和SageMaker Studio.

使用高级别SageMaker Python 开发工具要执行以下操作之一：

- 如果您想在 SageMaker 中快速采用分布式培训作业，请配置 SageMaker PyTorch要么TensorFlow框架估算器类。框架估算器会拿起你的训练脚本并自动匹配预构建 PyTorch 要么 TensorFlow Deep Learning Containers (DLC)，给定指定的值`framework_version`参数。
- 如果要扩展其中一个预构建的容器或构建自定义容器以使用 SageMaker 创建自己的机器学习环境，请使用 SageMaker 通用的`Estimator`类，并指定 Amazon Elastic Container Registry (Amazon ECR) 中托管的自定义 Docker 容器的映像 URI。

您的训练数据集应存储在 Amazon S3 中或Amazon FSx for Lustre中的Amazon Web Services 区域你正在开始训练工作。如果你使用 Jupyter 笔记本电脑，你应该有 SageMaker 笔记本实例或 SageMaker Studio 应用程序在同一个中运行Amazon Web Services 区域. 有关存储训练数据的更多信息，请参阅SageMaker Python SDK 数据输入文档中）。

> **Tip**
>
> 我们强烈建议您使用适用 Amazon FSx for Lustre，而不是 Amazon S3，以提高训练绩效。与 Amazon S3 相比，Amazon FSx 的吞吐量更高，延迟更低。

选择以下主题之一，了解有关如何运行 TensorFlow 或 PyTorch 训练脚本。启动培训作业后，您可以使用以下方法监视系统利用率和建模性能。亚马逊SageMakerDebugger (调试程序) (p. 1783)或 Amazon CloudWatch。

当您按照以下主题中的说明了解有关技术细节的更多信息时，我们还建议您尝试亚马逊 SageMaker 分布式训练记本示例 (p. 2043)开始使用。

主题

- 使用 SageMaker 框架估算器对于 PyTorch TensorFlow (p. 1983)
- 使用 SageMaker 用于扩展预构建容器的通用估计器 (p. 1984)
- 使用创建自己的 Docker 容器 SageMaker 分布式数据并行库 (p. 1985)

使用 SageMaker 框架估算器对于 PyTorch TensorFlow

您可以激活 SageMaker 分布式数据 parallel 库通过将其指定为`distribution`中的策略SageMaker 框架估算器类.

Using the SageMaker PyTorch estimator

```
from sagemaker.pytorch import PyTorch

pt_estimator = PyTorch(
    base_job_name="training_job_name_prefix",
    entry_point="adapted-training-script.py",
    role="SageMakerRole",
    py_version="py38",
    framework_version="1.10.2",

    # For running a multi-node distributed training job, specify a value greater than 1
    # Example: 2,3,4,..8
    instance_count=2,

    # Instance types supported by the SageMaker data parallel library:
    # ml.p4d.24xlarge, ml.p3dn.24xlarge, and ml.p3.16xlarge
    instance_type="ml.p3.16xlarge",

    # Training using the SageMaker data parallel distributed training strategy
    distribution={ "smdistributed": { "dataparallel": { "enabled": True } } }
)

pt_estimator.fit("s3://bucket/path/to/training/data")
```

Using the SageMaker TensorFlow estimator

```
from sagemaker.tensorflow import TensorFlow

tf_estimator = TensorFlow(
    base_job_name = "training_job_name_prefix",
    entry_point="adapted-training-script.py",
    role="SageMakerRole",
    framework_version="2.6.3",
    py_version="py38",

    # For running a multi-node distributed training job, specify a value greater than 1
    # Example: 2,3,4,..8
    instance_count=2,

    # Instance types supported by the SageMaker data parallel library:
    # ml.p4d.24xlarge, ml.p3dn.24xlarge, and ml.p3.16xlarge
    instance_type="ml.p3.16xlarge",
```

```
    # Training using the SageMaker data parallel distributed training strategy
    distribution={ "smdistributed": { "dataparallel": { "enabled": True } } }
)

tf_estimator.fit("s3://bucket/path/to/training/data")
```

的以下两个参数 SageMaker 需要框架估算器才能激活 SageMaker 数据并行。

`distribution`(dict)：包含有关如何运行分布式训练的信息的字典（默认值：`None`)。

- 使用`smdistributed.dataparallel`作为分发策略，请按以下代码所示配置字典：

```
distribution = { "smdistributed": { "dataparallel": { "enabled": True } } }
```

- **`custom_mpi_options`**(str)（可选）：自定义 MPI 选项。下面是说明定义时如何使用此参数的示例。`distribution`. 要了解更多信息，请参阅 自定义 MPI 选项 (p. 1993)。

```
distribution = {
    "smdistributed": {
        "dataparallel": {
            "enabled": True,
            "custom_mpi_options": "-verbose -x NCCL_DEBUG=VERSION"
        }
    }
}
```

`instance_type`(str): 要使用的 Amazon EC2 实例的类型。

- 如果使用`smdistributed`和`dataparallel`分配策略，您必须使用以下实例类型之一：`ml.p4d.24xlarge`、`ml.p3dn.24xlarge`，以及`ml.p3.16xlarge`. 为获得最佳性能，我们建议您使用启用 EFA 的实例，`ml.p3dn.24xlarge`和`ml.p4d.24xlarge`.

使用 SageMaker 用于扩展预构建容器的通用估计器

您可以自定义 SageMaker 预构建的容器或扩展，以满足预构建的算法或模型的任何其他功能要求 SageMaker Docker 镜像不支持。有关如何扩展预构建的容器的示例，请参阅扩展预构建的容器.

要扩展预构建的容器或调整自己的容器以使用库，必须使用中列出的其中一个映像支持的框架 (p. 1974).

> **Important**
>
> 从 TensorFlow 2.4.1 和 PyTorch 1.8.1，框架 DLC 支持启用 EFA 的实例类型（`ml.p3dn.24xlarge`、`ml.p4d.24xlarge`)。我们建议您使用包含 DLC 映像 TensorFlow 2.4.1 或更高版本 PyTorch 1.8.1 或更高版本。

例如，如果你使用 PyTorch，你的 Dockerfile 应该包含`FROM`类似于以下内容：

```
# SageMaker PyTorch image
FROM 763104351884.dkr.ecr.<aws-region>.amazonaws.com/pytorch-training:<image-tag>

ENV PATH="/opt/ml/code:${PATH}"

# this environment variable is used by the SageMaker PyTorch container to determine our
 user code directory.
ENV SAGEMAKER_SUBMIT_DIRECTORY /opt/ml/code
```

```
# /opt/ml and all subdirectories are utilized by SageMaker, use the /code subdirectory to
 store your user code.
COPY cifar10.py /opt/ml/code/cifar10.py

# Defines cifar10.py as script entrypoint
ENV SAGEMAKER_PROGRAM cifar10.py
```

您可以进一步自定义自己的 Docker 容器以使用 SageMaker 使用SageMaker 培训工具包和的二进制文件 SageMaker 分布式数据 parallel 库。要了解更多信息，请参阅以下部分中的说明。

使用创建自己的 Docker 容器 SageMaker 分布式数据并行库

要构建自己的 Docker 容器进行训练并使用 SageMaker data parallel 库，必须包含正确的依赖关系和的二进制文件 SageMaker Dockerfile 中的分布式 parallel 库。本节提供了有关如何使用数据 parallel 库在 SageMaker 中使用分布式训练创建一个完整的 Dockerfile 的说明，其中包含最少的依赖关系集。

> **Note**
>
> 这个自定义 Docker 选项带 SageMaker 数据 parallel 库作为二进制文件仅适用于 PyTorch。

使用 SageMaker 培训工具包和数据 parallel 库

1. 从中的 Docker 镜像开始NVIDIA CUDA. 使用包含 CUDA 运行时和开发工具（标头和库）的 CUDNN 开发人员版本从PyTorch 源代码.

   ```
   FROM nvidia/cuda:11.3.1-cudnn8-devel-ubuntu20.04
   ```

   > **Tip**
   >
   > 官方Amazon深度学习容器 (DLC) 映像是从NVIDIA CUDA 基本映像. 如果要在遵循其余说明的同时使用预构建的 DLC 映像作为参考，请参阅Amazon用于的 Deep Learning Containers PyTorch Dockerfile.

2. 添加以下参数以指定的版本 PyTorch 和其他软件包。另外，Amazon S3 明指向 SageMaker 数据 parallel 库和其他要使用的软件Amazon资源，例如 Amazon S3 插件。

   要使用以下代码示例中提供的版本以外的第三方库版本，我们建议您查看的官方 DockerfileAmazonPyTorch 的深度学习容器查找经过测试、兼容且适合您的应用程序的版本。

   查找适用于`SMDATAPARALLEL_BINARY`参数，请参见查找表支持的框架 (p. 1974).

   ```
   ARG PYTORCH_VERSION=1.10.2
   ARG PYTHON_SHORT_VERSION=3.8
   ARG EFA_VERSION=1.14.1
   ARG SMDATAPARALLEL_BINARY=https://smdataparallel.s3.amazonaws.com/binary/pytorch/
   ${PYTORCH_VERSION}/cu113/2022-02-18/smdistributed_dataparallel-1.4.0-cp38-cp38-
   linux_x86_64.whl
   ARG PT_S3_WHL_GPU=https://aws-s3-plugin.s3.us-west-2.amazonaws.com/
   binaries/0.0.1/1c3e69e/awsio-0.0.1-cp38-cp38-manylinux1_x86_64.whl
   ARG CONDA_PREFIX="/opt/conda"
   ARG BRANCH_OFI=1.1.3-aws
   ```

3. 设置以下环境变量以正确构建 SageMaker 训练组件并运行数据 parallel 库。在后续步骤中，您可以将这些变量用于组件。

   ```
   # Set ENV variables required to build PyTorch
   ENV TORCH_CUDA_ARCH_LIST="7.0+PTX 8.0"
   ENV TORCH_NVCC_FLAGS="-Xfatbin -compress-all"
   ENV NCCL_VERSION=2.10.3

   # Add OpenMPI to the path.
   ```

```
ENV PATH /opt/amazon/openmpi/bin:$PATH

# Add Conda to path
ENV PATH $CONDA_PREFIX/bin:$PATH

# Set this enviroment variable for SageMaker to launch SMDDP correctly.
ENV SAGEMAKER_TRAINING_MODULE=sagemaker_pytorch_container.training:main

# Add enviroment variable for processes to be able to call fork()
ENV RDMAV_FORK_SAFE=1

# Indicate the container type
ENV DLC_CONTAINER_TYPE=training

# Add EFA and SMDDP to LD library path
ENV LD_LIBRARY_PATH="/opt/conda/lib/python${PYTHON_SHORT_VERSION}/site-packages/
smdistributed/dataparallel/lib:$LD_LIBRARY_PATH"
ENV LD_LIBRARY_PATH=/opt/amazon/efa/lib/:$LD_LIBRARY_PATH
```

4. 安装或更新curl、wget, 和git在后续步骤中下载和构建软件包。

```
RUN --mount=type=cache,id=apt-final,target=/var/cache/apt \
    apt-get update && apt-get install -y  --no-install-recommends \
        curl \
        wget \
        git \
    && rm -rf /var/lib/apt/lists/*
```

5. 安装EFA Elastic Fabric Adapter (EFA)适用于 Amazon EC2 网络通信的软件。

```
RUN DEBIAN_FRONTEND=noninteractive apt-get update
RUN mkdir /tmp/efa \
    && cd /tmp/efa \
    && curl --silent -O https://efa-installer.amazonaws.com/aws-efa-installer-
${EFA_VERSION}.tar.gz \
    && tar -xf aws-efa-installer-${EFA_VERSION}.tar.gz \
    && cd aws-efa-installer \
    && ./efa_installer.sh -y --skip-kmod -g \
    && rm -rf /tmp/efa
```

6. 安装Conda来处理软件包管理。

```
RUN curl -fsSL -v -o ~/miniconda.sh -O  https://repo.anaconda.com/miniconda/Miniconda3-
latest-Linux-x86_64.sh  && \
    chmod +x ~/miniconda.sh && \
    ~/miniconda.sh -b -p $CONDA_PREFIX && \
    rm ~/miniconda.sh && \
    $CONDA_PREFIX/bin/conda install -y python=${PYTHON_SHORT_VERSION} conda-build pyyaml
 numpy ipython && \
    $CONDA_PREFIX/bin/conda clean -ya
```

7. 获取、构建和安装 PyTorch 以及它的依赖关系。我们建源代码中的 PyTorch因为我们需要控制 NCCL 版本才能保证与AmazonOFI NCCL 插件.

   a. 按照中的步骤操作PyTorch 官方 Dockerfile，安装构建依赖关系并设置ccache以加快重新编译速度。

```
RUN DEBIAN_FRONTEND=noninteractive \
    apt-get install -y --no-install-recommends \
        build-essential \
        ca-certificates \
        ccache \
        cmake \
        git \
```

```
        libjpeg-dev \
        libpng-dev \
    && rm -rf /var/lib/apt/lists/*

# Setup ccache
RUN /usr/sbin/update-ccache-symlinks
RUN mkdir /opt/ccache && ccache --set-config=cache_dir=/opt/ccache
```

b. 安装PyTorch 的常见和 Linux 的依赖关系.

```
# Common dependencies for PyTorch
RUN conda install astunparse numpy ninja pyyaml mkl mkl-include setuptools cmake cffi
 typing_extensions future six requests dataclasses

# Linux specific dependency for PyTorch
RUN conda install -c pytorch magma-cuda113
```

c. 克隆PyTorch GitHub 储库.

```
RUN --mount=type=cache,target=/opt/ccache \
    cd / \
    && git clone --recursive https://github.com/pytorch/pytorch -b v${PYTORCH_VERSION}
```

d. 安装和构建特定的NCCL版本。为此，请替换 PyTorch 默认 NCCL 文件夹中的内容 (`/pytorch/third_party/nccl`) 使用来自 NVIDIA 存储库的特定 NCCL 版本。NCCL 版本是在本指南的步骤 3 中设置的。

```
RUN cd /pytorch/third_party/nccl \
    && rm -rf nccl \
    && git clone https://github.com/NVIDIA/nccl.git -b v${NCCL_VERSION}-1 \
    && cd nccl \
    && make -j64 src.build CUDA_HOME=/usr/local/cuda NVCC_GENCODE="-
gencode=arch=compute_70,code=sm_70 -gencode=arch=compute_80,code=sm_80" \
    && make pkg.txz.build \
    && tar -xvf build/pkg/txz/nccl_*.txz -C $CONDA_PREFIX --strip-components=1
```

e. 构建并安装 PyTorch。完成此过程通常需要稍多时间。它使用上一步中下载的 NCCL 版本构建。

```
RUN cd /pytorch \
    && CMAKE_PREFIX_PATH="$(dirname $(which conda))/../" \
    python setup.py install \
    && rm -rf /pytorch
```

8. 构建和安装AmazonOFI NCCL 插件. 此操作启用Libfabric支持 SageMaker 数据 parallel 库。

```
RUN DEBIAN_FRONTEND=noninteractive apt-get update \
    && apt-get install -y --no-install-recommends \
        autoconf \
        automake \
        libtool
RUN mkdir /tmp/efa-ofi-nccl \
    && cd /tmp/efa-ofi-nccl \
    && git clone https://github.com/aws/aws-ofi-nccl.git -b v${BRANCH_OFI} \
    && cd aws-ofi-nccl \
    && ./autogen.sh \
    && ./configure --with-libfabric=/opt/amazon/efa \
    --with-mpi=/opt/amazon/openmpi \
    --with-cuda=/usr/local/cuda \
    --with-nccl=$CONDA_PREFIX \
    && make \
    && make install \
```

```
    && rm -rf /tmp/efa-ofi-nccl
```

9. 构建和安装TorchVision.

```
RUN pip install --no-cache-dir -U \
    packaging \
    mpi4py==3.0.3
RUN cd /tmp \
    && git clone https://github.com/pytorch/vision.git -b v0.9.1 \
    && cd vision \
    && BUILD_VERSION="0.9.1+cu111" python setup.py install \
    && cd /tmp \
    && rm -rf vision
```

10.安装和配置 OpenSSH。MPI 需要 OpenSSH 才能在容器之间进行通信。允许 OpenSSH 与集装箱交谈，而不提示确认。

```
RUN apt-get update \
    && apt-get install -y  --allow-downgrades --allow-change-held-packages --no-install-
recommends \
    && apt-get install -y --no-install-recommends openssh-client openssh-server \
    && mkdir -p /var/run/sshd \
    && cat /etc/ssh/ssh_config | grep -v StrictHostKeyChecking > /etc/ssh/ssh_config.new
 \
    && echo "    StrictHostKeyChecking no" >> /etc/ssh/ssh_config.new \
    && mv /etc/ssh/ssh_config.new /etc/ssh/ssh_config \
    && rm -rf /var/lib/apt/lists/*

# Configure OpenSSH so that nodes can communicate with each other
RUN mkdir -p /var/run/sshd && \
 sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -i /etc/
pam.d/sshd
RUN rm -rf /root/.ssh/ && \
 mkdir -p /root/.ssh/ && \
 ssh-keygen -q -t rsa -N '' -f /root/.ssh/id_rsa && \
 cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys \
 && printf "Host *\n StrictHostKeyChecking no\n" >> /root/.ssh/config
```

11.安装 PT S3 插件以高效访问 Amazon S3 中的数据集。

```
RUN pip install --no-cache-dir -U ${PT_S3_WHL_GPU}
RUN mkdir -p /etc/pki/tls/certs && cp /etc/ssl/certs/ca-certificates.crt /etc/pki/tls/
certs/ca-bundle.crt
```

12.安装libboost库. 该软件包是联网的异步 IO 功能所需的 SageMaker 数据 parallel 库。

```
WORKDIR /
RUN wget https://sourceforge.net/projects/boost/files/boost/1.73.0/boost_1_73_0.tar.gz/
download -O boost_1_73_0.tar.gz \
    && tar -xzf boost_1_73_0.tar.gz \
    && cd boost_1_73_0 \
    && ./bootstrap.sh \
    && ./b2 threading=multi --prefix=${CONDA_PREFIX} -j 64 cxxflags=-fPIC cflags=-fPIC
 install || true \
    && cd .. \
    && rm -rf boost_1_73_0.tar.gz \
    && rm -rf boost_1_73_0 \
    && cd ${CONDA_PREFIX}/include/boost
```

13.安装以下内容 SageMaker 用于的工具 PyTorch 训练。

```
WORKDIR /root
```

```
RUN pip install --no-cache-dir -U \
    smclarify \
    "sagemaker>=2,<3" \
    sagemaker-experiments==0.* \
    sagemaker-pytorch-training
```

14.最后，安装 SageMaker 数据 parallel 二进制文件和其余依赖项。

```
RUN --mount=type=cache,id=apt-final,target=/var/cache/apt \
  apt-get update && apt-get install -y  --no-install-recommends \
  jq \
  libhwloc-dev \
  libnuma1 \
  libnuma-dev \
  libssl1.1 \
  libtool \
  hwloc \
  && rm -rf /var/lib/apt/lists/*

RUN SMDATAPARALLEL_PT=1 pip install --no-cache-dir ${SMDATAPARALLEL_BINARY}
```

15.创建完 Dockerfile 后，请参阅修改自己的训练容器了解如何构建 Docker 容器，将其托管在 Amazon ECR 中，然后使用 SageMaker Python 开发工具包。

以下示例代码显示了组合之前的所有代码块之后的完整 Dockerfile。

```
# This file creates a docker image with minimum dependencies to run SageMaker data parallel
 training
FROM nvidia/cuda:11.3.1-cudnn8-devel-ubuntu20.04

# Set appropiate versions and location for components
ARG PYTORCH_VERSION=1.10.2
ARG PYTHON_SHORT_VERSION=3.8
ARG EFA_VERSION=1.14.1
ARG SMDATAPARALLEL_BINARY=https://smdataparallel.s3.amazonaws.com/binary/pytorch/
${PYTORCH_VERSION}/cu113/2022-02-18/smdistributed_dataparallel-1.4.0-cp38-cp38-
linux_x86_64.whl
ARG PT_S3_WHL_GPU=https://aws-s3-plugin.s3.us-west-2.amazonaws.com/binaries/0.0.1/1c3e69e/
awsio-0.0.1-cp38-cp38-manylinux1_x86_64.whl
ARG CONDA_PREFIX="/opt/conda"
ARG BRANCH_OFI=1.1.3-aws

# Set ENV variables required to build PyTorch
ENV TORCH_CUDA_ARCH_LIST="3.7 5.0 7.0+PTX 8.0"
ENV TORCH_NVCC_FLAGS="-Xfatbin -compress-all"
ENV NCCL_VERSION=2.10.3

# Add OpenMPI to the path.
ENV PATH /opt/amazon/openmpi/bin:$PATH

# Add Conda to path
ENV PATH $CONDA_PREFIX/bin:$PATH

# Set this enviroment variable for SageMaker to launch SMDDP correctly.
ENV SAGEMAKER_TRAINING_MODULE=sagemaker_pytorch_container.training:main

# Add enviroment variable for processes to be able to call fork()
ENV RDMAV_FORK_SAFE=1

# Indicate the container type
ENV DLC_CONTAINER_TYPE=training

# Add EFA and SMDDP to LD library path
```

```
ENV LD_LIBRARY_PATH="/opt/conda/lib/python${PYTHON_SHORT_VERSION}/site-packages/
smdistributed/dataparallel/lib:$LD_LIBRARY_PATH"
ENV LD_LIBRARY_PATH=/opt/amazon/efa/lib/:$LD_LIBRARY_PATH

# Install basic dependencies to download and build other dependencies
RUN --mount=type=cache,id=apt-final,target=/var/cache/apt \
  apt-get update && apt-get install -y  --no-install-recommends \
  curl \
  wget \
  git \
  && rm -rf /var/lib/apt/lists/*

# Install EFA.
# This is required for SMDDP backend communication
RUN DEBIAN_FRONTEND=noninteractive apt-get update
RUN mkdir /tmp/efa \
    && cd /tmp/efa \
    && curl --silent -O https://efa-installer.amazonaws.com/aws-efa-installer-
${EFA_VERSION}.tar.gz \
    && tar -xf aws-efa-installer-${EFA_VERSION}.tar.gz \
    && cd aws-efa-installer \
    && ./efa_installer.sh -y --skip-kmod -g \
    && rm -rf /tmp/efa

# Install Conda
RUN curl -fsSL -v -o ~/miniconda.sh -O  https://repo.anaconda.com/miniconda/Miniconda3-
latest-Linux-x86_64.sh  && \
    chmod +x ~/miniconda.sh && \
    ~/miniconda.sh -b -p $CONDA_PREFIX && \
    rm ~/miniconda.sh && \
    $CONDA_PREFIX/bin/conda install -y python=${PYTHON_SHORT_VERSION} conda-build pyyaml
 numpy ipython && \
    $CONDA_PREFIX/bin/conda clean -ya

# Install PyTorch.
# Start with dependencies listed in official PyTorch dockerfile
# https://github.com/pytorch/pytorch/blob/master/Dockerfile
RUN DEBIAN_FRONTEND=noninteractive \
    apt-get install -y --no-install-recommends \
        build-essential \
        ca-certificates \
        ccache \
        cmake \
        git \
        libjpeg-dev \
        libpng-dev && \
    rm -rf /var/lib/apt/lists/*

# Setup ccache
RUN /usr/sbin/update-ccache-symlinks
RUN mkdir /opt/ccache && ccache --set-config=cache_dir=/opt/ccache

# Common dependencies for PyTorch
RUN conda install astunparse numpy ninja pyyaml mkl mkl-include setuptools cmake cffi
 typing_extensions future six requests dataclasses

# Linux specific dependency for PyTorch
RUN conda install -c pytorch magma-cuda113

# Clone PyTorch
RUN --mount=type=cache,target=/opt/ccache \
    cd / \
    && git clone --recursive https://github.com/pytorch/pytorch -b v${PYTORCH_VERSION}
# Note that we need to use the same NCCL version for PyTorch and OFI plugin.
# To enforce that, install NCCL from source before building PT and OFI plugin.
```

```
# Install NCCL.
# Required for building OFI plugin (OFI requires NCCL's header files and library)
RUN cd /pytorch/third_party/nccl \
    && rm -rf nccl \
    && git clone https://github.com/NVIDIA/nccl.git -b v${NCCL_VERSION}-1 \
    && cd nccl \
    && make -j64 src.build CUDA_HOME=/usr/local/cuda NVCC_GENCODE="-
gencode=arch=compute_70,code=sm_70 -gencode=arch=compute_80,code=sm_80" \
    && make pkg.txz.build \
    && tar -xvf build/pkg/txz/nccl_*.txz -C $CONDA_PREFIX --strip-components=1

# Build and install PyTorch.
RUN cd /pytorch \
    && CMAKE_PREFIX_PATH="$(dirname $(which conda))/../" \
    python setup.py install \
    && rm -rf /pytorch

RUN ccache -C

# Build and install OFI plugin. \
# It is required to use libfabric.
RUN DEBIAN_FRONTEND=noninteractive apt-get update \
    && apt-get install -y --no-install-recommends \
        autoconf \
        automake \
        libtool
RUN mkdir /tmp/efa-ofi-nccl \
    && cd /tmp/efa-ofi-nccl \
    && git clone https://github.com/aws/aws-ofi-nccl.git -b v${BRANCH_OFI} \
    && cd aws-ofi-nccl \
    && ./autogen.sh \
    && ./configure --with-libfabric=/opt/amazon/efa \
        --with-mpi=/opt/amazon/openmpi \
        --with-cuda=/usr/local/cuda \
        --with-nccl=$CONDA_PREFIX \
    && make \
    && make install \
    && rm -rf /tmp/efa-ofi-nccl

# Build and install Torchvision
RUN pip install --no-cache-dir -U \
    packaging \
    mpi4py==3.0.3
RUN cd /tmp \
    && git clone https://github.com/pytorch/vision.git -b v0.9.1 \
    && cd vision \
    && BUILD_VERSION="0.9.1+cu111" python setup.py install \
    && cd /tmp \
    && rm -rf vision

# Install OpenSSH.
# Required for MPI to communicate between containers, allow OpenSSH to talk to containers
 without asking for confirmation
RUN apt-get update \
    && apt-get install -y  --allow-downgrades --allow-change-held-packages --no-install-
recommends \
    && apt-get install -y --no-install-recommends openssh-client openssh-server \
    && mkdir -p /var/run/sshd \
    && cat /etc/ssh/ssh_config | grep -v StrictHostKeyChecking > /etc/ssh/ssh_config.new \
    && echo "    StrictHostKeyChecking no" >> /etc/ssh/ssh_config.new \
    && mv /etc/ssh/ssh_config.new /etc/ssh/ssh_config \
    && rm -rf /var/lib/apt/lists/*
# Configure OpenSSH so that nodes can communicate with each other
RUN mkdir -p /var/run/sshd && \
    sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -i /
etc/pam.d/sshd
```

```
RUN rm -rf /root/.ssh/ && \
    mkdir -p /root/.ssh/ && \
    ssh-keygen -q -t rsa -N '' -f /root/.ssh/id_rsa && \
    cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys \
    && printf "Host *\n StrictHostKeyChecking no\n" >> /root/.ssh/config

# Install PT S3 plugin.
# Required to efficiently access datasets in Amazon S3
RUN pip install --no-cache-dir -U ${PT_S3_WHL_GPU}
RUN mkdir -p /etc/pki/tls/certs && cp /etc/ssl/certs/ca-certificates.crt /etc/pki/tls/
certs/ca-bundle.crt

# Install libboost from source.
# This package is needed for smdataparallel functionality (for networking asynchronous IO).
WORKDIR /
RUN wget https://sourceforge.net/projects/boost/files/boost/1.73.0/boost_1_73_0.tar.gz/
download -O boost_1_73_0.tar.gz \
    && tar -xzf boost_1_73_0.tar.gz \
    && cd boost_1_73_0 \
    && ./bootstrap.sh \
    && ./b2 threading=multi --prefix=${CONDA_PREFIX} -j 64 cxxflags=-fPIC cflags=-fPIC
 install || true \
    && cd .. \
    && rm -rf boost_1_73_0.tar.gz \
    && rm -rf boost_1_73_0 \
    && cd ${CONDA_PREFIX}/include/boost

# Install SageMaker PyTorch training.
WORKDIR /root
RUN pip install --no-cache-dir -U \
    smclarify \
    "sagemaker>=2,<3" \
    sagemaker-experiments==0.* \
    sagemaker-pytorch-training

# Install SageMaker data parallel binary (SMDDP)
# Start with dependencies
RUN --mount=type=cache,id=apt-final,target=/var/cache/apt \
    apt-get update && apt-get install -y  --no-install-recommends \
        jq \
        libhwloc-dev \
        libnuma1 \
        libnuma-dev \
        libssl1.1 \
        libtool \
        hwloc \
    && rm -rf /var/lib/apt/lists/*

# Install SMDDP
RUN SMDATAPARALLEL_PT=1 pip install --no-cache-dir ${SMDATAPARALLEL_BINARY}
```

Tip

有关在 SageMaker 中为训练创建自定义 Dockerfile 的更多常规信息，请参阅使用您自己的训练算法.

Tip

如果你想扩展自定义 Dockerfile 以纳入 SageMaker 模型 parallel 库，请参阅使用创建自己的 Docker 容器 SageMaker 分布式模型并行库 (p. 2019).

SageMaker 分布式数据并行配置提示和陷阱

在使用 SageMaker 的分布式数据并行库之前，请查看以下提示和陷阱。此列表包括适用于各个框架的提示。

主题

- 数据预处理 (p. 1993)
- 单节点与多个节点 (p. 1993)
- 使用调试器调试扩展效率 (p. 1993)
- Batch 大小 (p. 1993)
- 自定义 MPI 选项 (p. 1993)
- 使用 Amazon FSx 并设置最佳存储和吞吐量容量 (p. 1994)

数据预处理

如果在训练期间使用利用 CPU 的外部库预处理数据，则可能会遇到 CPU 瓶颈，因为 SageMaker 分布式数据并行使用 CPU`AllReduce`运算。您可以通过将预处理步骤移至使用 GPU 的库或在训练之前完成所有预处理来缩短训练时间。

单节点与多个节点

建议将此库与多个节点结合使用。该库可以与单主机、多设备设置一起使用（例如，具有多个 GPU 的单个 ML 计算实例）；但是，当您使用两个或更多节点时，库的`AllReduce`操作可以显著提高性能。此外，在单个主机上，nvLink 已经为节点内做出贡献`AllReduce`效率。

使用调试器调试扩展效率

您可以使用 Amazon SageMaker 调试器在训练期间监控和可视化 CPU 和 GPU 利用率以及其他感兴趣的指标。你可以使用调试器内置规则监控计算性能问题，例如`CPUBottleneck`、`LoadBalancing`, 和`LowGPUUtilization`. 你可以使用指定这些规则调试程序配置当您定义 Amazon SageMaker Python SDK 估算器时。如果您使用AmazonCLI 和AmazonBoto3 在 SageMaker 上进行训练，您可以启用调试器，如中所示使用 Amazon SageMaker API 配置调试程序.

要在 SageMaker 培训作业中查看使用调试器的示例，您可以在SageMaker 笔记本示例 GitHub 库. 要了解有关 Debug 的更多信息，请参阅Amazon SageMaker 调试器.

Batch 大小

在分布式训练中，随着更多节点的增加，批量大小应该按比例增加。为了在训练作业中添加更多节点并增加全球批量时提高收敛速度，请提高学习率。

实现这一目标的一种方法是使用渐进的学习率预热，随着培训工作的进展，学习率从小值提高到大值。这种坡道避免了学习率的突然提高，从而在训练开始时能够健康地融合。例如，您可以使用线性缩放规则其中每次小批量大小乘以 k 时，学习率也乘以 k。要了解有关此技术的更多信息，请参阅该研究论文，准确、大型迷你匹配 SGD：在 1 小时内训练 ImageNet，第 2 和 3 节。

自定义 MPI 选项

SageMaker 分布式数据并行库采用消息传递接口 (MPI)，这是管理高性能群集中节点之间通信的常用标准，并使用 NVIDIA 的 NCCL 库进行 GPU 级通信。当你将数据并行库与 TensorFlow 或 Pytorch 一起使用时`Estimator`，各自的容器设置 MPI 环境并执行`mpirun`命令在群集节点上启动作业。

您可以使用`custom_mpi_options`中的参数`Estimator`. 任何`mpirun`在此字段中传递的标志将添加到`mpirun`命令并由 SageMaker 执行以进行训练。例如，您可以将`distribution`的参数`Estimator`使用以下内容来使用`NCCL_DEBUG`变量来在程序开始时打印 NCCL 版本：

```
distribution = {'smdistributed':{'dataparallel':{'enabled': True, "custom_mpi_options": "-
verbose -x NCCL_DEBUG=VERSION"}}}
```

使用 Amazon FSx 并设置最佳存储和吞吐量容量

在具有分布式数据并行性的多个节点上训练模型时，强烈建议使用FSx for Lustre. Amazon FSx 是一种可扩展且高性能的存储服务，支持共享文件存储，吞吐量更快。大规模使用 Amazon FSx 存储，您可以在计算节点之间实现更快的数据加载速度。

通常情况下，使用分布式数据并行性，您预计总训练吞吐量将与 GPU 的数量接近线性扩展。但是，如果您使用次优的 Amazon FSx 存储空间，则由于 Amazon FSx 吞吐量较低，训练绩效可能会降低。

例如，如果您使用SCRATCH_2Amazon FSx 文件系统的部署类型使用最低 1.2 TiB 存储容量，I/O 吞吐量为 240 MB/s。Amazon FSx 存储的工作方式是，您可以分配物理存储设备，分配的设备越多，获得的吞吐量就越大。SRATCH_2 类型的最小存储增量为 1.2 TiB，相应的吞吐量增益为 240 MB/s。

假设您有一个模型可以在超过 100 GB 数据集的 4 节点群集上进行训练。对于针对集群进行了优化的给定批量大小，假设模型可以在大约 30 秒内完成一个纪元。在这种情况下，所需的最低 I/O 速度约为 3 Gb/s（100 GB /30 秒）。这显然比 240 MB/s 的吞吐量要高得多。由于 Amazon FSx 容量如此有限，将分布式培训作业扩展到更大的集群可能会加剧 I/O 瓶颈问题；随着缓存的积累，模型训练吞吐量可能会在稍后时期提高，但 Amazon FSx 吞吐量仍可能是瓶颈。

为了缓解此类 I/O 瓶颈问题，您应该增加 Amazon FSx 存储大小以获得更高的吞吐量容量。通常，为了找到最佳 I/O 吞吐量，您可以尝试不同的 Amazon FSx 吞吐量，分配的吞吐量等于或略低于估计值，直到您发现这足以解决 I/O 瓶颈问题为止。在上述示例中，具有 2.4 Gb/s 吞吐量和 67 GB RAM 缓存的 Amazon FSx 存储就足够了。如果文件系统具有最佳吞吐量，模型训练吞吐量应立即达到最大值，也可以在缓存建立起来的第一个纪元之后达到最大值。

要了解有关如何增加 Amazon FSx 存储和部署类型的更多信息，请参阅Amazon FSx for Lustre 文档：

- 如何增加存储容量
- 聚合文件系统性能

亚马逊SageMaker数据并行库常见问题

使用以下内容查找有关的常见问题的解答SageMaker的数据并行度库。

Q: 使用图书馆时，怎么样**`allreduce`**-支持管理的 CPU 实例？ 我必须创建异构 CPU-GPU 集群吗？还是SageMaker服务为使用库的作业创建额外的 C5s？

该库使用 GPU 实例可用的 CPU。不会启动额外的 C5 或 CPU 实例；如果SageMaker培训作业是 8 节点`ml.p3dn.24xlarge`群集，只有 8 个`ml.p3dn.24xlarge`使用实例。没有配置其他实例。

Q: 我有一份培训工作需要 5 天的时间**`ml.p3.24xlarge`**具有一组超参数 H1（学习速率、批量大小、优化器等）的实例。在使用SageMaker的数据并行度库和大五倍的集群足以实现大约五次的加速？ 或者在激活库之后，我必须重新审视它的训练超参数吗？

库改变了整体批量大小。新的总批量大小随使用的训练实例的数量进行线性扩展。因此，必须更改超参数，例如学习率，以确保收敛。

Q: 图书馆支持 Spot 吗？

可以。您可以使用托管的现场训练。您可以在SageMaker训练任务。您可以在训练脚本中保存和恢复检查点，如最后步骤中提到的the section called “TensorFlow” (p. 1977)和the section called “PyTorch” (p. 1979).

Q: 库在单主机、多设备设置中是否相关？

该库可用于单主机多设备培训，但该库仅在多主机培训中提供性能改进。

Q: 图书馆可以用PyTorch闪电？

否。 但是，使用图书馆的 DDPPyTorch，你可以编写自定义 DDP 来实现该功能。

Q: 训练数据集应该存储在哪里？

Amazon S3 存储桶或 Amazon FSx 驱动器中可存储训练数据集。请参阅这个培训作业的各种受支持的输入文件系统的文档.

Q: 使用图书馆时，是否必须在 FSx 中拥有 Lustre 的训练数据？ 可以使用亚马逊 EFS 和 Amazon S3 吗？

我们通常建议您使用 Amazon FSx，因为它的延迟更低，吞吐量更高。如果您愿意，可以使用 Amazon EFS 或 Amazon S3。

Q: 库可以与 CPU 节点一起使用吗？

否。 该图书馆支持`ml.p3.16xlarge`、`ml.p3dn.24xlarge`, 和`ml.p4d.24xlarge`此时的实例。

Q: 发布时库目前支持哪些框架和框架版本？

该库目前支持PyTorch和 v1.6.0 或后续版本TensorFlowv2.3.0 或后续版本。它不支持TensorFlow1.x。有关封装在哪个版本库中的更多信息Amazon深度学习容器，请参阅Deep Learning Containers 的发行说明.

Q: 图书馆支持 AMP 吗？

是，SageMaker的分布式数据并行库支持开箱即用的自动混合精度 (AMP)。除了对训练脚本进行框架级修改之外，无需采取额外操作即可使用 AMP。如果渐变位于 FP16 中，则SageMaker数据并行度库运行`AllReduce`在 FP16 中操作。有关在训练脚本中实施 AMP API 的更多信息，请参阅以下资源：

- 框架-PyTorch中的NVIDIA 深度学习性能文档
- 框架-TensorFlow中的NVIDIA 深度学习性能文档
- 深度学习的自动混合精度中的NVIDIA 开发者文档
- 介绍本机PyTorch自动混合精度可加快 NVIDIA GPU 上的训练速度中的PyTorch博客
- TensorFlow混合精度 API中的TensorFlow文档

Q: 如何确定我的分布式培训作业是否因为 I/O 瓶颈而减慢了速度？

对于较大的集群，训练作业需要更多的 I/O 吞吐量，因此，培训吞吐量可能需要更长的时间（更长的时间）才能提高到最高性能。这表明，随着节点扩展，I/O 正在受到瓶颈，缓存更难构建（吞吐量要求更高，网络拓扑更复杂）。有关监控 Amazon FSx 吞吐量的更多信息，请参阅CloudWatch请参阅监控 FSx for Lustre中的FSx for Lustre 用户指南.

Q: 在运行具有数据并行性的分布式培训作业时，如何解决 I/O 瓶颈？

如果您使用的是 Amazon S3，我们强烈建议您使用 Amazon FSx 作为数据渠道。如果您已经在使用 Amazon FSx 但仍然存在 I/O 瓶颈问题，则可能已将 Amazon FSx 文件系统设置为低 I/O 吞吐量和较小的存储容量。有关如何估算和选择正确的 I/O 吞吐量大小的更多信息，请参阅使用 Amazon FSx 并设置最佳存储和吞吐量容量 (p. 1994).

Q:（对于库 v1.4.0 或更高版本）我如何解决**`Invalid backend`**初始化进程组时出错。

如果遇到错误消息`ValueError: Invalid backend: 'smddp'`打电话时`init_process_group`，这是由于 v1.4.0 及更高版本的库发生了突破性变化。您必须导入PyTorch图书馆的客户`smdistributed.dataparallel.torch.torch_smddp`，其中注册`smddp`作为后端PyTorch. 要了解更多信息，请参阅 使用 SageMaker 作为后端的分布式数据并行库`torch.distributed` (p. 1980)。

Q:（对于图书馆 v1.4.0 或更高版本）我想称之为torch.分布式/分布界面。哪些原语做**`smddp`**后端支持？

在 v1.4.0 中，库支持`all_reduce`、`broadcast`、`reduce`、`all_gather`, 和`barrier`.

Q:（对于库 v1.4.0 或更高版本）此新 API 是否可以与其他自定义 DDP 类或类似 Apex DDP 之类的库配合使用？

这些区域有：SageMaker数据并行库通过其他第三方分布式数据并行库和框架实现进行测试，这些库使用`torch.distribtued`模块。使用SageMaker只要库支持自定义 DDP 类使用的集合，具有自定义 DDP 类的数据并行库就可以正常工作。有关受支持的集体列表，请参阅前面的问题。如果您有这些使用案例并需要进一步的支持，请联系SageMaker通过Amazon支持中心要么Amazon亚马逊开发者论坛SageMaker.

Q: 图书馆是否支持bring-your-own-容器（BYOC）选项？ 如果是这样，我该如何通过编写自定义 Dockerfile 来安装库并运行分布式培训作业？

如果要集成SageMaker数据并行库及其在自己的 Docker 容器中的最低依赖关系，BYOC 是正确的方法。您可以使用库的二进制文件构建自己的容器。建议的过程是编写包含库及其依赖项的自定义 Dockerfile、构建 Docker 容器、将其托管在 Amazon ECR 中，然后使用 ECR 映像 URI 启动训练作业SageMaker通用估计器类。有关如何为分布式培训准备自定义 Dockerfile 的更多说明，请参阅SageMaker使用SageMaker数据并行库，请参阅使用创建自己的 Docker 容器 SageMaker 分布式数据并行库 (p. 1985).

数据并行排查

如果在使用库时运行训练作业时遇到问题，请使用以下列表尝试进行故障排除。如果你需要进一步的支持，请联系SageMaker团队通过Amazon支持中心要么Amazon亚马逊开发者论坛亚马逊SageMaker.

主题

- 使用SageMaker与亚马逊并行分布式数据SageMaker调试程序和检查点 (p. 1996)
- 附加到模型参数键的意外前缀 (p. 1997)
- SageMaker初始化期间分布式培训 Job 停滞 (p. 1997)
- SageMaker培 Job 结束时的分布式培训作业停滞 (p. 1997)
- 观察由于 Amazon FSx 吞吐量瓶颈而导致的扩展效率下降 (p. 1998)
- SageMaker分布式训练 JobPyTorch返回弃用警告 (p. 1998)

使用SageMaker与亚马逊并行分布式数据SageMaker调试程序和检查点

监控系统瓶颈、分析框架操作并调试模型输出张量以用于训练作业SageMaker并行分布式数据，使用亚马逊SageMaker调试程序。

但是，当你使用SageMaker调试程序，SageMaker并行分布式数据，以及SageMaker检查点时，您可能会看到类似于以下示例的错误。

```
SMDebug Does Not Currently Support Distributed Training Jobs With Checkpointing Enabled
```

这是由于调试器和检查点之间存在内部错误，启用时会发生这种错误SageMaker并行分布式数据。

- 如果启用所有三个功能，SageMakerPython SDK 通过传递自动关闭调试器`debugger_hook_config=False`，该框架等效于以下框架`estimator`示例。

```
bucket=sagemaker.Session().default_bucket()
base_job_name="sagemaker-checkpoint-test"
checkpoint_in_bucket="checkpoints"

# The S3 URI to store the checkpoints
checkpoint_s3_bucket="s3://{}/{}/{}".format(bucket, base_job_name, checkpoint_in_bucket)

estimator = TensorFlow(
    ...

    distribution={"smdistributed": {"dataparallel": { "enabled": True }}},
    checkpoint_s3_uri=checkpoint_s3_bucket,
    checkpoint_local_path="/opt/ml/checkpoints",
    debugger_hook_config=False
)
```

- 如果你想继续使用两者SageMaker并行分布式数据SageMaker调试器，一种解决方法是手动将检查点函数添加到训练脚本中，而不是指定`checkpoint_s3_uri`和`checkpoint_local_path`来自评估程序的参数。有关在训练脚本中设置手动检查点的更多信息，请参阅保存检查点 (p. 2041).

附加到模型参数键的意外前缀

适用于PyTorch分布式培训作业，意想不到的前缀 (`model`例如）可能会附加到`state_dict`键（模型参数）。这些区域有：SageMaker在以下情况下，数据并行库不会直接更改或加上任何模型参数名称PyTorch训练作业可以保存模型工件。这些区域有：PyTorch的分布式培训改变了`state_dict`通过网络，在前缀之前。如果您在使用时由于参数名称不同而遇到任何模型故障问题SageMaker数据并行库和检查点PyTorch训练中调整下面的示例代码，以便在训练脚本中加载检查点的步骤中删除前缀。

```
state_dict = {k.partition('model.')[2]:state_dict[k] for k in state_dict.keys()}
```

这需要每个`state_dict`key 作为字符串值，在第一次出现时将字符串分隔`'model.'`，并获取分区字符串的第三个列表项（含索引 2）。

有关前缀问题的更多信息，请参阅以下内容的讨论话题：如果通过 Multi-GPU 训练，则在保存模型中加上参数名称中的PyTorch论坛.

有关的更多信息PyTorch保存和加载模型的方法，请参阅跨设备保存和加载模型中的PyTorch文档.

SageMaker初始化期间分布式培训 Job 停滞

如果您的SageMaker使用启用 EFA 的实例时，在初始化期间分布式数据并行训练作业停顿 (`ml.p3dn.24xlarge`和`ml.p4d.24xlarge`)，这可能是由于用于培训作业的 VPC 子网的安全组中配置错误造成的。EFA 需要适当的安全组配置才能启用节点之间的流量。

为安全组配置入站和出站规则

1. 登录到Amazon Web Services Management Console并打开 Amazon VPC 控制台，网址：https://console.aws.amazon.com/vpc/。
2. 选择个安全组在左侧导航窗格中。
3. 选择与您用于培训的 VPC 子网绑定的安全组。
4. 在详细信息部分中，复制安全组 ID.
5. 在 Inbound Rules (入站规则) 选项卡上，选择 Edit inbound rules (编辑入站规则)。
6. 在 Edit inbound rules (编辑入站规则) 页面上，执行以下操作：
   a. 选择 Add rule。
   b. 对于 Type (类型)，请选择 All traffic (所有流量)。
   c. 适用于源，选择Custom (自定义)中，将安全组 ID 粘贴到搜索框中，然后选择弹出的安全组。
7. 选择保存规则以完成安全组的入站规则配置。
8. 在存储库的出站规则选项卡，选择编辑出站规则.
9. 重复步骤 6 和 7 以添加与出站规则相同的规则。

完成上述使用入站和出站规则配置安全组的步骤后，请重新运行培训作业并验证停滞问题是否已解决。

有关为 VPC 和 EFA 配置安全组的更多信息，请参阅您的 VPC 的安全组和Elastic Fabric Adapter.

SageMaker培 Job 结束时的分布式培训作业停滞

培训结束时问题停滞的根本原因之一是不同等级中每个时代处理的批次数不匹配。所有 Worker (GPU) 在向后传递中同步其本地渐变，以确保在批处理迭代结束时他们都拥有相同的模型副本。如果在培训的最后一个时期，批量大小分配给不同的员工组不均衡，则培训作业就会停顿。例如，当一组工作人员（A 组）完成处理所有批次并退出训练循环时，另一组工作人员（B 组）开始处理另一个批次，仍然期望来自 A 组的通信以同步渐变。这会导致 B 组等待 A 组，该组已经完成训练，没有任何梯度要同步。

因此，在设置训练数据集时，重要的是每个工作人员获得相同数量的数据样本，以便每个工作人员在训练时经历相同数量的批次。确保每个排名获得相同数量的批次，以避免这种停滞的问题。

观察由于 Amazon FSx 吞吐量瓶颈而导致的扩展效率下降

缩放效率降低的一个潜在原因是 FSx 吞吐量限制。如果在切换到更大的训练集群时观察到扩展效率突然下降，请尝试使用具有更高吞吐量限制的 Lustre 文件系统的更大 FSx。有关更多信息，请参阅 。聚合文件系统性能和管理存储和吞吐量容量中的Amazon FSx for Lustre 用户指南.

SageMaker分布式训练 JobPyTorch返回弃用警告

自 1.4.0 版本以来，SageMaker分布式数据并行度库可作为PyTorch分布式。由于使用图书馆的重大变化PyTorch，您可能会遇到警告消息，指出`smdistributed`适用于PyTorch已弃用分布式软件包。警告消息应该类似于以下内容：

```
smdistributed.dataparallel.torch.dist is deprecated in the SageMaker distributed data
 parallel library v1.4.0+.
Please use torch.distributed and specify 'smddp' as a backend when initializing process
 group as follows:
torch.distributed.init_process_group(backend='smddp')
For more information, see the library's API documentation at
https://docs.aws.amazon.com/sagemaker/latest/dg/data-parallel-modify-sdp-pt.html
```

在 v1.4.0 及更高版本中，该库只需在训练脚本的顶部导入一次，然后在PyTorch分布式初始化。使用单行后端规范，您可以保留PyTorch训练脚本不变并直接使用PyTorch分布式模块。请参阅修改 PyTorch 训练脚本 (p. 1979)了解突破性变化以及使用图书馆的新方法PyTorch.

SageMaker的分布式模型并行

使用 AmazonSageMaker的分布式模型并行库，用于训练由于 GPU 内存限制而难以训练的大型深度学习 (DL) 模型。该库自动高效地将模型拆分到多个 GPU 和实例之间。使用该库，您可以通过高效训练具有数十亿或万亿参数的大型 DL 模型，更快地实现目标预测准确性。

你可以使用库自动分区你自己的TensorFlow和PyTorch跨多个 GPU 和多个节点的模型只需最少的代码更改。您可以通过SageMakerPython 开发工具包。

有关模型并行性的更多信息，请参阅以下部分。SageMaker模型并行库。该库的 API 文档位于分布式训练API中的SageMakerPython 开发工具包.

要跟踪库的最新更新，请参阅SageMaker分布式模型并行发行说明中的SageMakerPython 开发工具包.

主题

- 模型并行简介 (p. 1998)
- SageMaker 模型并行库的核心功能 (p. 1999)
- 运行 SageMaker 具有模型并行性的分布式培训 Job (p. 2003)
- PyTorch 的 SageMaker 模型并行库的扩展功能 (p. 2021)
- SageMaker 分布式模型并行最佳实 (p. 2036)
- SageMaker 分布式模型并行配置提示和陷阱 (p. 2038)
- 模型并行疑难解 (p. 2040)

模型并行简介

模型并行是在多个设备或节点（例如配备 GPU 的实例）之间拆分模型并创建一个高效的管道来跨这些设备训练模型以最大限度地提高 GPU 利用率的过程。

使用本页上的各节可以了解有关模型并行性的更多信息，包括如何使用它来克服训练大型机器学习模型时出现的问题，以及将其集成到 ML 训练脚本时的重要注意事项。

什么是模型并行性？

增加深度学习模型大小（图层和参数）可以提高准确性。但是，您可以在单个 GPU 中适合的最大型号大小有限制。在训练深度学习模型时，GPU 内存限制可能会成为以下方面的瓶颈：

- 他们可以限制你训练的模型的大小。鉴于较大的模型往往实现更高的准确度，这直接转化为训练的模型准确度。
- 它们可以限制您训练的批量大小，从而降低 GPU 利用率和减慢训练速度。

为了克服与在单个 GPU 上训练模型相关的限制，可以使用模型并行性在多个计算设备上分发和训练模型。

使用模型并行性时的重要注意事项

使用模型并行机制时，必须考虑以下几点：

- 你如何在不同设备之间拆分模型：模型的计算图、模型参数和激活的大小以及资源限制（例如时间与内存）决定了最佳的分区策略。

  为了减少高效拆分模型所需的时间和精力，您可以使用 Amazon SageMaker 的分布式模型并行库提供的自动模型拆分功能。
- 实现并行：模型训练（即向前计算和向后传播）本质上是顺序的，每个操作都必须等待其输入由另一个操作计算。出于这个原因，深度学习训练的前向和向后通过阶段不容易并行化，而且天真地将模型拆分到多个 GPU 可能会导致设备利用率低下。例如，GPU 上的图层`i+1`必须等待 GPU 上的图层的输出`i`，所以 GPU`i+1`在此等待期间仍处于闲置状态。

  模型并行库可以通过构建高效的计算计划来实现真正的并行化，方法是实现流水线执行，使不同的设备可以同时处理不同的数据样本的前向和向后传递。

要了解如何使用库在训练期间有效地在各种设备之间拆分模型并提高设备利用率，请参阅SageMaker 模型并行库的核心功能 (p. 1999).

SageMaker 模型并行库的核心功能

Amazon SageMaker 的模型并行库通过提供自动模型拆分和复杂的管道执行计划，使模型并行性更易于访问。模型拆分算法可以针对速度或内存消耗进行优化。该库还支持手动分区。当您使用库时，训练将以流水线方式通过微批量执行，以最大限度地提高 GPU 的使用率。

在创建训练脚本并定义 SageMaker 时，可以使用几行代码来配置这些功能`PyTorch`要么`TensorFlow`估计器。有关库的这些核心功能的更多信息，请参阅以下部分。

> Note
>
> SageMaker 分布式培训库只能通过AmazonSageMaker 培训平台中的 TensorFlow、PyTorch 和 HuggingFace 框架的深度学习容器。要使用这些库，您必须使用 SageMaker Python SDK 或者通过适用 SDK for Python (Boto3) 使用 SageMaker API 或Amazon Command Line Interface. 在整个文档中，说明和示例都重点介绍了如何将分布式培训库与 SageMaker Python SDK 结合使用。

自动拆分模型

当您使用 SageMaker 的模型并行库时，您可以利用自动拆分模型，也称作自动模型分区. 该库使用了一种分区算法，该算法可以平衡内存、最大限度地减少设备之间的通信并优化性能。您可以配置自动分区算法以针对速度或内存进行优化。

此外，您也可以使用手动拆分模型。我们建议自动分割模型，除非您非常熟悉模型架构并且对如何有效地对模型进行分区有一个很好的想法。

工作方式

自动分区是在第一个训练步骤中进行的，当时`smp.step`-装饰功能首先被调用。在这次调用过程中，库首先在 CPU RAM 上构建模型的版本（以避免 GPU 内存限制），然后分析模型图并做出分区决策。根据这一决定，每个模型分区都加载到 GPU 上，然后才执行第一步。由于这些分析和分区步骤，第一个训练步骤可能需要更长的时间。

在任何一个框架中，库都通过自己的后端管理设备之间的通信，后者经过优化Amazon基础设施。

自动分区设计适应框架的特征，库在每个框架中更自然的粒度级别进行分区。例如，在 TensorFlow 中，每个特定操作都可以分配给不同的设备，而在 PyTorch 中，分配是在模块级别完成的，其中每个模块由多个操作组成。以下部分回顾每个框架中的设计细节。

使用 PyTorch 自动拆分模型

在第一个训练步骤中，模型并行库内部运行一个追踪步骤，该步骤旨在构建模型图并确定张量和参数形状。在此跟踪步骤之后，库构建一个树，其中包括嵌套的`nn.Module`模型中的对象，以及从跟踪中收集的其他数据，例如存储的数量`nn.Parameters`，以及每个人的执行时间`nn.Module`.

接下来，库从根目录遍历这个树并运行分区算法，分配每个`nn.Module`用于平衡计算负载（通过模块执行时间衡量）和内存使用（以存储总量衡量）的设备`nn.Parameter`大小和激活）。如果多个`nn.Modules`分享同样的`nn.Parameter`，然后将这些模块放置在同一设备上，以避免维护同一参数的多个版本。一旦做出分区决策，分配的模块和权重就会加载到他们的设备上。

使用 TensorFlow 自动拆分模型

模型并行库分析可训变量的大小和图形结构，并在内部使用图表分区算法。该算法为每个操作提供了一个设备分配，其目的是最大限度地减少设备之间所需的通信量，但须受两个限制：

- 平衡每台设备中存储的变量数
- 平衡每台设备中执行的操作数量

如果你指定`speed`为了`optimize`（在 Python SDK 的模型并行参数中），库试图平衡操作数量和`tf.Variable`每台设备中的对象。否则，它会尝试平衡`tf.Variables`.

一旦做出分区决策，库就会创建每个设备需要执行的子图的序列化表示形式，然后将它们导入到每个设备上。在分区时，库放置消耗相同的操作`tf.Variable`以及属于同一个 Keras 层的一部分的操作在同一设备上。它还尊重 TensorFlow 施加的托管限制。这意味着，例如，如果有两个 Keras 图层共享`tf.Variable`，那么作为这些层的一部分的所有操作都放在单个设备上。

框架之间自动模型拆分的比较

在 TensorFlow 中，基本的计算单元是`tf.Operation`，TensorFlow 将模型表示为的有向非循环图 (DAG)`tf.Operations`，因此建模并行库分区这个 DAG，以便每个节点都转到一个设备。至关重要的是，`tf.Operation`对象具有足够丰富的可自定义属性，从这个意义上讲，它们是普遍的，因为每个模型都保证都包含这些对象的图形。

另一方面，PyTorch 没有足够丰富和普遍的等效运营概念。PyTorch 中具有这些特征的最接近的计算单位是`nn.Module`，粒度级别要高得多，这就是为什么库在 PyTorch 中确实在这个级别进行分区的原因。

手动拆分模型

如果要手动指定模型在不同设备之间的分区方式，则可以使用手动拆分模型`smp.partition`上下文管理器。

要使用此选项，请设置`auto_partition`到`False`，然后定义`default_partition`在 SageMaker Python 开发工具包中。任何未通过`smp.partition`上下文管理器在`default_partition`. 在这种情况下，会绕过

自动拆分逻辑，每个操作都会根据您的规格进行。基于生成的图形结构，模型并行库自动创建流水线执行计划。

管道执行计划

SageMaker 的分布式模型并行库的核心功能是管道执行，它决定了模型训练期间进行计算和跨设备处理数据的顺序。Pipelining 是一种技术，通过让 GPU 在不同的数据样本上同时计算，实现模型并行性能的真正并行化，并克服由于顺序计算而导致的性能损失。

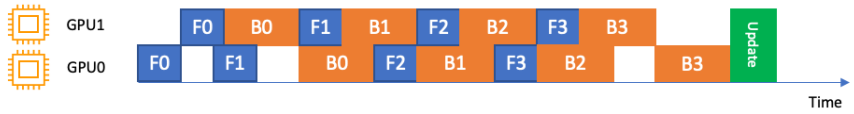
Pipelining 的基础是将小批次拆分成微批次，这些微批次逐个输入到训练管道中，并遵循库运行时间定义的执行计划。一个微批是给定训练小批次中的一个较小的子集。管道计划决定了每个时段由哪个设备执行哪个微批处理。

例如，根据管道计划和模型分区，GPU`i`可以在微批处理上执行（向前或向后）计算（向前`b`而 GPU`i+1`在微批处理上执行计算`b+1`，从而使两个 GPU 同时保持活动状态。在单次向前或向后传递期间，单个微批处理的执行流可能会多次访问同一台设备，具体取决于分区决策。例如，模型开头的操作可能与模型末尾的操作放在同一设备上，而两者之间的操作在不同的设备上，这意味着该设备被访问两次。

图书馆提供两种不同的管道时间表，简单的和交错，可以使用`pipeline`参数在 SageMaker Python 开发工具包中。在大多数情况下，交错管道可以通过更有效地利用 GPU 来实现更好的性能。

交错管道

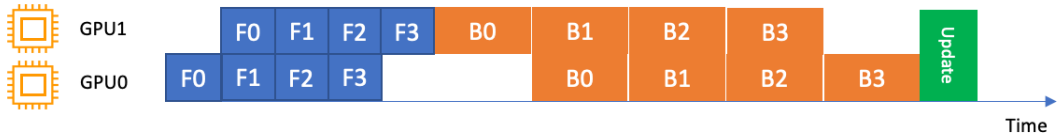
在交错管道中，尽可能优先向后执行微批量。这样可以更快地释放用于激活的内存，更有效地使用内存。它还允许扩展更多微批次的数量，从而减少 GPU 的空闲时间。处于稳定状态时，每台设备在向前和向后运行之间交替运行。这意味着一个微批次的向后传递可能会在另一个微批次的向前通过结束之前运行。



上图说明了超过 2 个 GPU 的交错管道的示例执行时间表。在图中，F0 表示微批次 0 的正向传递，B1 表示微批次 1 的向后传递。更新表示参数的优化程序更新。GPU0 总是尽可能优先向后传递（例如，在 F2 之前执行 B0），这允许更早清除用于激活的内存。

简单管道

相比之下，一条简单的管道在开始向后传递之前完成了每个微批次的正向传递。这意味着它只管道自己内部的前向传递和向后通过阶段。下图说明了超过 2 个 GPU 的工作原理的示例。

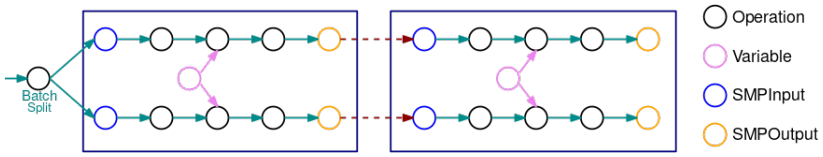


在特定框架中管道执行

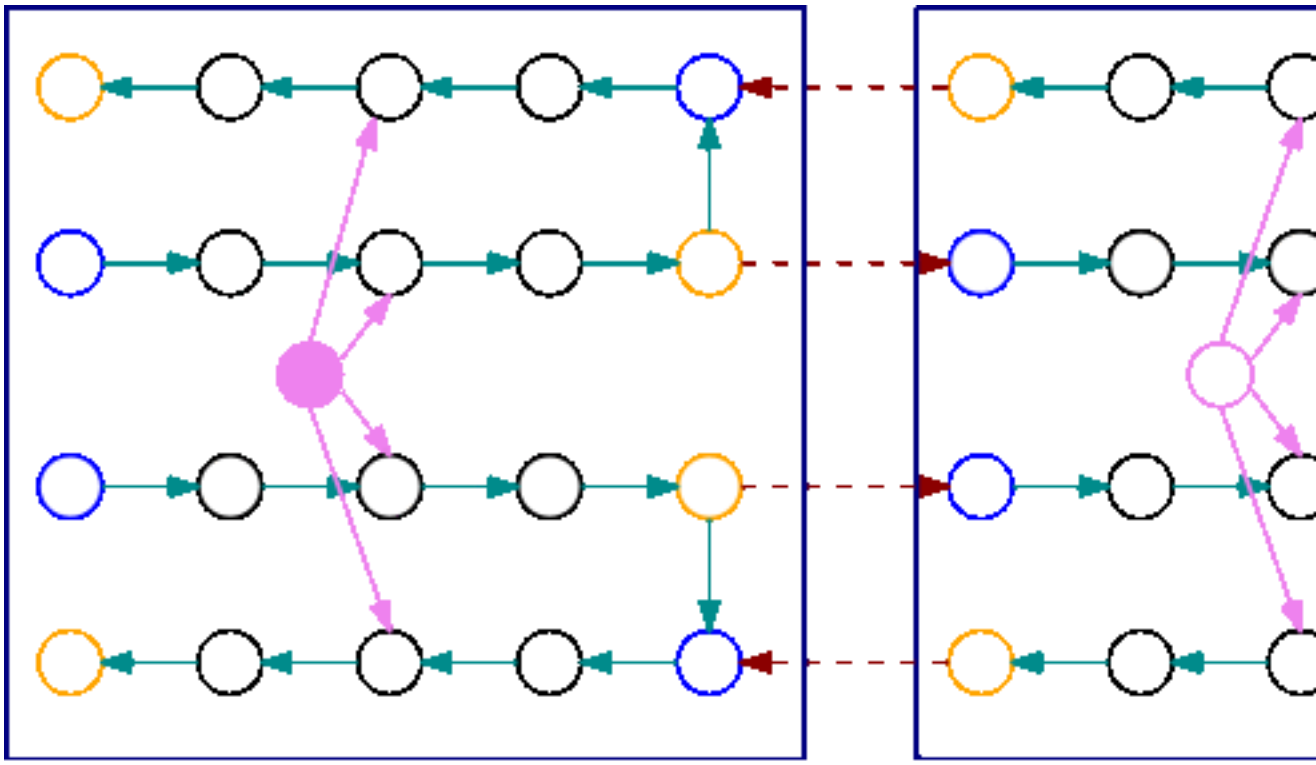
使用以下章节了解 SageMaker 的分布式模型并行库为 Tensorflow 和 PyTorch 制定的特定于框架的管道调度决策。

使用 TensorFlow 执行管道执行

下图是一个由模型并行库使用自动模型拆分进行分区的 TensorFlow 图的示例。分割图表时，每个生成的子图将被复制 B 次（变量除外），其中 B 是微批次的数量。在此图中，每个子图复制 2 次（B=2）。网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`SMPInput`将在子图的每个输入中插入操作，并且`SMPOutput`将在每个输出处插入操作。这些操作与库后端进行通信，以便彼此之间传输张量。



下图是用 B=2 拆分并添加渐变运算的 2 个子图示例。一个的渐变`SMPInput`OP 是`SMPOutput`op，反之亦然。这使渐变能够在反向传播期间向后流动。



本 GIF 演示了一个与 B=2 个微批和 2 个子图交错管道执行计划的示例。每台设备按顺序执行其中一个子图副本以提高 GPU 利用率。随着 B 的增长，空闲时段的比例将变为零。无论何时是对特定子图副本进行（向前或向后）计算的时候，管道层都会向相应的蓝色发出信号`SMPInput`要开始执行的操作。

一旦计算了来自单个小批次中所有微批次的渐变，库就会组合跨微批次的渐变，然后可以将渐变应用于参数。

使用 PyTorch ore 执行管道执行

从概念上讲，流水线在 PyTorch 中遵循了类似的想法。但是，由于 PyTorch 不涉及静态图形，因此模型并行库的 PyTorch 功能使用了更动态的流水线模式。

与在 TensorFlow 中一样，每个批次被分成多个微批次，在每台设备上一次执行一个微批次。但是，执行计划是通过在每台设备上启动的执行服务器来处理的。只要当前设备上需要放置在另一个设备上的子模块的输出，就会向远程设备的执行服务器发送执行请求以及输入张量发送到子模块。然后，服务器使用给定的输入执行此模块，并返回对当前设备的响应。

由于当前设备在远程子模块执行期间处于空闲状态，因此当前微批处理的本地执行暂停，库运行时将执行切换到当前设备可以主动处理的另一个微批处理。微批量的优先级取决于所选的管道时间表。对于交错管道计划，尽可能优先考虑处于计算后面阶段的微批量。

运行 SageMaker 具有模型并行性的分布式培训 Job

了解如何运行分布式模型 parallel 训练作业 SageMaker Python SDK 包含您自己的训练脚本和 SageMaker 的分布式模型 parallel 库。

有三种用例场景可用于运行 SageMaker 训练作业：

1. 您可以使用其中一个预构建的Amazon的深度学习容器 TensorFlow PyTorch。如果您是第一次使用模型 parallel 库，则建议使用此选项。查找有关如何运行 SageMaker 模型 parallel 训练作业，请参阅MNIST PyTorch 1.6 和 Amazon SageMaker 的分布式模型 parallel 库.
2. 您可以扩展预构建的容器，以满足预构建的算法或模型的任何其他功能要求。 SageMaker Docker 镜像不支持。有关如何扩展预构建的容器的示例，请参阅。扩展预构建的容器.
3. 你可以调整自己的 Docker 容器来使用 SageMaker 使用SageMaker 培训工具包. 有关示例，请参阅 。调整您自己的训练容器.

有关前面列表中的选项 2 和 3，请参阅扩展包含 SageMaker 分布式模型并行库的预构建 Docker 容器 (p. 2018)了解如何在扩展或自定义的 Docker 容器中安装模型 parallel 库。

在所有情况下，你都会启动你的训练作业 SageMaker `TensorFlow`要么`PyTorch`用于初始化库的估计器。要了解更多信息，请参阅以下主题。

主题

- 第 1 步：使用 SageMaker 的分布式模型并行库修改自己的训练脚本 (p. 2003)
- 第 2 步：使用 SageMaker Python 开发工具包启动训练作业 (p. 2015)

第 1 步：使用 SageMaker 的分布式模型并行库修改自己的训练脚本

使用此部分可以了解如何自定义您的训练脚本以使用亚马逊的核心功能。 SageMaker 分布式模型 parallel 库。要使用库特定的 API 函数和参数，我们建议您将本文档与SageMaker 模型 parallel 库 API中的SageMaker Python 开发工具包.

这些部分中提供的训练脚本示例经过简化，旨在突出显示使用库所需的更改。对于演示如何使用可运行的端到端笔记本示例 TensorFlow 要么 PyTorch 使用 SageMaker 分布式模型 parallel 库，请参阅亚马逊 SageMaker 分布式训练记本示例 (p. 2043).

主题

- 修改 TensorFlow 训练脚本 (p. 2003)
- 修改 PyTorch 训练脚本 (p. 2009)

修改 TensorFlow 训练脚本

在本节中，您将了解如何修改 TensorFlow 用于配置 SageMaker 分布式模型 parallel 库以进行自动分区和手动分区的训练脚本。这些示例选择还包括与 Horovod 集成的用于混合模型和数据并行性的示例。

支持的 TensorFlow 版本 SageMaker 和 SageMaker 分布式模型 parallel 库

TensorFlow 版本	SageMaker 分布式模型 parallel 库版本	**`smdistributed-modelparallel`**集成映像 URI	深度学习容器发行说明
v2.6.0	`smdistributed-modelparallel==v1.4.0`	`763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-`	Py2tf2c8p38

TensorFlow 版本	SageMaker 分布式模型 parallel 库版本	**smdistributed-modelparallel**集成映像 URI	深度学习容器发行说明
		`training:2.6.0-gpu-py38-cu112-ubuntu20.04`	
v2.5.1	`smdistributed-modelparallel==v1.4.0`	`763104351884.dkr.ecr.<region>.amazonaws.com/tensorflow-training:2.5.1-gpu-py37-cu112-ubuntu18.04`	Py1.2cu2n1Py37

要查看库的最新更新，请参阅SageMaker 分布式模型并行发行说中的SageMaker Python 开发工具包.

在继续修改之前 TensorFlow 训练脚本，建议您查看不支持的框架功能 (p. 2009).

中列出了您必须对训练脚本进行的使用库所需的修改TensorFlow (p. 2004).

要了解如何修改训练脚本以将混合模型和数据并行与 Horovod 结合使用，请参阅TensorFlow 与 Horood 实现混合模型和数据并行性 (p. 2006).

如果你想使用手动分区，还可以查看使用 TensorFlow 手动分区 (p. 2008).

> **Tip**
>
> 适用于 end-to-end 演示如何使用 TensorFlow 使用 SageMaker 分布式模型 parallel 库，请参阅TensorFlow 示例 (p. 2044).

以下主题展示了训练脚本的示例，您可以使用这些脚本配置 SageMaker 的模型 parallel 库以进行自动分区和手动分区 TensorFlow 模型。

> **Note**
>
> 默认情况下，将启用自动分区。除非另行指定，否则示例脚本使用自动分区。

主题

- TensorFlow (p. 2004)
- TensorFlow 与 Horood 实现混合模型和数据并行性 (p. 2006)
- 使用 TensorFlow 手动分区 (p. 2008)
- 不支持的框架功能 (p. 2009)

TensorFlow

需要更改以下训练脚本才能运行 TensorFlow 使用 SageMaker 的分布式模型 parallel 库的模型：

1. 使用导入并初始化库`smp.init()`.
2. 通过从继承来定义 Keras 模型`smp.DistributedModel`而不是 Keras 模型类。从中的调用方法返回模型输出`smp.DistributedModel`对象。请注意，从调用方法返回的任何张量都将在模型并行设备之间广播，从而产生通信开销，因此不应返回调用方法之外不需要的任何张量（例如中间激活）。
3. Set`drop_remainder=True`在`tf.Dataset.batch()`方法。这是为了确保批量大小始终被微批次数整除。
4. 使用为数据管道中的随机操作种子`smp.dp_rank()`例如，`shuffle(ds, seed=smp.dp_rank())`以确保包含不同模型分区的 GPU 之间的数据样本的一致性。
5. 将向前和向后逻辑放在一个步骤函数中然后用来装饰它`smp.step`.
6. 使用跨微批量对输出执行后处理`StepOutput`方法，例如`reduce_mean`. 这些区域有：`smp.step`函数必须具有取决于输出的返回值`smp.DistributedModel`.

7. 如果有评估步骤，同样将向前逻辑放在 smp.step 装饰函数中，然后使用后处理输出StepOutputAPI.

要了解 SageMaker 的分布式模型 parallel 库 API 的更多信息，请参阅API 文档.

以下 Python 脚本是进行更改后训练脚本的示例。

```
import tensorflow as tf

# smdistributed: Import TF2.x API
import smdistributed.modelparallel.tensorflow as smp

# smdistributed: Initialize
smp.init()

# Download and load MNIST dataset.
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data(
    "MNIST-data-%d" % smp.rank()
)
x_train, x_test = x_train / 255.0, x_test / 255.0

# Add a channels dimension
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]

# smdistributed: If needed, seed the shuffle with smp.dp_rank(), and drop_remainder
# in batching to make sure batch size is always divisible by number of microbatches
train_ds = (
    tf.data.Dataset.from_tensor_slices((x_train, y_train))
    .shuffle(10000, seed=smp.dp_rank())
    .batch(256, drop_remainder=True)
)

# smdistributed: Define smp.DistributedModel the same way as Keras sub-classing API
class MyModel(smp.DistributedModel):
    def __init__(self):
        super(MyModel, self).__init__()
        # define layers

    def call(self, x, training=None):
        # define forward pass and return the model output

model = MyModel()

loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
optimizer = tf.keras.optimizers.Adam()
train_accuracy = tf.keras.metrics.SparseCategoricalAccuracy(name="train_accuracy")

# smdistributed: Define smp.step. Return any tensors needed outside
@smp.step
def get_grads(images, labels):
    predictions = model(images, training=True)
    loss = loss_object(labels, predictions)

    grads = optimizer.get_gradients(loss, model.trainable_variables)
    return grads, loss, predictions


@tf.function
def train_step(images, labels):
    gradients, loss, predictions = get_grads(images, labels)

    # smdistributed: Accumulate the gradients across microbatches
    gradients = [g.accumulate() for g in gradients]
    optimizer.apply_gradients(zip(gradients, model.trainable_variables))
```

```
    # smdistributed: Merge predictions and average losses across microbatches
    train_accuracy(labels, predictions.merge())
    return loss.reduce_mean()


for epoch in range(5):
    # Reset the metrics at the start of the next epoch
    train_accuracy.reset_states()
    for images, labels in train_ds:
        loss = train_step(images, labels)
    accuracy = train_accuracy.result()
```

如果你准备完了训练脚本，请继续第 2 步：使用 SageMaker Python 开发工具包启动训练作业 (p. 2015). 如果您想运行混合模型和数据并 parallel 训练作业，请继续执行下一节。

TensorFlow 与 Horood 实现混合模型和数据并行性

您可以使用 SageMaker Horovod 的分布式模型 parallel 库用于混合模型和数据并行性。要阅读有关库如何拆分混合并行模型的更多信息，请参阅鉴于配置参数，库如何拆分模型 (p. 2018).

在此步骤中，我们重点介绍如何修改训练脚本以适应 SageMaker 分布式模型 parallel 库。

正确设置训练脚本以选择您将在其中设置的混合并行度配置第 2 步：使用 SageMaker Python 开发工具包启动训练作业 (p. 2015)，使用库的帮助函数，`smp.dp_rank()`和`smp.mp_rank()`，它分别自动检测数据 parallel 等级和模型 parallel 排名。

要查找库支持的所有 MPI 基元，请参阅MPI 基础知识中的 SageMaker Python 开发工具包文档。

脚本中需要进行的更改是：

- 添加`hvd.allreduce`
- 根据 Horovod 的要求，在第一批之后广播变量
- 在数据管道中使用播种进行洗牌和/或分片操作`smp.dp_rank()`.

> **Note**
>
> 当你使用 Horovod 时，你不能直接致电`hvd.init`在训练脚本中。相反，您必须设置`"horovod"`到`True`中的 SageMaker Python 开发工具包`modelparallel`中的参数第 2 步：使用 SageMaker Python 开发工具包启动训练作业 (p. 2015). 这允许库根据模型分区的设备分配来内部初始化 Horood。调用`hvd.init()`直接在训练脚本中可能会导致问题。

> **Note**
>
> 使用`hvd.DistributedOptimizer`直接在训练脚本中的 API 可能会导致训练性能和速度较差，因为 API 隐含地放置`AllReduce`内操作`smp.step`. 我们建议您直接调用 Horovod 将模型 parallel 库与 Horovod 一起使用`hvd.allreduce`打电话后`accumulate()`要么`reduce_mean()`关于从返回的渐变`smp.step`，如以下示例所示。

要了解 SageMaker 的分布式模型 parallel 库 API 的更多信息，请参阅API 文档.

```
import tensorflow as tf
import horovod.tensorflow as hvd

# smdistributed: Import TF2.x API
import smdistributed.modelparallel.tensorflow as smp

# smdistributed: Initialize
smp.init()

# Download and load MNIST dataset.
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data(
```

```
    "MNIST-data-%d" % smp.rank()
)
x_train, x_test = x_train / 255.0, x_test / 255.0

# Add a channels dimension
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]

# smdistributed: Seed the shuffle with smp.dp_rank(), and drop_remainder
# in batching to make sure batch size is always divisible by number of microbatches
train_ds = (
    tf.data.Dataset.from_tensor_slices((x_train, y_train))
    .shuffle(10000, seed=smp.dp_rank())
    .batch(256, drop_remainder=True)
)

# smdistributed: Define smp.DistributedModel the same way as Keras sub-classing API
class MyModel(smp.DistributedModel):
    def __init__(self):
        super(MyModel, self).__init__()
        # define layers

    def call(self, x, training=None):
        # define forward pass and return model outputs


model = MyModel()

loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
optimizer = tf.keras.optimizers.Adam()
train_accuracy = tf.keras.metrics.SparseCategoricalAccuracy(name="train_accuracy")

# smdistributed: Define smp.step. Return any tensors needed outside
@smp.step
def get_grads(images, labels):
    predictions = model(images, training=True)
    loss = loss_object(labels, predictions)

    grads = optimizer.get_gradients(loss, model.trainable_variables)
    return grads, loss, predictions


@tf.function
def train_step(images, labels, first_batch):
    gradients, loss, predictions = get_grads(images, labels)

    # smdistributed: Accumulate the gradients across microbatches
    # Horovod: Allreduce the accumulated gradients
    gradients = [hvd.allreduce(g.accumulate()) for g in gradients]
    optimizer.apply_gradients(zip(gradients, model.trainable_variables))

    # Horovod: Broadcast the variables after first batch
    if first_batch:
        hvd.broadcast_variables(model.variables, root_rank=0)
        hvd.broadcast_variables(optimizer.variables(), root_rank=0)

    # smdistributed: Merge predictions across microbatches
    train_accuracy(labels, predictions.merge())
    return loss.reduce_mean()


for epoch in range(5):
    # Reset the metrics at the start of the next epoch
    train_accuracy.reset_states()

    for batch, (images, labels) in enumerate(train_ds):
```

```
        loss = train_step(images, labels, tf.constant(batch == 0))
```

使用 TensorFlow 手动分区

使用`smp.partition`上下文管理器将操作放在特定的分区中。任何未放入任何操作`smp.partition`上下文放置在`default_partition`. 要了解 SageMaker 的分布式模型 parallel 库 API 的更多信息，请参阅API 文档.

```
import tensorflow as tf

# smdistributed: Import TF2.x API.
import smdistributed.modelparallel.tensorflow as smp

# smdistributed: Initialize
smp.init()

# Download and load MNIST dataset.
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data(
    "MNIST-data-%d" % smp.rank()
)
x_train, x_test = x_train / 255.0, x_test / 255.0

# Add a channels dimension
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]

# smdistributed: If needed, seed the shuffle with smp.dp_rank(), and drop_remainder
# in batching to make sure batch size is always divisible by number of microbatches.
train_ds = (
    tf.data.Dataset.from_tensor_slices((x_train, y_train))
    .shuffle(10000, seed=smp.dp_rank())
    .batch(256, drop_remainder=True)
)

# smdistributed: Define smp.DistributedModel the same way as Keras sub-classing API.
class MyModel(smp.DistributedModel):
    def __init__(self):
         # define layers

    def call(self, x):
        with smp.partition(0):
            x = self.layer0(x)
        with smp.partition(1):
            return self.layer1(x)


model = MyModel()

loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
optimizer = tf.keras.optimizers.Adam()
train_accuracy = tf.keras.metrics.SparseCategoricalAccuracy(name="train_accuracy")

# smdistributed: Define smp.step. Return any tensors needed outside
@smp.step
def get_grads(images, labels):
    predictions = model(images, training=True)
    loss = loss_object(labels, predictions)

    grads = optimizer.get_gradients(loss, model.trainable_variables)
    return grads, loss, predictions


@tf.function
def train_step(images, labels):
```

```
    gradients, loss, predictions = get_grads(images, labels)

    # smdistributed: Accumulate the gradients across microbatches
    gradients = [g.accumulate() for g in gradients]
    optimizer.apply_gradients(zip(gradients, model.trainable_variables))

    # smdistributed: Merge predictions and average losses across microbatches
    train_accuracy(labels, predictions.merge())
    return loss.reduce_mean()


for epoch in range(5):
    # Reset the metrics at the start of the next epoch
    train_accuracy.reset_states()
    for images, labels in train_ds:
        loss = train_step(images, labels)
    accuracy = train_accuracy.result()
```

不支持的框架功能

以下 TensorFlow 库不支持功能：

- `tf.GradientTape()`当前不支持。您可以使用`Optimizer.get_gradients()`要么`Optimizer.compute_gradients()`而是计算渐变。
- 这些区域有：`tf.train.Checkpoint.restore()`当前不支持 API。对于检查点，请使用`smp.CheckpointManager`相反，它提供了相同的 API 和功能。请注意，检查点还原`smp.CheckpointManager`应该在第一步之后进行。

修改 PyTorch 训练脚本

在本节中，您将了解如何修改 PyTorch 训练脚本来配置 SageMaker 用于自动分区和手动分区的分布式模型 parallel 库。

PyTorch 版本支持 SageMaker 和 SageMaker 分布式模型 parallel 库

PyTorch 版本	SageMaker 分布式模型 parallel 库版本	**`smdistributed-modelparallel`**集成映像 URI
v1.11.0*	`smdistributed-modelparallel==v1.9.0`	`763104351884.dkr.ecr.<region>.amazon` `pytorch-training:1.11.0-gpu-py38-cu113-ubuntu20.04-sagemaker`
v1.10.2*	`smdistributed-modelparallel==v1.7.0`	`763104351884.dkr.ecr.<region>.amazon` `pytorch-training:1.10.2-gpu-py38-cu113-ubuntu20.04-sagemaker`
v1.10.0	`smdistributed-modelparallel==v1.5.0`	`763104351884.dkr.ecr.<region>.amazon` `pytorch-training:1.10.0-gpu-py38-cu113-ubuntu20.04-sagemaker`
v1.9.1	`smdistributed-modelparallel==v1.4.0`	`763104351884.dkr.ecr.<region>.amazon` `pytorch-training:1.9.1-gpu-py38-cu111-ubuntu20.04`
v1.8.1*	`smdistributed-modelparallel==v1.6.0`	`763104351884.dkr.ecr.<region>.amazon` `pytorch-training:1.8.1-`

PyTorch 版本	SageMaker 分布式模型 parallel 库版本	**smdistributed-modelparallel**集成映像 URI
		gpu-py36-cu111-ubuntu18.04

* 该 SageMaker 分布式模型 parallel 库 v1.6.0 及更高版本为 PyTorch 提供扩展功能。有关更多信息，请参阅 PyTorch 的 SageMaker 模型并行库的扩展功能 (p. 2021)。

要查看库的最新更新，请参阅SageMaker 分布式模型并行发行说中的SageMaker Python 开发工具包.

以下主题展示了训练脚本的示例，您可以使用这些脚本配置 SageMaker 的模型 parallel 库以进行自动分区和手动分区 PyTorch 模型。建议您查看重要注意事项 (p. 2013)和不支持的框架功能 (p. 2015)在创建训练脚本之前。

中列出了您必须对训练脚本进行的使用库所需的修改PyTorch (p. 2010).

如果你想使用手动分区，还可以查看使用 PyTorch 手动分区 (p. 2012).

> **Tip**
>
> 适用于 end-to-end 演示如何使用 PyTorch 使用 SageMaker 分布式模型 parallel 库，请参阅PyTorch 示例 (p. 2044).

请注意，默认情况下，将启用自动分区。除非另行指定，否则以下脚本使用自动分区。

主题

- PyTorch (p. 2010)
- 使用 PyTorch 手动分区 (p. 2012)
- 重要注意事项 (p. 2013)
- 不支持的框架功能 (p. 2015)

PyTorch

需要更改以下训练脚本才能运行 PyTorch 使用 SageMaker 的分布式模型 parallel 库的模型：

1. 使用导入并初始化库`smp.init()`.
2. 用模型包裹`smp.DistributedModel`. 请注意，从`forward`基础方法`nn.Module`对象将在模型并行设备之间广播，从而产生通信开销，因此不应返回调用方法之外不需要的任何张量（例如中间激活）。
3. 用包装优化器`smp.DistributedOptimizer`.
4. 使用返回的`DistributedModel`对象而不是用户模型。
5. 将向前和向后逻辑放在一个步骤函数中然后用来装饰它`smp.step`.
6. 将每个进程限制在自己的设备上`torch.cuda.set_device(smp.local_rank())`.
7. 使用将输入张量移动到 GPU`.to()`之前的 API`smp.step`电话（见下面的例子）。
8. Replace`torch.Tensor.backward`和`torch.autograd.backward`和`DistributedModel.backward`.
9. 使用跨微批量对输出执行后处理`StepOutput`方法，例如`reduce_mean`.
10. 如果有评估步骤，同样地将向前逻辑放在`smp.step`-装饰功能并使用后处理输出`StepOutput`API.
11. Set`drop_last=True`在`DataLoader`. 或者，如果批量大小不能被微批次数整除，则手动跳过训练循环中的批次。

要了解 SageMaker 的分布式模型 parallel 库 API 的更多信息，请参阅API 文档.

```
import torch
```

```
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchnet.dataset import SplitDataset
from torchvision import datasets

import smdistributed.modelparallel.torch as smp

class GroupedNet(nn.Module):
    def __init__(self):
        super(GroupedNet, self).__init__()
        # define layers

    def forward(self, x):
        # define forward pass and return model outputs


# smdistributed: Define smp.step. Return any tensors needed outside.
@smp.step
def train_step(model, data, target):
    output = model(data)
    loss = F.nll_loss(output, target, reduction="mean")
    model.backward(loss)
    return output, loss


def train(model, device, train_loader, optimizer):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        # smdistributed: Move input tensors to the GPU ID used by the current process,
        # based on the set_device call.
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        # Return value, loss_mb is a StepOutput object
        _, loss_mb = train_step(model, data, target)

        # smdistributed: Average the loss across microbatches.
        loss = loss_mb.reduce_mean()

        optimizer.step()

# smdistributed: initialize the backend
smp.init()

# smdistributed: Set the device to the GPU ID used by the current process.
# Input tensors should be transferred to this device.
torch.cuda.set_device(smp.local_rank())
device = torch.device("cuda")

# smdistributed: Download only on a single process per instance.
# When this is not present, the file is corrupted by multiple processes trying
# to download and extract at the same time
dataset = datasets.MNIST("../data", train=True, download=False)

# smdistributed: Shard the dataset based on data-parallel ranks
if smp.dp_size() > 1:
    partitions_dict = {f"{i}": 1 / smp.dp_size() for i in range(smp.dp_size())}
    dataset = SplitDataset(dataset, partitions=partitions_dict)
    dataset.select(f"{smp.dp_rank()}")

# smdistributed: Set drop_last=True to ensure that batch size is always divisible
# by the number of microbatches
train_loader = torch.utils.data.DataLoader(dataset, batch_size=64, drop_last=True)

model = GroupedNet()
optimizer = optim.Adadelta(model.parameters(), lr=4.0)
```

```
# smdistributed: Use the DistributedModel container to provide the model
# to be partitioned across different ranks. For the rest of the script,
# the returned DistributedModel object should be used in place of
# the model provided for DistributedModel class instantiation.
model = smp.DistributedModel(model)
optimizer = smp.DistributedOptimizer(optimizer)

train(model, device, train_loader, optimizer)
```

使用 PyTorch 手动分区

使用`smp.partition`上下文管理器将模块放置在特定设备中。任何未放置在任何模块中`smp.partition`上下文放置在`default_partition`. 这些区域有：`default_partition`如果需要提供`auto_partition`设置为`False`. 在特定内部创建的模块`smp.partition`上下文放置在相应的分区上。

要了解 SageMaker 的分布式模型 parallel 库 API 的更多信息，请参阅API 文档.

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchnet.dataset import SplitDataset
from torchvision import datasets

import smdistributed.modelparallel.torch as smp

class GroupedNet(nn.Module):
    def __init__(self):
        super(GroupedNet, self).__init__()
        with smp.partition(0):
            # define child modules on device 0
        with smp.partition(1):
            # define child modules on device 1

    def forward(self, x):
        # define forward pass and return model outputs


# smdistributed: Define smp.step. Return any tensors needed outside.
@smp.step
def train_step(model, data, target):
    output = model(data)
    loss = F.nll_loss(output, target, reduction="mean")
    model.backward(loss)
    return output, loss


def train(model, device, train_loader, optimizer):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        # smdistributed: Move input tensors to the GPU ID used by the current process,
        # based on the set_device call.
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        # Return value, loss_mb is a StepOutput object
        _, loss_mb = train_step(model, data, target)

        # smdistributed: Average the loss across microbatches.
        loss = loss_mb.reduce_mean()

        optimizer.step()

# smdistributed: initialize the backend
```

```
smp.init()

# smdistributed: Set the device to the GPU ID used by the current process.
# Input tensors should be transferred to this device.
torch.cuda.set_device(smp.local_rank())
device = torch.device("cuda")

# smdistributed: Download only on a single process per instance.
# When this is not present, the file is corrupted by multiple processes trying
# to download and extract at the same time
dataset = datasets.MNIST("../data", train=True, download=False)

# smdistributed: Shard the dataset based on data-parallel ranks
if smp.dp_size() > 1:
    partitions_dict = {f"{i}": 1 / smp.dp_size() for i in range(smp.dp_size())}
    dataset = SplitDataset(dataset, partitions=partitions_dict)
    dataset.select(f"{smp.dp_rank()}")

# smdistributed: Set drop_last=True to ensure that batch size is always divisible
# by the number of microbatches
train_loader = torch.utils.data.DataLoader(dataset, batch_size=64, drop_last=True)

model = GroupedNet()
optimizer = optim.Adadelta(model.parameters(), lr=4.0)

# smdistributed: Use the DistributedModel container to provide the model
# to be partitioned across different ranks. For the rest of the script,
# the returned DistributedModel object should be used in place of
# the model provided for DistributedModel class instantiation.
model = smp.DistributedModel(model)
optimizer = smp.DistributedOptimizer(optimizer)

train(model, device, train_loader, optimizer)
```

重要注意事项

当你配置 PyTorch 使用 SageMaker 的分布式模型 parallel 库使用训练脚本，您应了解以下内容：

- 如果您使用的优化技术依赖于全局渐变规范，例如整个模型的渐变范数，例如 LAMP 优化器或全局渐变剪切的某些变体，则需要在模型分区中收集所有规范以确保正确性。你可以使用图书馆的通信基本数据类型来做到这一点。
- 全部`torch.Tensor`对前瞻方法的参数`nn.Modules`在模型中必须用于计算模块输出。换句话说，图书馆不支持存在`torch.Tensor`参数指向模块输出不依赖的模块。
- 对`smp.DistributedModel.backward()`调用必须取决于所有模型输出。换句话说，无法从`smp.DistributedModel.forward`在计算被输入的张量时没有使用的调用`smp.DistributedModel.backward`调用。
- 如果有`torch.cuda.synchronize()`在你的代码中调用，你可能需要打电话`torch.cuda.set_device(smp.local_rank())`紧接在同步调用之前。否则，可能会在设备 0 中创建不必要的 CUDA 上下文，这将不必要地消耗内存。
- 自图书馆放置以来`nn.Modules`在不同的设备上，模型中的模块不能依赖于内部修改的任何全局状态`smp.step`. 在整个训练过程中保持固定或在外部修改的任何状态`smp.step`允许以所有进程都可见的方式。
- 您不需要将模型移动到 GPU（例如，使用`model.to(device)`）在使用图书馆时。如果在模型分区之前尝试将模型移动到 GPU（在第一个模型之前）`smp.step`调用），移动呼叫将被忽略。库会自动将分配给排名的模型部分移动到其 GPU。一旦开始使用库进行训练，不要将模型移动到 CPU 并使用它，因为对于没有分配给过程保存的分区的模块，它将没有正确的参数。如果您想在使用模型并行库训练后重新训练模型或在没有库的情况下使用它进 parallel 推理，建议的方法是使用我们的检查点 API 保存完整模型并将其加载回常规模。 PyTorch 模块。
- 如果你有一个模块列表，例如一个输出的输出到另一个模块，请将该列表替换为`nn.Sequential`可以显著提高性能。

- 权重更新（optimizer.step()）需要在外面发生smp.step因为那是整个向后传递完成并且渐变准备就绪的时候。当使用具有模型和数据并行性的混合模型时，此时，梯度的 Allreduce 也可以保证完成。
- 将库与数据并行结合使用时，请确保所有数据并 parallel 排名上的批次数相同，以便 Allreduce 不会挂起等待未参与该步骤的排名。
- 如果您使用 ml.p4d 实例类型（例如 ml.p4d.24xlarge）启动训练作业，则必须设置数据加载器变量num_workers=0. 例如，您可以定义您的DataLoader如下所示：

```
dataloader = torch.utils.data.DataLoader(
            data,
            batch_size=batch_size,
            num_workers=0,
            pin_memory=True,
            drop_last=True,
            shuffle=shuffle,
        )
```

- 对的输入smp.step必须是由生成的模型输入DataLoader. 这是因为smp.step沿批处理维度内部拆分输入张量并将它们管道。这意味着通过DataLoader本身转换为smp.step在里面生成模型输入的函数不起作用。

  例如，如果你定义了DataLoader如下所示：

```
train_loader = torch.utils.data.DataLoader(dataset, batch_size=64, drop_last=True)
```

  你应该访问生成的模型输入train_loader然后把那些传递给smp.step装饰功能。不要通过train_loader直接转换为smp.stepfunction.

```
def train(model, device, train_loader, optimizer):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        ...
        _, loss_mb = train_step(model, data, target)
        ...

@smp.step
def train_step(model, data, target):
    ...
    return output, loss
```

- 输入张量smp.step必须使用移动到当前设备.to()API，必须在torch.cuda.set_device(local_rank())调用。

  例如，您可以定义train函数如下方式。此函数添加data和target使用调用当前设备.to()在使用这些输入张量调用之前的 APItrain_step.

```
def train(model, device, train_loader, optimizer):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        # smdistributed: Move input tensors to the GPU ID used by the current process,
        # based on the set_device call.
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        # Return value, loss_mb is a StepOutput object
        _, loss_mb = train_step(model, data, target)

        # smdistributed: Average the loss across microbatches.
        loss = loss_mb.reduce_mean()

        optimizer.step()
```

这个的输入张量`smp.set`装饰功能已移动到中的当前设备`train`上面的函数。模型确实如此不需要移动到当前设备。库会自动将分配给排名的模型部分移动到其 GPU。

```
@smp.step
def train_step(model, data, target):
    output = model(data)
    loss = F.nll_loss(output, target, reduction="mean")
    model.backward(loss)
    return output, loss
```

不支持的框架功能

以下 PyTorch SageMaker 的分布式模型 parallel 库不支持功能：

- 如果你将数据并行与本机一起使用PyTorch DDP，`torch.nn.parallel.DistributedDataParallel`库不支持包装模块。该图书馆内部管理与集成 PyTorch DDP，包括参数广播和梯度 allReduce。使用库时，模块缓冲区仅在训练开始时广播一次。如果模型的模块缓冲区在每个步骤都需要在数据 parallel 组之间进行同步，则可以通过`torch.distributed`API，使用可以通过以下方式获得的进程组`smp.get_dp_process_group()`.
- 对于混合精度训练，`apex.amp`不支持模块。使用自动混合精度库的推荐方法是使用`torch.cuda.amp`，除了使用`smp.amp.GradScaler`而不是在火炬中实施。
- `torch.jit.ScriptModules`要么`ScriptFunctions`不支持`smp.DistributedModel`.
- `apex`：`FusedLayerNorm`、`FusedAdam`、`FusedLAMB`, 和`FusedNovoGrad`从`apex`不支持。您可以通过以下方式使用库实现。`smp.optimizers`和`smp.nn`而不是 API。

第 2 步：使用 SageMaker Python 开发工具包启动训练作业

这些区域有： SageMaker Python SDK 支持使用 ML 框架对模型进行托管训练，例如 TensorFlow PyTorch。要使用这些框架之一启动培训作业，您需要定义 SageMaker TensorFlow 评估程序，a SageMaker PyTorch 估算程序，或者 SageMaker 通用的估算程序以使用修改后的训练脚本和模型并行度配置。

主题

- 使用 SageMaker TensorFlow 和 PyTorch 估算程序 (p. 2015)
- 鉴于配置参数，库如何拆分模型 (p. 2018)
- 扩展包含 SageMaker 分布式模型并行库的预构建 Docker 容器 (p. 2018)
- 使用创建自己的 Docker 容器 SageMaker 分布式模型并行库 (p. 2019)

使用 SageMaker TensorFlow 和 PyTorch 估算程序

这些区域有： TensorFlow 和 PyTorch 估计器类包含`distribution`参数，您可以使用该参数来指定使用分布式训练框架的配置参数。这些区域有： SageMaker 模型 parallel 库内部使用 MPI 进行混合数据和模型并行性，因此您必须将 MPI 选项与库一起使用。

下面的模板 TensorFlow 要么 PyTorch 估算程序显示如何配置`distribution`用于使用 SageMaker 使用 MPI 模型 parallel 库。

Using the SageMaker TensorFlow estimator

```
import sagemaker
from sagemaker.tensorflow import TensorFlow
```

```
smp_options = {
    "enabled":True,              # Required
    "parameters": {
        "partitions": 2,         # Required
        "microbatches": 4,
        "placement_strategy": "spread",
        "pipeline": "interleaved",
        "optimize": "speed",
        "horovod": True,         # Use this for hybrid model and data parallelism
    }
}

mpi_options = {
    "enabled" : True,            # Required
    "processes_per_host" : 8,    # Required
    # "custom_mpi_options" : "--mca btl_vader_single_copy_mechanism none"
}

smd_mp_estimator = TensorFlow(
    entry_point="your_training_script.py", # Specify your train script
    source_dir="location_to_your_script",
    role=sagemaker.get_execution_role(),
    instance_count=1,
    instance_type='ml.p3.16xlarge',
    framework_version='2.6.3',
    py_version='py38',
    distribution={
        "smdistributed": {"modelparallel": smp_options},
        "mpi": mpi_options
    },
    base_job_name="SMD-MP-demo",
)

smd_mp_estimator.fit('s3://my_bucket/my_training_data/')
```

Using the SageMaker PyTorch estimator

```
import sagemaker
from sagemaker.pytorch import PyTorch

smp_options = {
    "enabled":True,
    "parameters": {                        # Required
        "pipeline_parallel_degree": 2,     # Required
        "microbatches": 4,
        "placement_strategy": "spread",
        "pipeline": "interleaved",
        "optimize": "speed",
        "ddp": True,
    }
}

mpi_options = {
    "enabled" : True,                      # Required
    "processes_per_host" : 8,              # Required
    # "custom_mpi_options" : "--mca btl_vader_single_copy_mechanism none"
}

smd_mp_estimator = PyTorch(
    entry_point="your_training_script.py", # Specify your train script
    source_dir="location_to_your_script",
    role=sagemaker.get_execution_role(),
    instance_count=1,
    instance_type='ml.p3.16xlarge',
    framework_version='1.10.2',
```

```
    py_version='py38',
    distribution={
        "smdistributed": {"modelparallel": smp_options},
        "mpi": mpi_options
    },
    base_job_name="SMD-MP-demo",
)

smd_mp_estimator.fit('s3://my_bucket/my_training_data/')
```

要启用库，你需要将配置字典传递给"smdistributed"和"mpi"密钥通过distribution的参数 SageMaker 估计函数。

的配置参数 SageMaker 模型并行

- 对于"smdistributed"关键，用"modelparallel"key 和以下内部字典。

  Note

  使用"modelparallel"和"dataparallel"在一个培训工作中不受支持。

  - "enabled" – 必需。要启用模型并行性，请设置"enabled": True.
  - "parameters" – 必需。指定一组参数 SageMaker 模型并行。
    - 有关常见参数的完整列表，请参阅。的参数smdistributed中的SageMaker Python 开发工具包文.

      对于 TensorFlow，请参阅TensorFlow 特定的参数.

      对于 PyTorch，请参阅PyTorch 特定的参数.
    - "pipeline_parallel_degree"（或者"partitions"在smdistributed-modelparallel<v1.6.0) — 必需。其中的参数smdistributed，则需要此参数来指定要拆分到多少个模型分区。

      Important

      参数名称发生了突破性的变化。这些区域有："pipeline_parallel_degree"参数替换"partitions"以来smdistributed-modelparallelv1.6.0。有关更多信息，请参阅 。常见参数为了 SageMaker 模型并行度配置和SageMaker 分布式模型并行发行说中的SageMaker Python 开发工具包文.
- 对于"mpi"key，传递一个包含以下内容的字典：
  - "enabled" – 必需。SetTrue使用 MPI 启动分布式培训作业。
  - "processes_per_host" – 必需。指定 MPI 应在每台主机上启动的进程数。In SageMaker 主机是单个Amazon EC2 ML 实例. 这些区域有： SageMaker Python 开发工具包维护 one-to-one 跨模型和数据并行性在流程和 GPU 之间进行映射。这意味着 SageMaker 在单个独立的 GPU 上安排每个进程，并且没有 GPU 包含多个进程。如果您使用 PyTorch，则必须通过以下方式将每个进程限制在自己的设备上torch.cuda.set_device(smp.local_rank()). 要了解更多信息，请参阅 PyTorch (p. 2010)。

    Important

    process_per_host 必须不大于每个实例的 GPU 数量，通常等于每个实例的 GPU 数量。
  - "custom_mpi_options"（可选）— 使用此键传递您可能需要的任何自定义 MPI 选项。如果没有将任何 MPI 自定义选项传递给密钥，则默认情况下，MPI 选项将设置为以下标志。

```
--mca btl_vader_single_copy_mechanism none
```

    Note

    您无需为密钥显式指定此默认标志。如果明确指定它，则分布式模型 parallel 训练作业可能会失败并显示以下错误：

```
The following MCA parameter has been listed multiple times on the command line:
MCA param: btl_vader_single_copy_mechanism MCA parameters can only be listed
 once
on a command line to ensure there is no ambiguity as to its value.
Please correct the situation and try again.
```

Tip

如果您使用启用 EPA 的实例类型启动培训作业，例
如`ml.p4d.24xlarge`和`ml.p3dn.24xlarge`，使用以下标志以获得最佳性能：

```
-x FI_EFA_USE_DEVICE_RDMA=1 -x FI_PROVIDER=efa -x RDMAV_FORK_SAFE=1
```

使用估算器启动训练作业和 SageMaker 模型 parallel 配置的训练脚本，运行`estimator.fit()`function.

鉴于配置参数，库如何拆分模型

在为模型分区数量指定值时（`pipeline_parallel_degree`为了 PyTorch 要么`partition`对于 TensorFlow），GPU 的总数（`processes_per_host`）必须由模型分区的数量整除。要正确设置它，你必须指定正确的值`pipeline_parallel_degree`和`processes_per_host`参数。简单的数学如下：

```
(pipeline_parallel_degree) x (data_parallelism_degree) = processes_per_host
```

该库负责计算模型副本的数量（`data_parallelism_degree`）考虑到你提供的两个输入参数。例如，如果您设置`pipeline_parallel_degree=4`和`processes_per_host=8`将机器学习实例与八个 GPU 工作器一起使用，例如`ml.p3.16xlarge`，库自动设置双向数据并行机制（`data_parallelism_degree=2`）。

下图说明了双向数据并行度和四向模型并行度如何分布在八个 GPU 上：模型分区在四个 GPU 上，每个分区分配给两个 GPU。

使用以下资源了解有关使用中的模型并行性功能的更多信息： SageMaker Python 开发工具包：

- 使用 TensorFlow 使用 SageMaker Python 开发工具包
- 使用 PyTorch 使用 SageMaker Python 开发工具包

- 我们建议您使用 SageMaker 笔记本实例（如果你是新用户）。有关如何使用启动培训作业的示例，请参阅 SageMaker 笔记本实例，请参阅亚马逊 SageMaker 分布式训练记本示例 (p. 2043).
- 您还可以使用以下方式从计算机上提交分布式培训作业Amazon CLI. 设置Amazon CLI在计算机上，请参阅设置Amazon证书和发展区域.

扩展包含 SageMaker 分布式模型并行库的预构建 Docker 容器

要扩展预构建的容器并使用 SageMaker 的分布式模型 parallel 库，必须使用其中一个可用AmazonDeep Learning Containers (DLC) 映像 PyTorch 或 TensorFlow。这些区域有： SageMaker 分布式模型 parallel 库包含在 TensorFlow （2.3.0 及更高版本）和 PyTorch （1.6.0 及更高版本）使用 CUDA 的 DLC 映像（`cuxyz`）。有关 DLC 映像的完整列表，请参阅可用的 Deep Learning Containers 镜像中的AmazonDeep Learning 容器 GitHub 知识库.

Tip

我们建议您使用包含 TensorFlow 的最新版本的镜像，或 PyTorch 以获得最多 up-to-date 的版本 SageMaker 分布式模型 parallel 库。

例如，你的 Dockerfile 应该包含FROM类似于以下内容的语句：

```
# Use the SageMaker DLC image URI for TensorFlow or PyTorch
FROM aws-dlc-account-id.dkr.ecr.aws-region.amazonaws.com/framework-training:{framework-version-tag}

# Add your dependencies here
RUN ...

ENV PATH="/opt/ml/code:${PATH}"

# this environment variable is used by the SageMaker container to determine our user code
 directory.
ENV SAGEMAKER_SUBMIT_DIRECTORY /opt/ml/code
```

此外，当您定义 PyTorch 要么 TensorFlow 估算程序，则必须指定entry_point为您的训练脚本。这应与标识的路径相同。ENV SAGEMAKER_SUBMIT_DIRECTORY在 Dockerfile 中。

> **Tip**
>
> 您必须将此 Docker 容器推送到 Amazon Elastic Container Registry (Amazon ECR) 并使用镜像 URI (image_uri) 定义 SageMaker 用于训练的估计器。有关更多信息，请参阅 。扩展预构建的容器.

完成托管 Docker 容器并检索容器的映像 URI 后，创建 SageMaker PyTorch估计器对象如下。此示例假定您已定义。smp_options和mpi_options.

```
smd_mp_estimator = Estimator(
    entry_point="your_training_script.py",
    role=sagemaker.get_execution_role(),
    instance_type='ml.p3.16xlarge',
    sagemaker_session=sagemaker_session,
    image_uri='your_aws_account_id.dkr.ecr.region.amazonaws.com/name:tag'
    instance_count=1,
    distribution={
        "smdistributed": smp_options,
        "mpi": mpi_options
    },
    base_job_name="SMD-MP-demo",
)

smd_mp_estimator.fit('s3://my_bucket/my_training_data/')
```

使用创建自己的 Docker 容器 SageMaker 分布式模型并行库

要构建自己的 Docker 容器进行训练并使用 SageMaker 模型 parallel 库，则必须在 Dockerfile 中包含正确的依赖关系和 SageMaker 分布式 parallel 库的二进制文件。本节提供了正确准备 SageMaker 训练环境和您自己的 Docker 容器中的模型 parallel 库。

> **Note**
>
> 这个自定义 Docker 选项带 SageMaker 模型 parallel 库作为二进制文件仅适用于 PyTorch。

使用 SageMaker 培训工具包和模型 parallel 库

1. 从其中一个开始NVIDIA CUDA 基本映像.

```
FROM <cuda-cudnn-base-image>
```

Tip

官方Amazon深度学习容器 (DLC) 映像是从NVIDIA CUDA 基本映像. 我们建议您查看的官方 DockerfileAmazonPyTorch 的深度学习容器查找需要安装哪些版本的库以及如何配置它们。官方 Dockerfile 已完整、基准测试和管理 SageMaker 和深度学习容器服务团队。在提供的链接中，选择你使用的 PyTorch 版本，选择 CUDA (`cuxyz`) 文件夹，然后选择以结尾的 Dockerfile`.gpu`要么`.sagemaker.gpu`.

2. 要设置分布式培训环境，您需要安装用于通信和网络设备的软件，例如EFA Elastic Fabric Adapter (EFA)、NVIDIA 集体沟通图书馆 (NCCL), 和Open MPI. 取决于 PyTorch 和你选择的 CUDA 版本，你必须安装兼容版本的库。

   Important

   由于 SageMaker 模型 parallel 库需要 SageMaker parallel 后续步骤中，我们强烈建议您按照以下内容的说明操作：使用创建自己的 Docker 容器 SageMaker 分布式数据并行库 (p. 1985)正确设置 SageMaker 分布式培训的培训环境。

   有关使用 NCCL 和 Open MPI 设置 EFA 的更多信息，请参阅。EFA 和 MPI 入门和EFA 和 NCCL 入门.

3. 添加以下参数以指定 SageMaker 为 PyTorch 分发了培训包。这些区域有： SageMaker 模型 parallel 库需要 SageMaker 数据 parallel 库以使用跨节点远程直接内存访问 (RDMA)。

```
ARG SMD_MODEL_PARALLEL_URL=https://sagemaker-distributed-model-parallel.s3.us-
west-2.amazonaws.com/pytorch-1.10.0/build-artifacts/2022-02-21-19-26/
smdistributed_modelparallel-1.7.0-cp38-cp38-linux_x86_64.whl
ARG SMDATAPARALLEL_BINARY=https://smdataparallel.s3.amazonaws.com/binary/pytorch/1.10.2/
cu113/2022-02-18/smdistributed_dataparallel-1.4.0-cp38-cp38-linux_x86_64.whl
```

4. 安装依赖项 SageMaker 模型 parallel 库需要。

   a. 安装METIS库.

```
ARG METIS=metis-5.1.0

RUN rm /etc/apt/sources.list.d/* \
  && wget -nv http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/${METIS}.tar.gz \
  && gunzip -f ${METIS}.tar.gz \
  && tar -xvf ${METIS}.tar \
  && cd ${METIS} \
  && apt-get update \
  && make config shared=1 \
  && make install \
  && cd .. \
  && rm -rf ${METIS}.tar* \
  && rm -rf ${METIS} \
  && rm -rf /var/lib/apt/lists/* \
  && apt-get clean
```

   b. 安装RAPID 内存管理器库. 这需要CMake3.14 或更高版本。

```
ARG RMM_VERSION=0.15.0

RUN  wget -nv https://github.com/rapidsai/rmm/archive/v${RMM_VERSION}.tar.gz \
  && tar -xvf v${RMM_VERSION}.tar.gz \
  && cd rmm-${RMM_VERSION} \
  && INSTALL_PREFIX=/usr/local ./build.sh librmm \
  && cd .. \
  && rm -rf v${RMM_VERSION}.tar* \
  && rm -rf rmm-${RMM_VERSION}
```

5. 安装 SageMaker 模型 parallel 库。

```
RUN pip install --no-cache-dir -U ${SMD_MODEL_PARALLEL_URL}
```

6. 安装 SageMaker 数据 parallel 库。

```
RUN SMDATAPARALLEL_PT=1 pip install --no-cache-dir ${SMDATAPARALLEL_BINARY}
```

7. 安装sagemaker 培训工具包. 该工具包包含创建与兼容的容器所必须使用的常见功能。 SageMaker 培训平台和 SageMaker Python 开发工具包。

```
RUN pip install sagemaker-training
```

8. 创建完 Dockerfile 后，请参阅调整您自己的训练容器了解如何构建 Docker 容器并将其托管在 Amazon ECR 中。

   Tip

   有关在 SageMaker 中创建自定义 DockerFile 的更多常规信息，请参阅。使用您自己的训练算法.

PyTorch 的 SageMaker 模型并行库的扩展功能

除了它核心功能， SageMaker 分布式模型 parallel 库为使用 PyTorch 训练深度学习模型提供了节省内存的功能：张量并行、优化器状态分片、激活检查点和激活卸载。

Note

可以通过适用于 PyTorch 的 Deep Learning Containers 实现扩展内存节省功能，该容器实现 SageMaker 分布式模型 parallel 库 v1.6.0 或更高版本。

对于以下每项功能，您都可以保留相同的两步工作流程运行 SageMaker 具有模型并行性的分布式培训 Job (p. 2003)部分，然后将一些额外的参数和代码行添加到 SageMaker `PyTorch`估算器和训练脚本。

要查找如何使用扩展功能的示例，请参阅使用训练 GPT-2 PyTorch 使用 1.8.1 和张量并行度 SageMaker 模型并行度库.

主题

- Tensor 并行度 (p. 2021)
- 分片优化器状态 (p. 2031)
- 激活检查点 (p. 2032)
- 激活分载 (p. 2033)
- 排名机制 (p. 2034)

Tensor 并行度

Tensor 并行度是一种模型并行性，其中特定的模型权重、渐变和优化程序状态在不同设备之间进行分割。与管道并行性不同，管道并行度保持了个别权重不变，但是将设置在权重中，张量并行度分割单个权重。这通常涉及对模型的特定操作、模块或层进行分布式计算。

在单个参数消耗大部分 GPU 内存的情况下（例如，词汇量较大的大型嵌入表或具有大量类的大型 softmax 层），则需要 Tensor 并行性。在这种情况下，将这种大张量或操作当作原子单元处理效率低下，并且会阻碍内存负载的平衡。

对于纯流水线根本不够的极大型模型，Tensor 并行性也很有用。例如，对于需要在数十个实例上进行分区的 GPT-3 级模型，因此纯微批量流水线效率低下，因为管道深度变得太高，开销变得令人望而却步。

主题

- Tensor 并行度的工作原理 (p. 2022)
- 运行 SageMaker 具有 Tensor 并行性的分布式模型并行训练作业 (p. 2023)
- 使用 Tensor 并行度进行检查点的说明 (p. 2028)
- Support 拥抱脸变压器模型 (p. 2029)

Tensor 并行度的工作原理

Tensor 并行性发生在`nn.Modules`; 它将模型中的特定模块划分为张量 parallel 等级。这是对现有的分区的补充一组模块在管道并行机制中使用。

当一个模块通过张量并行分区时，它的向前和向后传播都是分布的。该库处理必要的跨设备通信，以实现这些模块的分布式执行；这些模块分布在多个数据 parallel 等级中。与传统的工作负载分布相反，每个数据 parallel 排名都是不当使用库的张量并行度时，拥有完整的模型副本。相反，除了未分发的整个模块之外，每个数据 parallel 排名可能只有分布式模块的一个分区。

示例：考虑跨数据并 parallel 排名的张量并行度，其中数据并行度为 4，张量并行度为 2。假设在对模块集进 parallel 分区之后，有一个包含以下模块树的数据并行组。

```
A
### B
|   ### E
|   ### F
### C
### D
    ### G
    ### H
```

假设模块 B、G 和 H 支持张量并行性，此模型的张量 parallel 分区的一个可能结果可能是：

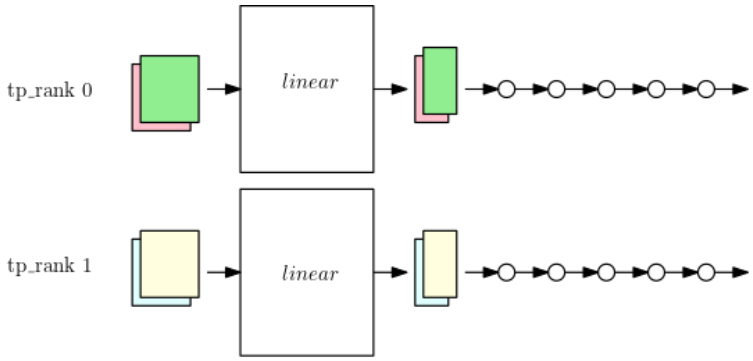
```
dp_rank 0 (tensor parallel rank 0): A, B:0, C, D, G:0, H
dp_rank 1 (tensor parallel rank 1): A, B:1, C, D, G:1, H
dp_rank 2 (tensor parallel rank 0): A, B:0, C, D, G:0, H
dp_rank 3 (tensor parallel rank 1): A, B:1, C, D, G:1, H
```

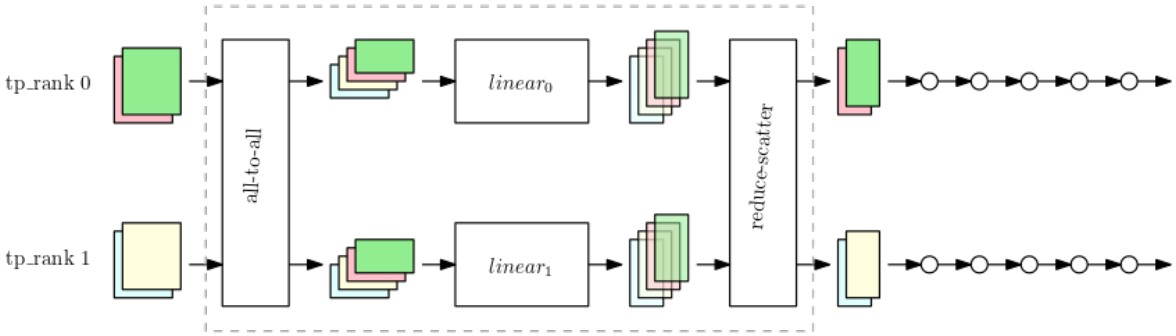
每行代表存储在其中的一组模块`dp_rank`，以及符号`X:y`表示`y`模块的第一部分`X`. 请注意以下几点：

1. 分区发生在数据并行等级的子集之间进行，我们称之为`TP_GROUP`，而不是整个`DP_GROUP`，以便在其中复制确切的模型分区`dp_rank`0 和`dp_rank`2，以及类似的横跨`dp_rank`1 和`dp_rank`3.
2. 模块`E`和`F`不再是模型的一部分，因为他们的父模块`B`已分区，并且任何通常属于其中一部分的执行`E`和`F`发生在（已分区）`B`模块。
3. 尽管`H`支持张量并行性，在本例中它不是分区的，这突出了是否对模块进行分区取决于用户的输入。一个模块受张量并行性支持这一事实并不一定意味着它被分区。

库如何使张量并行性适应 PyTorch`nn.Linear`模块

当对数据并行等级执行张量 parallel 度时，将在张量并 parallel 设备之间划分参数、渐变和优化器状态的子集对于已分区的模块. 对于其余模块，张量 parallel 器件以常规的数据并行方式运行。要执行分区模块，设备首先收集必要的部分所有数据示例跨同一张量并行度组中的对等设备。然后对所有这些数据样本执行模块的局部部分，然后再执行另一轮同步，这两轮同步结合了每个数据样本的输出部分，并将组合的数据样本返回到数据样本最初源自的 GPU。下图显示了分区上此过程的示例。`nn.Linear`模块。





第一张图显示了一个带有大号的小型模型`nn.Linear`模块的数据并行度超过两个张量并行度等级。这些区域有：`nn.Linear`模块被复制到两个 parallel 的等级中。

第二张图显示了在拆分时应用于较大模型的张量并行度`nn.Linear`模块。每个`tp_rank`拥有线性模块的一半，以及其余操作的全部。在线性模块的执行过程中，每个`tp_rank`收集所有数据样本中的相关一半并将其传递给他们的一半`nn.Linear`模块。结果需要减少分散（以求和作为缩减运算），以便每个等级都有自己的数据样本的最终线性输出。模型的其余部分以典型的数据 parallel 方式运行。

运行 SageMaker 具有 Tensor 并行性的分布式模型并行训练作业

在本节中，您将了解：

- 如何配置 SageMaker pyTorch 估算器和 SageMaker 使用张量并行度的分布式模型并行度选项。
- 如何使用扩展版调整训练脚本`smdistributed.modelparallel`用于张量并行度的模块。

要了解有关的更多信息`smdistributed.modelparallel`模块，请参阅SageMaker 分布式模型 parallel API中的SageMaker Python 开发工具包文.

主题

- 单独张量并行度 (p. 2023)
- Tensor 并行度与管道并行度相结合 (p. 2025)

单独张量并行度

以下是一个分布式训练选项的示例，该选项可以单独激活张量并行性，而不需要管道并行性。配置`mpi_options`和`smp_options`用于指定分布式训练选项的字典 SageMaker `PyTorch`估算器。

> **Note**
>
> 可以通过适用于 PyTorch 的 Deep Learning Containers 实现扩展内存节省功能，该容器实现 SageMaker 分布式模型 parallel 库 v1.6.0 或更高版本。

配置 SageMaker PyTorch 估算器

```
mpi_options = {
    "enabled" : True,
    "processes_per_host" : 8,               # 8 processes
    "custom_mpi_options" : "--mca btl_vader_single_copy_mechanism none "
}

smp_options = {
    "enabled":True,
    "parameters": {
        "pipeline_parallel_degree": 1,    # alias for "partitions"
        "placement_strategy": "cluster",
        "tensor_parallel_degree": 4,      # tp over 4 devices
        "ddp": True
    }
}

smd_mp_estimator = PyTorch(
    entry_point='your_training_script.py', # Specify
    role=role,
    instance_type='ml.p3.16xlarge',
    sagemaker_session=sagemaker_session,
    framework_version='1.10.2',
    py_version='py36',
    instance_count=1,
    distribution={
        "smdistributed": {"modelparallel": smp_options},
        "mpi": mpi_options
    },
    base_job_name="SMD-MP-demo",
)

smd_mp_estimator.fit('s3://my_bucket/my_training_data/')
```

Tip

要查找的完整参数列表`distribution`，请参阅模型并行度的配置参数在 SageMaker Python 开发工具包文档中。

适应您的 PyTorch 训练脚本

以下示例训练脚本显示了如何调整 SageMaker 分布式模型并行度库到训练脚本。在此示例中，假设脚本被命名为`your_training_script.py`.

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchnet.dataset import SplitDataset
from torchvision import datasets

import smdistributed.modelparallel.torch as smp

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
```

```
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.fc2(x)
        return F.log_softmax(x, 1)

def train(model, device, train_loader, optimizer):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        # smdistributed: Move input tensors to the GPU ID used by
        # the current process, based on the set_device call.
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target, reduction="mean")
        loss.backward()
        optimizer.step()

# smdistributed: Initialize the backend
smp.init()

# smdistributed: Set the device to the GPU ID used by the current process.
# Input tensors should be transferred to this device.
torch.cuda.set_device(smp.local_rank())
device = torch.device("cuda")

# smdistributed: Download only on a single process per instance.
# When this is not present, the file is corrupted by multiple processes trying
# to download and extract at the same time
if smp.local_rank() == 0:
    dataset = datasets.MNIST("../data", train=True, download=False)
smp.barrier()

# smdistributed: Shard the dataset based on data parallel ranks
if smp.dp_size() > 1:
    partitions_dict = {f"{i}": 1 / smp.dp_size() for i in range(smp.dp_size())}
    dataset = SplitDataset(dataset, partitions=partitions_dict)
    dataset.select(f"{smp.dp_rank()}")

train_loader = torch.utils.data.DataLoader(dataset, batch_size=64)

# smdistributed: Enable tensor parallelism for all supported modules in the model
# i.e., nn.Linear in this case. Alternatively, we can use
# smp.set_tensor_parallelism(model.fc1, True)
# to enable it only for model.fc1
with smp.tensor_parallelism():
    model = Net()

# smdistributed: Use the DistributedModel wrapper to distribute the
# modules for which tensor parallelism is enabled
model = smp.DistributedModel(model)

optimizer = optim.AdaDelta(model.parameters(), lr=4.0)
optimizer = smp.DistributedOptimizer(optimizer)

train(model, device, train_loader, optimizer)
```

Tensor 并行度与管道并行度相结合

以下是一个分布式训练选项的示例，该选项可以实现张量并行性与管道并行性相结合。 设置`mpi_options`和`smp_options`参 parallel，用于在配置 SageMaker `PyTorch`估算器。

Note

可以通过适用于 PyTorch 的 Deep Learning Containers 实现扩展内存节省功能，该容器实现 SageMaker 分布式模型 parallel 库 v1.6.0 或更高版本。

配置 SageMaker PyTorch 估算器

```
mpi_options = {
    "enabled" : True,
    "processes_per_host" : 8,               # 8 processes
    "custom_mpi_options" : "--mca btl_vader_single_copy_mechanism none "
}

smp_options = {
    "enabled":True,
    "parameters": {
    "microbatches": 4,
        "pipeline_parallel_degree": 2,    # alias for "partitions"
        "placement_strategy": "cluster",
        "tensor_parallel_degree": 2,      # tp over 2 devices
        "ddp": True
    }
}

smd_mp_estimator = PyTorch(
    entry_point='your_training_script.py', # Specify
    role=role,
    instance_type='ml.p3.16xlarge',
    sagemaker_session=sagemaker_session,
    framework_version='1.10.2',
    py_version='py36',
    instance_count=1,
    distribution={
        "smdistributed": {"modelparallel": smp_options},
        "mpi": mpi_options
    },
    base_job_name="SMD-MP-demo",
)

smd_mp_estimator.fit('s3://my_bucket/my_training_data/')
```

适应您的 PyTorch 训练脚本

以下示例训练脚本显示了如何调整 SageMaker 分布式模型并行度库到训练脚本。请注意，训练脚本现在包含`smp.step`装饰器:

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchnet.dataset import SplitDataset
from torchvision import datasets

import smdistributed.modelparallel.torch as smp

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)
```

```
    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.fc2(x)
        return F.log_softmax(x, 1)


# smdistributed: Define smp.step. Return any tensors needed outside.
@smp.step
def train_step(model, data, target):
    output = model(data)
    loss = F.nll_loss(output, target, reduction="mean")
    model.backward(loss)
    return output, loss

def train(model, device, train_loader, optimizer):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        # smdistributed: Move input tensors to the GPU ID used by
        # the current process, based on the set_device call.
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        # Return value, loss_mb is a StepOutput object
        _, loss_mb = train_step(model, data, target)

        # smdistributed: Average the loss across microbatches.
        loss = loss_mb.reduce_mean()

        optimizer.step()

# smdistributed: Initialize the backend
smp.init()

# smdistributed: Set the device to the GPU ID used by the current process.
# Input tensors should be transferred to this device.
torch.cuda.set_device(smp.local_rank())
device = torch.device("cuda")

# smdistributed: Download only on a single process per instance.
# When this is not present, the file is corrupted by multiple processes trying
# to download and extract at the same time
if smp.local_rank() == 0:
    dataset = datasets.MNIST("../data", train=True, download=False)
smp.barrier()

# smdistributed: Shard the dataset based on data parallel ranks
if smp.dp_size() > 1:
    partitions_dict = {f"{i}": 1 / smp.dp_size() for i in range(smp.dp_size())}
    dataset = SplitDataset(dataset, partitions=partitions_dict)
    dataset.select(f"{smp.dp_rank()}")

# smdistributed: Set drop_last=True to ensure that batch size is always divisible
# by the number of microbatches
train_loader = torch.utils.data.DataLoader(dataset, batch_size=64, drop_last=True)

model = Net()

# smdistributed: enable tensor parallelism only for model.fc1
smp.set_tensor_parallelism(model.fc1, True)
```

```
# smdistributed: Use the DistributedModel container to provide the model
# to be partitioned across different ranks. For the rest of the script,
# the returned DistributedModel object should be used in place of
# the model provided for DistributedModel class instantiation.
model = smp.DistributedModel(model)

optimizer = optim.AdaDelta(model.parameters(), lr=4.0)
optimizer = smp.DistributedOptimizer(optimizer)

train(model, device, train_loader, optimizer)
```

使用 Tensor 并行度进行检查点的说明

这些区域有： SageMaker 模型 parallel 库支持使用张量并行性来保存部分或完整的检查点。以下指南介绍了在使用 tensor 并行机制时如何修改脚本以保存和加载检查点。

1. 准备一个模型对象并用库的包装函数包装它`smp.DistributedModel()`.

   ```
   model = MyModel(...)
   model = smp.DistributedModel(model)
   ```

2. 为模型准备优化器。一组模型参数是优化器函数所需的可迭代参数。要准备一组模型参数，必须处理`model.parameters()`为单个模型参数分配唯一 ID。

   如果模型参数可迭代对象中存在具有重复 ID 的参数，则加载检查点优化程序状态将失败。要为优化程序创建具有唯一 ID 的模型参数的可迭代对象，请参阅以下内容：

   ```
   unique_params = []
   unique_params_set = set()
   for p in model.parameters():
     if p not in unique_params_set:
       unique_params.append(p)
       unique_params_set.add(p)
   del unique_params_set

   optimizer = MyOpt(unique_params, ...)
   ```

3. 使用库的包装函数包装优化器`smp.DistributedOptimizer()`.

   ```
   optimizer = smp.DistributedOptimizer(optimizer)
   ```

4. 使用保存模型和优化程序状态`smp.save()`. 根据您要保存检查点的方式，请选择以下两个选项之一：

   - 选项 1：在每个模型上保存部分模型`mp_rank`单个`MP_GROUP`.

     ```
     model_dict = model.local_state_dict() # save a partial model
     opt_dict = optimizer.local_state_dict() # save a partial optimizer state
     # Save the dictionaries at rdp_rank 0 as a checkpoint
     if smp.rdp_rank() == 0:
         smp.save(
             {"model_state_dict": model_dict, "optimizer_state_dict": opt_dict},
             f"/checkpoint.pt",
             partial=True,
         )
     ```

     使用 tensor 并行性，库保存以下格式命名的检查点文件：`checkpoint.pt_{pp_rank}_{tp_rank}`.

     Note

     使用张量并行度，请务必将 if 语句设置为`if smp.rdp_rank() == 0`而不是`if smp.dp_rank() == 0`. 当优化程序状态与张量并行分区时，所有减少的数据并 parallel 排名

都必须保存自己的优化程序状态分区。使用错误如果检查点声明可能会导致培训工作停滞。有关使用的更多信息`if smp.dp_rank() == 0`没有张量并行性，请参见保存和加载的一般说明中的SageMaker Python 开发工具包文.

- 选项 2：保存完整模型。

```
if smp.rdp_rank() == 0:
    model_dict = model.state_dict(gather_to_rank0=True) # save the full model
    if smp.rank() == 0:
        smp.save(
            {"model_state_dict": model_dict},
            "/checkpoint.pt",
            partial=False,
        )
```

Note

请考虑以下内容进行完整的检查点：

- 如果你设置`gather_to_rank0=True`，除以外的所有级别`0`返回空字典。
- 对于完整的检查点，你只能检查模型。目前不支持优化程序状态的完整检查点。
- 完整模型只需要保存在`smp.rank() == 0`.

5. 使用加载检查点`smp.load()`. 根据您在上一步中检查点的方式，选择以下两个选项之一：
   - 选项 1：加载部分检查点。

```
checkpoint = smp.load("/checkpoint.pt", partial=True)
model.load_state_dict(checkpoint["model_state_dict"], same_partition_load=False)
optimizer.load_state_dict(checkpoint["optimizer_state_dict"])
```

您可以设置`same_partition_load=True`在`model.load_state_dict()`为了更快地加载，如果你知道分区不会改变。

   - 选项 2：加载完整的检查点。

```
if smp.rdp_rank() == 0:
    checkpoint = smp.load("/checkpoint.pt", partial=False)
    model.load_state_dict(checkpoint["model_state_dict"])
```

这些区域有：`if smp.rdp_rank() == 0`条件不是必需的，但它可以帮助避免不同之间的冗余装载`MP_GROUP`。张量并行性目前不支持完整的检查点优化器状态 dict。

Support 拥抱脸变压器模型

这些区域有： SageMaker 模型 parallel 库的张量并行性提供 out-of-the-box 支持以下拥抱脸变压器型号：

- GPT-2、BERT 和 Roberta（可在 SageMaker 模型 parallel 库 v1.7.0 及更高版本）
- GPT-J（可在 SageMaker 模型 parallel 库 v1.8.0 及更高版本）

Note

对于任何其他变形金刚型号，你需要使用smp.tp_注册er_Wit_module ()API 来应用张量并行性。

Note

要使用 tensor 并行性来训练 Huging Face 变压器模型，你应该添加`transformers==4.4.2`到`requirements.txt`用于在 PyTorch v1.8.1 及更高版本的深度学习容器中安装变形金刚库。

如果您使用其中一个拥抱 Face 变压器模型，则无需手动实现挂钩即可将变压器 API 转换为smdistributed变压器层。你也不需要使用smp.tp_registerAPI。你可以通过使用上下文管理器激活张量并行smp.tensor_parallelism()然后通过包装模型smp.DistributedModel(). 例如，请参阅以下代码：

```
from transformers import AutoModelForCausalLM

with smp.tensor_parallelism():
    model = AutoModelForCausalLM.from_config(hf_gpt2_config)

model = smp.DistributedModel(model)
```

此外，鉴于state_dict来自 的DistributedModel对象，你可以将权重加载到原始 HuggingFace GPT-2 模型使用translate_state_dict_to_hf_gpt2API：

```
from smdistributed.modelparallel.torch.nn.huggingface.gpt2 \
                                        import translate_state_dict_to_hf_gpt2
max_seq_len = 1024

with smp.tensor_parallelism():
    model = AutoModelForCausalLM.from_config(hf_gpt2_config)

model = smp.DistributedModel(model)

# [...run training...]

if smp.rdp_rank() == 0:
    state_dict = dist_model.state_dict()
    hf_state_dict = translate_state_dict_to_hf_gpt2(state_dict, max_seq_len)

    # can now call model.load_state_dict(hf_state_dict) to the original HF model
```

同样，给定受支持 HuggingFace 模型state_dict，您可以使用translate_hf_state_dict_to_smdistributedAPI 将其转换为可读的格式smp.DistributedModel. 这在转移学习用例中非常有用，在这种情况下，预训练的模型被加载到smp.DistributedModel对于模型并行微调：

```
from smdistributed.modelparallel.torch.nn.huggingface.roberta \
                                        import translate_state_dict_to_smdistributed

model = AutoModelForMaskedLM.from_config(roberta_config)
model = smp.DistributedModel(model)

pretrained_model = AutoModelForMaskedLM.from_pretrained("roberta-large")
translated_state_dict =
        translate_state_dict_to_smdistributed(pretrained_model.state_dict())

# load the translated pretrained weights into the smp.DistributedModel
model.load_state_dict(translated_state_dict)

# start fine-tuning...
```

相关state_dict拥抱 Face 和之间的翻译功能smp可以按照如下方式访问。

- from smdistributed.modelparallel.torch.nn.huggingface.gpt2 import translate_state_dict_to_hf_gpt2
- from smdistributed.modelparallel.torch.nn.huggingface.gpt2 import translate_hf_state_dict_to_smdistributed
- from smdistributed.modelparallel.torch.nn.huggingface.bert import translate_state_dict_to_hf_bert

- `from smdistributed.modelparallel.torch.nn.huggingface.bert import translate_hf_state_dict_to_smdistributed`
- `from smdistributed.modelparallel.torch.nn.huggingface.roberta import translate_state_dict_to_hf_roberta`
- `from smdistributed.modelparallel.torch.nn.huggingface.roberta import translate_hf_state_dict_to_smdistributed`
- `from smdistributed.modelparallel.torch.nn.huggingface.gptj import translate_hf_gptj_state_dict_to_smdistributed`（在中可用 SageMaker 模型 parallel 库 v1.8.0 及更高版本）
- `from smdistributed.modelparallel.torch.nn.huggingface.gptj import translate_state_dict_to_hf_gptj`（在中可用 SageMaker 模型 parallel 库 v1.8.0 及更高版本）

分片优化器状态

优化程序状态分片是一种有用的节省内存的技术，它可以跨数据并行设备组对优化程序状态（描述优化程序状态的权重集）进行分片。无论何时使用有状态优化器（例如亚当）或 FP16 优化器（存储参数的 FP16 和 FP32 副本），都可以使用优化程序状态分片。

如何使用优化器状态分片

优化程序状态分片可以通过设置打开`"shard_optimizer_state": True`中的`modelparallel`配置。

启用此功能后，库将根据数据并行度对模型参数集进行分区。对应于`i`'第个分区只能在`i`'第个数据 parallel 排名。在第一次通话结束时`smp.step`装饰器函数，包装的优化器`smp.DistributedOptimizer`将其参数重新定义为仅限于与当前数据 parallel 排名的分区相对应的参数。重新定义的参数被称为`virtual_parameters`并使用原始参数共享底层存储。在第一次打电话期间`optimizer.step`，优化程序状态是基于这些重新定义的参数创建的，这些参数是由于原始分区而进行分片的。优化程序更新后，`allgather`操作（作为`optimizer.step`调用）在数据并行等级中执行，以实现一致的参数状态。

> **Tip**
>
> 当数据并行度大于 1 且模型具有超过十亿个参数时，优化程序状态分片可能非常有用。
> 数据并行度的计算方法是`(processes_per_host * instance_count / pipeline_parallel_degree)`，以及`smp.dp_size()`函数在后台处理大小。

配置 SageMaker PyTorch 估算器

```
mpi_options = {
    "enabled" : True,
    "processes_per_host" : 8,               # 8 processes
    "custom_mpi_options" : "--mca btl_vader_single_copy_mechanism none "
}

smp_options = {
    "enabled":True,
    "parameters": {
        "microbatches": 4,
        "pipeline_parallel_degree": 2,    # alias for "partitions"
        "placement_strategy": "cluster",
        "tensor_parallel_degree": 2,      # tp over 2 devices
        "ddp": True,
        "shard_optimizer_state": True
    }
}
```

适应您的 PyTorch 训练脚本

请参阅适应您的 PyTorch 训练脚本 (p. 2026)中的Tensor 并行度与管道并行度相结合部分。脚本不需要进行额外的修改。

激活检查点

激活检查点（或渐变检查点）是一种通过清除某些层的激活并在向后过程中重新计算它们来减少内存使用的技术。实际上，这会交换额外的计算时间以减少内存使用量。如果模块处于选中点状态，则在向前传结束时，模块的输入和输出将保留在内存中。任何本来是该模块内计算的一部分的中间张量都会在向前传递期间释放出来。在检查点模块的向后传递期间，这些张量被重新计算。此时，超出此检查点模块的层已经完成了向后传递，因此带检查点的峰值内存使用率可能会降低。

如何使用激活检查点

与`smdistributed.modelparallel`，你可以在模块的粒度上使用激活检查点。对于所有`torch.nn`模块除外`torch.nn.Sequential`，只有从管道并行度的角度来看，如果模块树位于一个分区内，才能检查模块树。如果是`torch.nn.Sequential`模块，顺序模块内的每个模块树都需要完全位于一个分区内才能启用激活检查点才能正常工作。当您使用手动分区时，您需要了解这些限制。

当您使用自动模型分区，你可以找到分区分配日志`Partition assignments:`在培训作业日志中。如果一个模块被分成多个等级（例如，一个子体在一个等级，另一个子体处于不同等级），则库将忽略检查模块的尝试，并引发一条警告消息，说明该模块不会被检查点。

Note

这些区域有： SageMaker 模型并行库同时支持重叠和非重叠`allreduce`结合检查点。

Note

PyTorch 的原生检查点 API 不兼容`smdistributed.modelparallel`.

示例 1：以下示例代码显示了在脚本中有模型定义时如何使用激活检查点。

```
import torch.nn as nn
import torch.nn.functional as F

from smdistributed.modelparallel.torch.patches.checkpoint import checkpoint

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = F.max_pool2d(x, 2)
        x = torch.flatten(x, 1)
        # This call of fc1 will be checkpointed
        x = checkpoint(self.fc1, x)
        x = self.fc2(x)
        return F.log_softmax(x, 1)
```

示例 2：以下示例代码显示了在脚本中有顺序模型时如何使用激活检查点。

```
import torch.nn as nn
from smdistributed.modelparallel.torch.patches.checkpoint import checkpoint_sequential
```

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.seq = nn.Sequential(
            nn.Conv2d(1,20,5),
            nn.ReLU(),
            nn.Conv2d(20,64,5),
            nn.ReLU()
        )

    def forward(self, x):
        # This call of self.seq will be checkpointed
        x = checkpoint_sequential(self.seq, x)
        return F.log_softmax(x, 1)
```

示例 3：以下示例代码显示了从库导入预构建模型时如何使用激活检查点，例如 PyTorch 和拥抱脸变形金刚。无论您是否检查点顺序模块，您都需要执行以下操作：

1. 将模型包裹起来`smp.DistributedModel()`.
2. 为顺序图层定义对象。
3. 按顺序图层对象包裹`smp.set_activation_checkpointig()`.

```
import smdistributed.modelparallel.torch as smp
from transformers import AutoModelForCausalLM

smp.init()
model = AutoModelForCausalLM(*args, **kwargs)
model = smp.DistributedModel(model)

# Call set_activation_checkpointing API
transformer_layers = model.module.module.module.transformer.seq_layers
smp.set_activation_checkpointing(
    transformer_layers, pack_args_as_tuple=True, strategy='each')
```

激活分载

当启用激活检查点和管道并行性且微批次数大于 1 时，激活分载是可以进一步减少内存使用的额外功能。激活卸载异步移动与当前未在 CPU 中运行的微批对应的检查点激活。就在 GPU 需要激活来实现微批次的向后传递之前，此功能会预先从 CPU 中取回卸载的激活。

如何使用激活卸载

在下列时，使用激活卸载以减少内存使用微批次数大于 1，并启用激活检查点（请参阅激活检查点 (p. 2032)）。如果不使用激活检查点，激活卸载将不起作用。当它只与一个微批处理一起使用时，它不会节省内存。

要使用激活卸载，请设置`"offload_activations": True`中的`modelparallel`配置。

激活卸载将检查点激活移动到`nn.Sequential`模块异步到 CPU。通过 PCIe 链路传输的数据与 GPU 计算重叠。一旦计算特定检查点图层的前向通道，卸载立即发生。在特定微批次的向后传递需要激活之前不久将激活加载回 GPU。CPU-GPU 传输与计算类似重叠。

要调整激活加载回 GPU 的早期，可以使用配置参数`"activation_loading_horizon"`（默认设置为 4，必须是`int`大于 0）。更大的激活加载水平将导致激活更早地加载回 GPU。如果地平线太大，激活卸载对内存节省的影响可能会降低。如果地平线太小，激活可能无法及时加载，从而减少重叠量并降低性能。

> Tip
>
> 激活卸载对于具有超过 1000 亿个参数的大型模型非常有用。

配置 SageMaker PyTorch 估算器

```
mpi_options = {
    "enabled" : True,
    "processes_per_host" : 8,               # 8 processes
    "custom_mpi_options" : "--mca btl_vader_single_copy_mechanism none "
}

smp_options = {
    "enabled":True,
    "parameters": {
        "microbatches": 4,
        "pipeline_parallel_degree": 2,    # alias for "partitions"
        "placement_strategy": "cluster",
        "tensor_parallel_degree": 2,      # tp over 2 devices
        "ddp": True,
        "offload_activations": True,
        "activation_loading_horizon": 4   # optional. default is 4.
    }
}
```

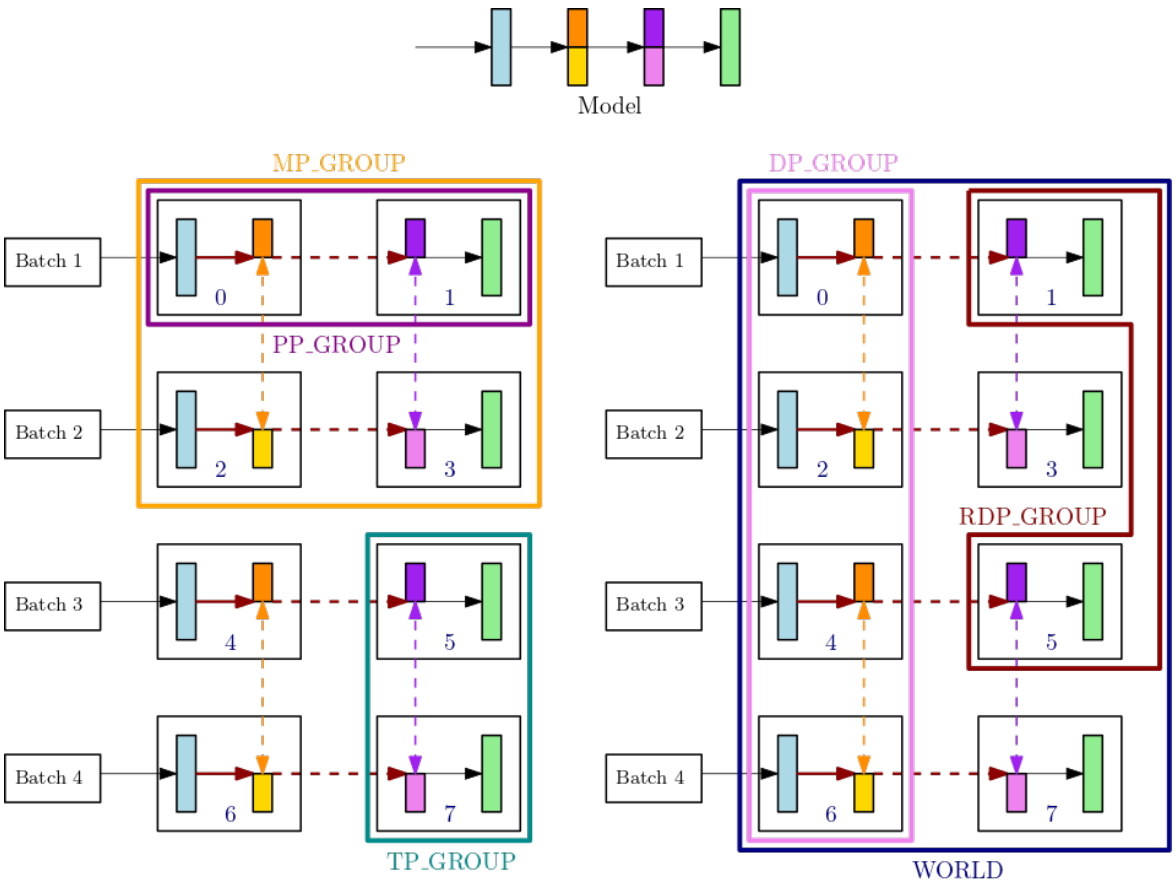
排名机制

本节介绍了模型并行度的排名机制如何与张量并行性一起工作。这是从排名基础为了SageMaker 模型并行库的核心功能 (p. 1999). 借助 tensor 并行性，该库引入了三种类型的排名和流程组 API：smp.tp_rank()为了张量 parallel 等级、smp.pp_rank()为了管道 parallel 排名, 和smp.rdp_rank()为了减少数据 parallel 排名. 相应的沟通流程组包括张量 parallel 组(TP_GROUP),管道 parallel 组(PP_GROUP)，以及减少的数据 parallel 组(RDP_GROUP)。这些组的定义如下：

- 张量 parallel 组 (TP_GROUP) 是数据 parallel 组的一个均匀可分割的子集，模块的张量 parallel 分布发生在该组上。当管道并行度为 1 时TP_GROUP和相同模型 parallel 组(MP_GROUP)。
- 管道 parallel 组 (PP_GROUP) 是管道并行性发生的一组流程。当张量并行度度为 1 时PP_GROUP和相同MP_GROUP.
- 减少的数据 parallel 组 (RDP_GROUP) 是一组进程，它们保存相同的管道并行分区以及相同的张量并行分区，并在它们之间执行数据并行处理。这被称为减少的数据 parallel 组，因为它是整个数据并行度组的一个子集，DP_GROUP. 对于分布在TP_GROUP，渐变allreduce操作仅对减少的数据 parallel 组执行，而对于未分布的参数，则为梯度allreduce发生在整个DP_GROUP.
- 模型 parallel 组 (MP_GROUP) 是指集体存储整个模型的一组进程。它由联盟组成PP_GROUP在所有队伍中的STP_GROUP当前的流程。当张量并行度为 1 时MP_GROUP等同于PP_GROUP. 它也符合现有的MP_GROUP来自以前smdistributed版本。请注意，当前TP_GROUP是当前两者的子集DP_GROUP和最新的MP_GROUP.

要了解有关通信流程 API 的更多信息，请参阅 SageMaker 分布式模型并行度库，请参阅常见 API和PyTorch 特定于 API中的SageMaker Python 开发工具包文.

下图说明了排名机制、参数分布以及相关的allreduce运算。



上图说明了具有 8 个 GPU 的节点的进程组，其中张量并行度为 2，管道并行度为 2，数据并行度为 4。第一张图显示了一个具有 4 层的示例模型。在其他图中，4 层模型分布在 4 个 GPU 中，使用管道并行度和张量并行度，其中张量并行度用于中间两层。这些数字是简单的副本，用于说明不同的群体边界线。为跨 GPU 0-3 和 4-7 进行数据并行性复制了分区模型。左下图显示了`MP_GROUP`、`PP_GROUP`, 和`TP_GROUP`. 右下图显示`RDP_GROUP`、`DP_GROUP`, 和`WORLD`在同一组 GPU 上。具有相同颜色的图层和图层切片的渐变为`allreduced` 共同实现数据并行性。例如，第一层（浅蓝色）获取`allreduce`横向的运算`DP_GROUP`，而第二层中的深橙色切片只能获得`allreduce`内的操作`RDP_GROUP`它的过程。大胆的暗红色箭头代表张量及其整批`TP_GROUP`.

```
GPU0: pp_rank 0, tp_rank 0, rdp_rank 0, dp_rank 0, mp_rank 0
GPU1: pp_rank 1, tp_rank 0, rdp_rank 0, dp_rank 0, mp_rank 1
GPU2: pp_rank 0, tp_rank 1, rdp_rank 0, dp_rank 1, mp_rank 2
GPU3: pp_rank 1, tp_rank 1, rdp_rank 0, dp_rank 1, mp_rank 3
GPU4: pp_rank 0, tp_rank 0, rdp_rank 1, dp_rank 2, mp_rank 0
GPU5: pp_rank 1, tp_rank 0, rdp_rank 1, dp_rank 2, mp_rank 1
GPU6: pp_rank 0, tp_rank 1, rdp_rank 1, dp_rank 3, mp_rank 2
GPU7: pp_rank 1, tp_rank 1, rdp_rank 1, dp_rank 3, mp_rank 3
```

在此示例中，管道并行性发生在 GPU 对 (0,1); (2,3); (4,5) 和 (6,7) 之间。此外，我们还有数据并行性（`allreduce`) 在 GPU 0、2、4、6 之间进行，并独立于 GPU 1、3、5、7。Tensor 并行性发生在`dp_groups`，跨 GPU 对 (0,2); (1,3); (4,6) 和 (5,7)。

请注意，在这种类型的张量并行度下，`(the degree of data parallelism)*(the degree of pipeline parallelism) = (the number of GPUs)`.

SageMaker 分布式模型并行最佳实

使用 SageMaker 模型并行库运行分布式训练作业时，请遵循以下准则。

为给定模型设置正确的配置

在扩展模型时，我们建议您按顺序浏览以下列表。每个列表项都讨论了使用图书馆技术的优势以及可能出现的权衡。

> Tip
>
> 如果使用库的一部分功能可以很好地配合模型，那么添加更多的模型并行性或节省内存的功能通常不会提高性能。

使用大型 GPU 实例类型

- 在模型并行性领域，最好使用具有大量 GPU 内存的强大实例来处理模型并行操作（例如跨多个 GPU 分区模型）的开销。我们建议使用`ml.p4d`要么`ml.p3dn`用于训练大型 DL 模型的实例。这些实例还配备了 Elastic Fabric Adapter (EFA)，该适配器可提供更高的网络带宽，并支持具有模型并行性的大规模训练。

分片优化程序状态

- 分片优化程序状态的影响取决于数据并行排名数。通常，更高程度的数据并行度（与计算节点的大小成正比）可以提高内存使用效率。

  当您想缩小群集规模时，请确保检查优化程序状态分片配置。例如，具有优化程序状态分片的大型 DL 模型适合 16 个 GPU 的节点不适合具有 8 个 GPU 的节点，因为根本没有足够的 GPU 来分片优化程序状态。

  有关更多信息，请参阅 分片优化器状态 (p. 2031)。

激活检查点

- 通过对一组模块使用激活检查点可以提高内存效率。对模块进行分组越多，内存使用就越有效。当检查图层的顺序模块时，`strategy`参数`smp.set_activation_checkpointing`函数将图层分组在一起以进行检查点。例如，将两个或更多图层组合在一起以进行检查点比一次检查点一个图层更有效的内存效率，而这将额外的计算时间以减少内存使用量。

  有关更多信息，请参阅 激活检查点 (p. 2032)。

Tensor 并行

- 张量并行度应该是二（2、4、8、...、2）的幂^{n}），其中最大程度必须等于每个节点的 GPU 数量。例如，如果您使用具有 8 个 GPU 的节点，则张量并行度的可能数字为 2、4 和 8。对于张量并行度，我们不建议任意数字（例如 3、5、6 和 7）。当你使用多个节点时，错误配置张量并行度可能会导致跨节点运行张量并行度；这增加了跨节点激活的通信所产生的巨大开销，并且可能会变得昂贵的计算成本。

  有关更多信息，请参阅 Tensor 并行度 (p. 2021)。

跨节点的管道并行性

- 您可以在单个节点内和多个节点之间运行管道并行。当您将管道并行与张量并行结合使用时，我们建议在多个节点之间运行管道并行性，并在单个节点内保持张量并行度。
- 管道并行度具有以下三个旋钮：`microbatches`、`active_microbatches`, 和`prescaled_batch`.

  - 当你在管道并行度中使用张量并行度时，我们建议激活`prescaled_batch`以便可以增加每个模型并行组的批量大小以实现高效的流水线。与`prescaled_batch`激活时，训练脚本中设置的批量大小变为`tp_size`乘以为每个等级设置的批量大小的倍数，`prescaled_batch`.
  - 增加数量`microbatches`有助于实现高效的流水线和更好的性能。请注意，有效微批量大小是批量大小除以微批次数。如果在保持批量大小不变的同时增加微批次的数量，则每个微批处理的样品会减少。
  - 的数量`active_microbatches`是流水线输送过程中同时处理的微批量的最大数量。对于处理中的每个活动微批量，其激活和渐变会占用 GPU 内存。因此，增加`active_microbatches`占用更多 GPU 内存。
- 如果 GPU 和 GPU 内存均未充分利用，请增加`active_microbatches`以便在流水线过程中实现更好的并行化。
- 有关如何将 Tensor 并行度与管道并行度用法的更多信息，请参阅Tensor 并行度与管道并行度相结合 (p. 2025).
- 要查找上述参数的说明，请参阅的参数`smdistributed`中的SageMaker Python 开发工具包.

将激活卸载到 CPU

- 确保将其与激活检查点和管道并行性结合使用。为确保卸载和预加载在后台进行，请为 microbatch 参数指定一个大于 1 的值。
- 当卸载激活时，您也许能够增加`active_microbatches`有时候与微粒的总数相匹配。这取决于哪些模块被检查点以及模型的分区方式。

  有关更多信息，请参阅 激活分载 (p. 2033)。

配置引用

SageMaker 分布式模型并行训练团队根据 GPT-2 模型的实验、序列长度为 512 和 50,000 的词汇量提供了以下参考点。

数量模型参数	实例类型	管道并行	Tensor 并行	优化器状态分片	激活检查点	批量预扩展	Batch 大小
100 亿	16ml.p4d.24xlarge	4	4	True	每个变压器层	True	`batch_size=40`
300 亿	16ml.p4d.24xlarge	8	4	True	每个变压器层	True	`batch_size=32`
600 亿	32ml.p4d.24xlarge	8	8	True	每个变压器层	True	`batch_size=56, microbatches=4, active_microbatches`

您可以从前面的配置中推断出来估计模型配置的 GPU 内存使用情况。例如，如果您增加 10 亿个参数模型的序列长度或将模型的大小增加到 200 亿，则可能需要先降低批量大小。如果模型仍然不合适，请尝试提高张量并行度。

修改训练脚本

- 在训练脚本中使用 SageMaker 模型并行库的功能之前，请查看SageMaker 分布式模型并行配置提示和陷阱 (p. 2038).
- 要更快地启动培训作业，请使用SageMaker 本地模式. 这有助于您在 SageMaker 笔记本实例上快速在本地运行培训作业。根据运行 SageMaker 笔记本实例的机器学习实例的比例，您可能需要通过更改模型配置（例如隐藏宽度、变压器层数和注意头）来调整模型的大小。在使用大型集群训练完整模型之前，验证简化模型是否在笔记本实例上运行良好。

使用 SageMaker 控制台和 Amazon CloudWatch 监控和记录培训 Job

要监控系统级指标，例如 CPU 内存利用率、GPU 内存利用率和 GPU 利用率，请使用SageMaker 控制台.

1. 在左侧导航窗格中，选择训练.
2. 选择 Training jobs (训练作业)。
3. 在主窗格中，选择要查看更多详细信息的培训作业名称。
4. 浏览主窗格并找到显示器部分以查看自动可视化。
5. 要查看培训作业日志，请选择查看日志中的显示器部分。您可以在 CloudWatch 中访问培训作业的分布式培训作业日志。如果您启动了多节点分布式培训，则应该看到多个日志流，其格式为算法 n-1234567890. 这些区域有：算法 -1日志流跟踪来自主（第 0 个）节点的训练日志。

有关更多信息，请参阅 使用 Amazon CloudWatch 指标监控和分析训练作业 (p. 2093)。

权限

要运行具有模型并行性的 SageMaker 培训作业，或者SageMaker 分布式培训示例笔记，请确保您在 IAM 角色中拥有正确的权限，例如以下权限：

- 使用FSx for Lustre添加`AmazonFSxFullAccess`.
- 要使用 Amazon S3 作为数据渠道，请添加`AmazonS3FullAccess`.
- 要使用 Docker、构建自己的容器并将其推送到 Amazon ECR，请添加`AmazonEC2ContainerRegistryFullAccess`.
- 要拥有使用整个 SageMaker 功能套件的完全访问权限，请添加`AmazonSageMakerFullAccess`.

SageMaker 分布式模型并行配置提示和陷阱

在使用 Amazon SageMaker 的分布式模型并行库之前，请查看以下提示和陷阱。此列表包括适用于各个框架的提示。有关 TensorFlow 和 PyTorch 的具体提示，请参阅修改 TensorFlow 训练脚本 (p. 2003)和修改 PyTorch 训练脚本 (p. 2009)，分别。

Batch 大小和微批次数

- 当批量大小增加时，库效率最高。对于模型适合于单个设备但只能用小批量进行训练的用例，可以而且应该在集成库后增加批量大小。模型并行性可为大型模型节省内存，使您能够使用以前不适合内存的批量大小进行训练。
- 选择许多太小或太大的微粒可以降低性能。库在每台设备中按顺序执行每个微批处理，因此微批量大小（批量大小除以微批次数）必须足够大，以充分利用每个 GPU。与此同时，管道效率随着微批次的数量而提高，因此实现适当的平衡非常重要。通常，一个很好的起点是尝试 2 或 4 个微批，将批量大小增加到内存限制，然后尝试更大的批量和微批次数。随着微批次数量的增加，如果使用交错管道，可能会变得更大的批量大小可能变得可行。
- 您的批量大小必须始终被微批量的数量整除。请注意，根据数据集的大小，有时每个时代的最后一批大小可能比其他时代小，而且这个较小的批次也需要被微批量的数量整除。如果不是，您可以设置`drop_remainder=True`中的`tf.Dataset.batch()`调用（在 TensorFlow 中），或者设置`drop_last=True`在`DataLoader`（在 PyTorch 中），以便不使用最后一个小批量。如果您为数据管道使用不同的 API，则可能需要手动跳过最后一批次，只要不能被微批次数整除，则可能需要手动跳过该批次。

手动分区

- 如果使用手动分区，请注意模型中的多个操作和模块所占用的参数，例如变压器体系结构中的嵌入表。为了正确起见，共享相同参数的模块必须放在同一设备中。使用自动分区时，库会自动强制执行此约束。

数据准备

- 如果模型需要多个输入，请确保在数据管道中进行随机操作（例如，洗牌）`smp.dp_rank()`. 如果数据集在数据并行设备之间进行确定性分片，请确保分片的索引是`smp.dp_rank()`. 这是为了确保构成模型分区的所有级别上看到的数据顺序一致。

从返回张量`smp.DistributedModel`

- 从中返回的任何张量`smp.DistributedModel.call`（对于 TensorFlow）或`smp.DistributedModel.forward`（对于 PyTorch）函数被广播到所有其他级别，从计算特定张量的排名中。因此，不应返回在调用和转发方法之外不需要的任何张量（例如中间激活），因为这会导致不必要的通信和内存开销并损害性能。

这些区域有：`@smp.step`装饰器

- 如果`smp.step`-装饰函数有一个不具有批处理维度的 tensor 参数，参数名称必须在`non_split_inputs`打电话时列出`smp.step`. 这可以防止库尝试将张量分割成微批量。有关更多信息，请参阅`smp.step`在 API 文档中。

延迟参数初始化

对于超过 1000 亿个参数的大型型号，通过 CPU 内存进行权重初始化可能会导致内存不足错误。为了解决这个问题，图书馆提供`smp.delay_param_initialization`上下文管理器。这会延迟参数的物理分配，直到它们在首次执行`smp.step`-装饰功能。这避免了训练初始化期间 CPU 不必要的内存使用。创建模型对象时，请使用上下文管理器，如以下代码中所示。

```
with smp.delay_param_initialization(enabled=True):
    model = MyModel()
```

PyTorch 的张量并行性

- 如果你使用种子获得确定性结果，请根据`smp.dp_rank()`（例如，`torch.manual_seed(42 + smp.dp_rank())`）。如果你不这样做，那么不同的分区`nn.Parameter`以同样的方式初始化，影响趋同。
- SageMaker 的模型并行度库使用 NCCL 来实现分发模块所需的集体内容。特别是对于较小型号，如果同时在 GPU 上安排太多的 NCCL 调用，则由于 NCCL 使用的额外空间，内存使用量可能会增加。为了抵消这种情况，`smp`限制 NCCL 呼叫，以便任何给定时间正在进行的 NCCL 操作数量小于或等于给定的限制。默认限制为 8，但可以使用环境变量调整`SMP_NCCL_THROTTLE_LIMIT`. 如果在使用 tensor 并行性时观察到的内存使用量超出预期，则可以尝试降低此限制。但是，选择太小的限制可能会导致吞吐量损失。要完全禁用限制，你可以设置`SMP_NCCL_THROTTLE_LIMIT=-1`.
- 当张量并行度为 1 时，以下标识在张量并行度大于 1 时将保持，当张量并行度大于 1 时不会保持：`smp.mp_size() * smp.dp_size() == smp.size()`. 这是因为张量并行组既是模型并行度组又是数据并行度组的一部分。如果你的代码现有引用`mp_rank`、`mp_size`、`MP_GROUP`等等，如果你想只使用管道并行组，则可能需要将引用替换为`smp.pp_size()`. 以下身份永远是真的：
  - `smp.mp_size() * smp.rdp_size() == smp.size()`
  - `smp.pp_size() * smp.dp_size() == smp.size()`
  - `smp.pp_size() * smp.tp_size() * smp.rdp_size() == smp.size()`
- 由于`smp.DistributedModel`封装器在启用了张量并行度时修改模型参数，应在调用之后创建优化器`smp.DistributedModel`，带有分布式参数。例如，下面的操作不起作用：

```
## WRONG
model = MyModel()
optimizer = SomeOptimizer(model.parameters())
```

```
model = smp.DistributedModel(model)  # optimizer now has outdated parameters!
```

相反，应该使用smp.DistributedModel如下所示：

```
## CORRECT
model = smp.DistributedModel(MyModel())
optimizer = SomeOptimizer(model.optimizers())
```

- 当一个模块通过张量并行性替换为分布式对应模块时，分布式模块不会继承原始模块的权重，而是初始化新的权重。这意味着，例如，如果需要在特定调用中初始化权重（例如，通过load_state_dict打电话），这需要在smp.DistributedModel在模块分发发生后调用。
- 直接访问分布式模块的参数时，请注意权重与原始模块的形状不同。例如，

```
with smp.tensor_parallelism():
    linear = nn.Linear(60, 60)

# will pass
assert tuple(linear.weight.shape) == (60, 60)

distributed_linear = smp.DistributedModel(linear)

# will fail. the number of input channels will have been divided by smp.tp_size()
assert tuple(distributed_linear.module.weight.shape) == (60, 60)
```

- 使用torch.utils.data.distributed.DistributedSampler强烈建议用于张量并行性。这可确保每个数据并行排名接收相同数量的数据样本，从而防止因不同而可能导致的挂起dp_ranks 采取了不同数量的步骤。
- 如果您将joinPyTorch 的 APIDistributedDataParallel类来处理不同数据并行排名具有不同批次数的情况，您仍然需要确保排名处于相同TP_GROUP批次数相同；否则模块的分布式执行中使用的通信集体可能会挂起。不同的等级TP_GROUPs 可以有不同数量的批次，只要join使用 API。
- 如果要检查模型并使用张量并行性，请考虑以下几点：
  - 为避免在使用张量并行时保存和加载模型时停滞和竞争条件，请确保在减少的数据并行度排名中调用以下模型和优化程序状态中的适当函数。
  - 如果要转换现有的管道并行脚本并为脚本启用张量并行，请确保修改任何if smp.dp_rank() == 0用于保存和加载的块if smp.rdp_rank() == 0块。否则，这可能会导致你的训练工作停顿。
  - 张量并行度目前不支持优化程序状态的完整检查点。如果你想从检查点恢复训练，请改用部分检查点。

  有关检查点具有张量并行性的模型的更多信息，请参阅使用 Tensor 并行度进行检查点的说明 (p. 2028).

模型并行疑难解

如果遇到错误，可以使用以下列表尝试对训练作业进行故障排除。如果问题仍存在，请联系Amazon支持.

主题

- 将 SageMaker 调试器与 SageMaker 分布式模型并行结合使用的注 (p. 2040)
- 保存检查点 (p. 2041)
- 使用模型并行和 TensorFlow 进行收敛 (p. 2042)
- 停顿或崩溃分布式培训工作 (p. 2042)
- NCCL 错误 (p. 2043)

将 SageMaker 调试器与 SageMaker 分布式模型并行结合使用的注

SageMaker 调试器不适用于 SageMaker 分布式模型并行模型。默认情况下，对于所有 SageMaker TensorFlow 和 PyTorch 训练作业启用调试器，您可能会看到如下所示的错误：

```
FileNotFoundError: [Errno 2] No such file or directory: '/opt/ml/checkpoints/
metadata.json.sagemaker-uploading
```

要修复此问题，请通过传递以禁用调试器`debugger_hook_config=False`创建框架时`estimator`如以下示例所示。

```
bucket=sagemaker.Session().default_bucket()
base_job_name="sagemaker-checkpoint-test"
checkpoint_in_bucket="checkpoints"

# The S3 URI to store the checkpoints
checkpoint_s3_bucket="s3://{}/{}/{}".format(bucket, base_job_name, checkpoint_in_bucket)

estimator = TensorFlow(
    ...

    distribution={"smdistributed": {"modelparallel": { "enabled": True }}},
    checkpoint_s3_uri=checkpoint_s3_bucket,
    checkpoint_local_path="/opt/ml/checkpoints",
    debugger_hook_config=False
)
```

保存检查点

在 SageMaker 上保存大型模型检查点时，您可能会遇到以下错误：

```
InternalServerError: We encountered an internal error. Please try again
```

这可能是由于在训练期间将本地检查点上传到 Amazon S3 时存在 SageMaker 限制。要在 SageMaker 中禁用检查点，请使用以下示例显式上传检查点。

如果遇到前面的错误，请不要使用`checkpoint_s3_uri`使用 SageMaker`estimator`调用。在为大型模型保存检查点时，我们建议将检查点保存到自定义目录中，然后将检查点传递给助手函数（作为`local_path`参数）。

```
import os

def aws_s3_sync(source, destination):
    """aws s3 sync in quiet mode and time profile"""
    import time, subprocess
    cmd = ["aws", "s3", "sync", "--quiet", source, destination]
    print(f"Syncing files from {source} to {destination}")
    start_time = time.time()
    p = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    p.wait()
    end_time = time.time()
    print("Time Taken to Sync: ", (end_time-start_time))
    return

def sync_local_checkpoints_to_s3(local_path="/opt/ml/checkpoints",
 s3_uri=os.path.dirname(os.path.dirname(os.getenv('SM_MODULE_DIR', '')))+'/checkpoints'):
    """ sample function to sync checkpoints from local path to s3 """

    import boto3
    #check if local path exists
    if not os.path.exists(local_path):
        raise RuntimeError("Provided local path {local_path} does not exist. Please check")

    #check if s3 bucket exists
    s3 = boto3.resource('s3')
    if not s3_uri.startswith("s3://"):
```

```
        raise ValueError(f"Provided s3 uri {s3_uri} is not valid.")

    s3_bucket = s3_uri.replace('s3://','').split('/')[0]
    print(f"S3 Bucket: {s3_bucket}")
    try:
        s3.meta.client.head_bucket(Bucket=s3_bucket)
    except Exception as e:
        raise e
    aws_s3_sync(local_path, s3_uri)
    return

def sync_s3_checkpoints_to_local(local_path="/opt/ml/checkpoints",
 s3_uri=os.path.dirname(os.path.dirname(os.getenv('SM_MODULE_DIR', '')))+'/checkpoints'):
    """ sample function to sync checkpoints from s3 to local path """

    import boto3
    #try to create local path if it does not exist
    if not os.path.exists(local_path):
        print(f"Provided local path {local_path} does not exist. Creating...")
        try:
            os.makedirs(local_path)
        except Exception as e:
            raise RuntimeError(f"Failed to create {local_path}")

    #check if s3 bucket exists
    s3 = boto3.resource('s3')
    if not s3_uri.startswith("s3://"):
        raise ValueError(f"Provided s3 uri {s3_uri} is not valid.")

    s3_bucket = s3_uri.replace('s3://','').split('/')[0]
    print(f"S3 Bucket: {s3_bucket}")
    try:
        s3.meta.client.head_bucket(Bucket=s3_bucket)
    except Exception as e:
        raise e
    aws_s3_sync(s3_uri, local_path)
    return
```

帮助函数的用法：

```
#base_s3_uri - user input s3 uri or save to model directory (default)
#curr_host - to save checkpoints of current host
#iteration - current step/epoch during which checkpoint is saved

# save checkpoints on every node using local_rank
if smp.local_rank() == 0:
    base_s3_uri = os.path.dirname(os.path.dirname(os.getenv('SM_MODULE_DIR', '')))
    curr_host = os.environ['SM_CURRENT_HOST']
    full_s3_uri = f'{base_s3_uri}/checkpoints/{curr_host}/{iteration}'
    sync_local_checkpoints_to_s3(local_path=checkpoint_dir, s3_uri=full_s3_uri)
```

使用模型并行和 TensorFlow 进行收敛

当您将 SageMaker 多节点训练与 TensorFlow 和分布式模型并行使用时，由于每个节点上的训练输入文件的顺序可能不同，损失可能不如预期收敛。这可能会导致同一模型并行组中的不同排名处理不同的输入文件，从而导致不一致。为防止这种情况，请确保输入文件在转换为 TensorFlow 数据集之前在所有排列中都以相同的方式排序。实现这一目标的一种方法是对训练脚本中的输入文件名进行排序。

停顿或崩溃分布式培训工作

如果您的培训工作存在停滞、崩溃或没有响应问题，请阅读以下故障排除项目以确定问题的原因。如果您需要任何进一步的支持，请通过联系 SageMaker 分布式培训团队Amazon支持.

- 如果你看到分布式培训作业在 NCCL 初始化步骤停滞，请考虑以下事项：
  - 如果您使用的是其中一个启用了 EFA 的实例（`ml.p4d`要么`ml.p3dn`实例）具有自定义 VPC 及其子网，请确保所使用的安全组具有往返于同一 SG 的所有端口的入站和出站连接。作为一项单独的规则（对于互联网访问），您通常还需要与任何 IP 的出站连接。要查找有关如何为 EFA 通信添加入站和出站规则的说明，请参阅SageMaker初始化期间分布式培训 Job 停滞 (p. 1997).
- 如果你看到检查点时分布式培训作业停滞完整的模型，这可能是因为`state_dict()`并非所有级别对模型或优化器进行调用`rdp_rank()==0`（使用张量并行时）或`dp_rank()==0`（当仅使用管道并行时）。这些队伍需要进行沟通才能构建要保存的检查点。当检查点部分优化程序时，也可能会发生类似的停滞问题`shard_optimizer_state`已启用。

  有关检查点具有模型并行性的模型的更多信息，请参阅保存和加载的一般说明和使用 Tensor 并行度进行检查点的说明 (p. 2028).
- 如果培训工作因内存不足错误，这意味着需要调整分布式训练配置以适应 GPU 群集上的模型。有关更多信息和最佳实践，请参阅为给定模型设置正确的配置 (p. 2036).
- 如果培训工作因不可纠正ECC 错误，这意味着群集中的一个 GPU 已经坏了。如果你需要技术支持，请与Amazon如果可能的话，然后从检查点重新启动你的训练作业。
- 在极少数情况下，以前工作但接近 GPU 内存限制的作业配置可能会在以后因为内存不足错误. 这可能是因为由于 ECC 错误，某些 GPU 的可用内存低于平时。
- 网络超时崩溃当运行不使用节点中所有 GPU 的多节点作业时，可能会发生这种情况。要解决这个问题，请确保在节点上使用所有 GPU`processes_per_host`参数设置为每个实例中 GPU 的数量。例如，这是`processes_per_host=8`为了`ml.p3.16xlarge`、`ml.p3dn.24xlarge`, 和`ml.p4d.24xlarge`实例。
- 如果您发现您的培训工作在数据下载阶段需要很长时间，请确保向其提供的 Amazon S3 路径`checkpoint_s3_uri`对于 SageMaker`Estimator`课程对于当前的培训工作来说是唯一的。如果在同时运行的多个训练作业中重复使用此路径，则所有这些检查点都将上传并下载到同一个 Amazon S3 路径，这可能会显著增加检查点加载时间。
- 处理大型数据和模型时，使用 FSx for Lustre。
  - 如果您的数据集很大且需要很长时间才能获取，我们建议将数据集保存在FSx for Lustre.
  - 当训练模型超过 100 亿个参数时，我们建议使用 FSx for Lustre 进行检查点。
  - 创建文件系统后，请确保等待状态变为可用然后才开始使用它的培训工作。

NCCL 错误

如果遇到以下错误，可能是由于进程耗尽 GPU 内存造成的。

```
NCCL error in: ../torch/lib/c10d/ProcessGroupNCCL.cpp:825, unhandled system error, NCCL
 version 2.7.8
ncclSystemError: System call (socket, malloc, munmap, etc) failed.
```

你可以通过减少批量大小或解决这个问题`active_microbatches`. 如果自动分区不能导致均衡的分区，则可能需要考虑手动分区。有关更多信息，请参阅跨节点的管道并行性 (p. 2036)。

亚马逊 SageMaker 分布式训练记本示例

以下案例研究和笔记本提供了为受支持的深度学习框架（PyTorch、TensorFlow 和 HuggingFace）实施 SageMaker 分布式培训库的示例，例如用于视觉的 CNN 和 MaskrCNN 以及用于自然语言处理的 BERT。

这些笔记本在SageMaker 示例 GitHub 知识库. 你也可以在SageMaker 示例网站.

这些示例设置为使用`p3.16xlarge`工作节点的实例，但你可以选择`ml.p3dn.24xlarge`要么`ml.p4d.24xlarge`实例类型， SageMaker 分布式培训库进行了优化。您可以使用单个节点的集群来测试笔记本电脑；但是，要查看性能改进，如训练基准部分中，使用多个节点（两个或更多）组成的群集。这些示例说明了您在其中修改此配置的部分。

博客和案例研究

以下博客讨论了有关使用 SageMaker 分布式培训库。

SageMaker 分布式数据并行

- 现代汽车使用 Amazon SageMaker 缩短自动驾驶模型的 ML 模型训练时间
- 分布式训练：使用变形金刚和 Amazon SageMaker 训练 BART/T5 进行总结中的HuggingFace 文档
- 在训练 yoLov5Amazon和 PyTorch 和 SageMaker 分布式数据 parallel 库在中等
- 加快 EfficientNet 模型训练 SageMaker 和 PyTorch 和 SageMaker 分布式数据 parallel 库在中等
- 加快 EfficientNet 训练Amazon使用 SageMaker 分布式数据 parallel 库在向数据科学

PyTorch 示例

SageMaker 分布式数据并行

- 有线电视新闻网 PyTorch 1.6 和 SageMaker 并行数据
- maskrCNN PyTorch 1.6 和 SageMaker 并行数据
- BERT 与 PyTorch 1.6 和 SageMaker 并行数据

SageMaker 分布式模型并行

- 使用训练 GPT-2 PyTorch 使用 1.8.1 和 Tensor 并行度 SageMaker 模型并行度库
- BERT 与 PyTorch 1.6 和 SageMaker 并行模型

TensorFlow 示例

SageMaker 分布式数据并行

- 有线电视新闻网 TensorFlow 2.3.1 和 SageMaker 并行数据
- maskrCNN TensorFlow 2.3.1 和 SageMaker 并行数据
- BERT 与 TensorFlow 2.3.1 和 SageMaker 并行数据

SageMaker 分布式模型并行

- 有线电视新闻网 TensorFlow 2.3.1 和 SageMaker 并行模型

HuggingFace 示例

以下 HuggingFace 上 SageMaker 示例可在HuggingFace 笔记本存储库.

SageMaker 分布式数据并行

- HuggingFace 分布式数据并行培训 PyTorch 上 SageMaker -分布式问题回答
- HuggingFace 分布式数据并行培训 PyTorch 上 SageMaker -分布式文本总结
- HuggingFace 分布式数据并行培训 TensorFlow SageMaker

SageMaker 分布式模型并行

- HuggingFace TensorFlow SageMaker 上的分布式模型并行训

如何访问或下载 SageMaker 分布式训练记本示例

按照说明访问或下载 SageMaker 分布式训练示例

选项 1：使用 SageMaker 笔记本实例

要使用上述示例，我们建议您使用亚马逊 SageMaker 笔记本实例。笔记本实例运行 Jupyter 笔记本并且 JupyterServer Amazon EC2 实例上的应用程序，针对机器学习进行了优化。如果您没有有效的笔记本实例，请按照中的说明操作创建笔记本实例 (p. 149)中的 SageMaker 开发人员指南来创建一个。

创建实例之后，在笔记本实例的页面 SageMaker 控制台，执行以下操作：

1. 打开JupyterLab.
2. 选择示例图标 ( ) 在左边的托盘中。
3. 浏览示例了解训练然后寻找标题为的笔记本并行数据要么并行分布式模型.

选项 2：克隆 SageMaker 示例存储库 SageMaker 工作室或笔记本实例

要下载并使用上述示例笔记本，请执行以下操作来克隆该示例 GitHub 存储库：

1. 打开终端。
2. 在命令行中，导航到 SageMaker folder。

```
cd SageMaker
```

3. 克隆SageMaker 示例 GitHub 知识库.

```
git clone https://github.com/aws/amazon-sagemaker-examples.git
```

Note

下载HuggingFace 示例 (p. 2044)，克隆HuggingFace 笔记本 GitHub 知识库：

```
git clone https://github.com/huggingface/notebooks huggingface-notebooks
```

4. 在 JupyterLab 界面，导航到`amazon-sagemaker-examples`folder。
5. 在`training/distributed_training`文件夹中，有框架的文件夹，每个文件夹中都有用于`data_parallel`和`model_parallel`. 选择您选择的示例，然后按照说明启动分布式培训，使用 SageMaker 分布式培训库。

亚马逊 SageMaker 澄清偏差检测和模型可解释性

本主题介绍如何配置亚马逊。 SageMaker 澄清能够计算偏见指标和功能归因的处理作业以获得可解释性。它是使用专门的 SageMaker 澄清容器映像。提供了有关如何查找和下载其中一个容器映像的说明。简要概述如何 SageMaker 草绘了澄清作品。本文介绍了配置处理作业所需的参数和分析类型。概述了先决条件以及有关消耗的计算资源的建议 SageMaker 提供了澄清处理作业。

示例笔记本

亚马逊 SageMaker 澄清为训练后偏差检测和模型可解释性提供了以下示例笔记本：

- 亚马逊 SageMaker 澄清处理使用 — 使用 SageMaker 澄清以创建处理作业，以检测偏差并使用功能归因解释模型预测。示例包括使用 CSV 和JSON 行数据格式、自带容器以及使用 Spark 运行处理作业。

这款笔记本已经验证可在亚马逊上运行 SageMaker 仅限 Studio。如果您需要有关如何在亚马逊打开笔记本电脑的说明 SageMaker 工作室，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86). 如果系统会提示您选择内核，请选择Python 3（数据科学）.

主题

- 先决条件 (p. 2046)
- 怎么样 SageMaker 澄清处理作业工作 (p. 2046)
- 开始使用 SageMaker 澄清容器 (p. 2047)
- 配置 SageMaker 澄清处理 Job 容器的输入和输出参数 (p. 2048)
- 配置分析 (p. 2049)
- 运行 SageMaker 澄清处理工作以进行偏见分析和可解释性 (p. 2055)
- 使用亚马逊检测训练后数据和模型偏差 SageMaker 澄清 (p. 2058)
- 亚马逊 SageMaker 澄清模型的可解释性 (p. 2072)
- 故障排除 SageMaker 澄清处理任务 (p. 2079)

先决条件

在开始之前，您需要满足以下先决条件：

- 您需要在 CSV 中以表格文件的形式提供输入数据集或JSON 行格式的日期和时间。输入数据集必须包含用于偏差分析的标签列。数据集应为机器学习做好准备，并且任何需要的预处理（例如数据清理或要素工程）已经完成。
- 您需要提供一个支持 CSV 或 JSON Line 文件格式的模型工件作为其内容类型输入之一。为了获得训练后的偏差指标和可解释性，我们使用数据集来推断模型工件。减去标签列的每一行都必须准备好用作推断的负载。
- 使用创建处理作业时 SageMaker 容器镜像，你需要以下内容：
  - 必须为处理作业禁用网络隔离。
  - 如果模型在 VPC 中，则处理作业必须与模型在同一 VPC 中。
  - 调用者的IAM 用户/角色必须有权限 SageMaker API。建议使用 `"arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"` 托管式策略。

怎么样 SageMaker 澄清处理作业工作

一个 SageMaker 处理作业使用 SageMaker 在机器学习工作流程生命周期的几个阶段澄清容器。您可以使用 SageMaker 使用数据集和模型澄清容器以计算以下类型的分析：

- 预培训偏差指标
- 培训后偏差指标
- SHAP 值的可解释性
- 部分依赖图（PDP）

在配置处理作业时，您可以控制计算哪些分析。对于预训练偏差指标，您需要提供数据集。通过提供数据集和模型名称，您可以在模型经过训练后计算训练后偏差指标和可解释性。您必须以 JSON 配置文件的形式配置必要的参数，并将其作为处理作业的输入提供。

处理作业完成后，分析结果将保存在ProcessingOutput处理作业的参数。然后，您可以从那里下载并查看输出结果，也可以在 Studio 中查看结果（如果你在那里运行了笔记本）。

为了计算训练后偏差指标和 SHAP 值，计算需要获得对所提供的模型名称的推断。为了实现这一目标，处理作业会创建一个具有模型名称的临时端点，称为影子端节点. 计算完成后，处理作业将删除影子端点。

在较高层次上，处理作业完成以下步骤：

1. 验证输入和参数。
2. 创建影子端点。
3. 计算训练前偏差指标。
4. 计算训练后偏差指标。
5. 计算本地和全局功能属性。
6. 删除影子终端节点。
7. 生成输出文件。

开始使用 SageMaker 澄清容器

亚马逊 SageMaker 提供预构建 SageMaker 澄清容器映像，其中包括计算偏差指标和功能归因所需的库和其他依赖关系以确保可解释性。此映像已启用运行 SageMaker 处理数据 (p. 728)在您的账户中。

容器的图像 URI 采用以下形式：

```
<ACCOUNT_ID>.dkr.ecr.<REGION_NAME>.amazonaws.com/sagemaker-clarify-processing:1.0
```

例如：

```
205585389593.dkr.ecr.us-east-1.amazonaws.com/sagemaker-clarify-processing:1.0
```

下表按以下方式列出了地址：Amazon区域。

用于澄清处理作业的 Docker 镜像

区域	映像地址
us-east-1	205585389593.dkr.ecr.us-east-1.amazonaws.com/sagemaker 澄清处理：1.0
us-east-2	211330385671.dkr.ecr.us-east-2.amazonaws.com/sagemaker 澄清处理：1.0
us-west-1	740489534195.dkr.ecr.us-west-1.amazonaws.com/sagemaker 澄清处理：1.0
us-west-2	306415355426.dkr.ecr.us-west-2.amazonaws.com/sagemaker 澄清处理：1.0
ap-east-1	0987607982.dkr.ecr.ap-east-1.amazonaws.com/sagemaker 澄清处理：1.0
ap-south-1	452307495513.dkr.ecr.ap-南1.amazonaws.com/sagemaker 澄清处理：1.0
ap-northeast-2	263625296855.dkr.ecr.ap-东北-2.amazonaws.com/sagemaker 澄清处理：1.0
ap-southeast-1	834264404009.dkr.ecr.ap-东南-1.amazonaws.com/sagemaker 澄清处理：1.0
ap-southeast-2	007051062584.dkr.ecr.ap-东南-2.amazonaws.com/sagemaker 澄清处理：1.0
ap-northeast-1	377024640650.dkr.ecr.ap-东北-1.amazonaws.com/sagemaker 澄清处理：1.0

区域	映像地址
ca-central-1	675030665977.dkr.ecr.ca-中央 -1.amazonaws.com/sagemaker 澄清处理：1.0
eu-central-1	017069133835.dkr.ecr.eu-中央 -1.amazonaws.com/sagemaker 澄清处理：1.0
eu-west-1	131013547314.dkr.ecr.eu-west-1.amazonaws.com/sagemaker 澄清处理：1.0
eu-west-2	440796970383.dkr.ecr.eu-west-2.amazonaws.com/sagemaker 澄清处理：1.0
eu-west-3	341593696636.dkr.ecr.eu-west-3.amazonaws.com/sagemaker 澄清处理：1.0
eu-north-1	763603941244.dkr.ecr.eu-北-1.amazonaws.com/sagemaker 澄清处理：1.0
me-south-1	835444307964.dkr.ecr.me-南1.amazonaws.com/sagemaker 澄清处理：1.0
sa-east-1	520018980103.dkr.ecr.sa-east-1.amazonaws.com/sagemaker 澄清处理：1.0
af-south-1	811711786498.dkr.ecr.af-南1.amazonaws.com/sagemaker 澄清处理：1.0
eu-south-1	638885417683.dkr.ecr.eu-南1.amazonaws.com/sagemaker 澄清处理：1.0
cn-north-1	122526803553.dkr.ecr.cn-北-1.amazonaws.com.cn/sagemaker 澄清处理：1.0
cn-northwest-1	122578899357.dkr.ecr.cn-西北 -1.amazonaws.com.cn/sagemaker 澄清处理：1.0

配置 SageMaker 澄清处理 Job 容器的输入和输出参数

“处理 Job” 要求您指定以下输入参数：具有输入名称的数据集文件`"dataset"`作为 Amazon S3 对象或前缀，以及带有输入名称的分析配置文件`"analysis_config"`作为 Amazon S3 对象。该作业还需要输出参数：作为 Amazon S3 前缀的输出位置。

您可以使用创建并运行处理作业。 SageMaker `CreateProcessingJob`使用 APIAmazonSDK 或 CLI 或开发工具包.

使用 SageMaker Python SDK，创建一个`Processor`使用 SageMaker 澄清容器映像 URI：

```
from sagemaker import clarify
clarify_processor = clarify.SageMakerClarifyProcessor(role=role,
    instance_count=1,
    instance_type='ml.c5.xlarge',
    max_runtime_in_seconds=1200,
    volume_size_in_gb=100)
```

完成创建澄清处理器后，需要为处理器设置输入和输出对象。

> **Note**
>
> 如果你提供`"dataset_uri"`通过 “分析 _config.json”（请参阅以下主题配置分析 (p. 2049)），您无需创建`dataset_input`对象。

```
    dataset_path = "s3://my_bucket/my_folder/train.csv"
    analysis_config_path = "s3://my_bucket/my_folder/analysis_config.json"
    analysis_result_path = "s3://my_bucket/my_folder/output"

    analysis_config_input = ProcessingInput(
        input_name="analysis_config",
        source=analysis_config_path,
```

```
        destination="/opt/ml/processing/input/config",
        s3_data_type="S3Prefix",
        s3_input_mode="File",
        s3_compression_type="None")
    dataset_input = ProcessingInput(
        input_name="dataset",
        source=dataset_path,
        destination="/opt/ml/processing/input/data",
        s3_data_type="S3Prefix",
        s3_input_mode="File",
        s3_compression_type="None")
    analysis_result_output = ProcessingOutput(
        source="/opt/ml/processing/output",
        destination=analysis_result_path,
        output_name="analysis_result",
        s3_upload_mode="EndOfJob")
```

配置分析

分析的输入是由ProcessingInputAPI。这些区域有："analysis_config"的值input_name指定名为的 JSON 文件分析 _config.json包含配置值。JSON 文件的路径在source的参数ProcessingInput. 在参数说明之后，为 CSV、JSON 行、图像和文本数据集的 JSON 文件提供了分析配置 JSON 文件的示例。

主题

- JSON 配置文件的参数 (p. 2049)
- 示例 JSON 配置文件 (p. 2052)

JSON 配置文件的参数

在 JSON 配置文件中，可以指定以下参数。

- **"version"**— （可选）配置文件的架构版本。如果未提供，则使用最新支持的版本。
- **"dataset_type"**— （必填）数据集的格式。有效值为"text/csv"对于 CSV，"application/jsonlines"对于 JSON 行，application/x-parquet对于 Apache 实木复合地板，以及application/x-image以实现计算机视觉的可解释性。有关更多信息，请参阅用于推理的常见数据格式.
- **"dataset_uri"**— （可选）数据集 S3 前缀/对象URI（如果没有给定为ProcessingInput)。如果是前缀，则处理作业会递归收集前缀下的所有 S3 文件。对于计算机视觉，URI 是必需的，它可以是图像清单文件的路径，也可以是要解释的图像存储桶的路径。
- **"headers"**— （可选）数据集中的列名列表。如果dataset_type是"application/jsonlines"和"label"被指定，那么最后一个标题应为标签列的标题。
- **"label"**— （可选）要用于的模型的目标属性偏差指标. 指定为列名称；如果数据集格式为 CSV，则指定为索引；如果数据集格式为 JSON 线，则指定为 JSONPath。
- **"probability_threshold"**— （可选）一个浮点值，用于指示在二进制分类情况下选择二进制标签的阈值。此参数用于对象检测，用于过滤掉被置信度低于probability_threshold值。默认值为 0.5。
- **"features"**— （可选）如果数据集格式为 JSON 线，则 JsonPath 用于查找偏差指标的要素列。
- **"label_values_or_threshold"**— （可选）标签值或阈值列表。表示用于偏见指标的积极结果。
- **"facet"**— （可选）敏感属性的要素列表，称为小平面。刻面用于偏差指标以货币对的形式，包括以下内容：
  - "name_or_index"— Facet 列名称或索引。
  - "value_or_threshold"— （可选）小平面列可以采用的值或阈值的列表。表示敏感组，例如用于衡量偏差的组。如果未提供，则偏差指标将作为每个唯一值（而不是所有值）的一组计算。如果小平面列为数字，则应用此阈值作为选择敏感组的下限。

- **"group_variable"**— （可选）一个列名或索引，用于指示要用于偏差指标 有条件的人口差异。
- **"joinsource_name_or_index"**— （可选）数据集中用作标识符列的列的名称或索引，例如，在执行连接时。此列仅用作标识符，不用于任何其他计算。除了数据集包含多个文件之外，除非在所有情况下，这都是可选字段。"save_local_shap_values"设置为true.
- **"methods"**— 分析和报告的方法及其参数的列表。如果省略了任何部分，则不会计算它。
  - "pre_training_bias"— （可选）关于培训前偏差指标的章节。
    - "methods"— 要计算的预训练指标列表。
  - "post_training_bias"— （可选）关于训练后偏差指标的部分。
    - "methods"— 要计算的训练后指标的列表。
  - "shap"— （可选）关于 SHAP 值计算的部分。
    - "agg_method"— 全局 SHAP 值的聚合方法。有效值如下所示：
      - "mean_abs"— 所有实例的绝对 SHAP 值的均值。
      - "median"— 所有实例的 SHAP 值的中位数。
      - "mean_sq"— 所有实例的 SHAP 平方值的均值。
    - "baseline"— （可选）行列表（至少一个）或 S3 对象 URI。用作内核 SHAP 算法中的基线数据集（也称为背景数据集）。格式应与数据集格式相同。每行应只包含特征列（或值），并忽略在发送到模型之前必须排除的任何列。这包括labelColumnjoinsource列和包含在"exclude_column"字段中返回的子位置类型。
      - 对于计算机视觉，指向基线图像的路径，用于掩盖输入图像中的要素。默认值：RGB 噪声掩码。
      - 对于文本列的自然语言处理 (NLP)，基线值应该是用于替换指定的文本单位granularity中的"text_config". 文本单位的有效值为token、sentence, 和paragraph.
    - "num_clusters"— （可选）如果"baseline"未提供，Clarify 尝试通过对数据集进行聚类来计算基线。这些区域有："num_clusters"确定数据集聚集到的集群的数量。每个集群都有助于一个基线，因此群集的数量直接影响 SHAP 解释的运行时间。
    - "num_samples"— 内核 SHAP 算法中要使用的样本数量。此数字决定了用于计算 SHAP 值的生成合成数据集的大小。
    - "seed"— （可选）随机数生成器的种子值，以获得确定性 SHAP 结果。
    - "text_config"— （必需）指定自然语言处理 (NLP) 功能的配置。如果提供此配置，则将文本功能视为文本，并为单个文本单元提供说明。文本单位由"granularity".
      - "granularity"— （必需）对于 NLP，确定文本要素分析的粒度单位。有效值为"token"、"sentence" 或 "paragraph"。这些单位中的每个单位都被视为要素，并为指定的要素计算 Shap 值。
      - "language"— （必填）指定用于自然语言处理 (NLP) 的文本要素的语言。有效值为"chinese"、"danish"、"dutch"、"english"、"french"、"german"、"greek"、"italian"、"ja language"、"norwegian bokmål"、"polish"、"portuguese"、"romanian"、"russian"、"spanish"、"afrikaans"、"alba使用"multi-language" 混合使用多种语言。
    - "use_logit"— （可选）布尔值，用于指示是否将 logit 函数应用于模型预测。如果"use_logit"是true，那么 SHAP 值具有对数赔率单位。原定设置值为 false。
    - "image_config"— （可选）用于配置 SHAP 配置参数以实现计算机视觉可解释性的部分。
      - "model_type"— 物体检测模型或图像分类模型。
      - "num_segments"— （可选）确定输入图像中要标记的片段的大致数量。默认值为 20。
      - "segment_compactness"— （可选）确定由 SKLearn 的 slic 方法生成的图像片段的形状和大小。有关更多详细信息，请参阅cikit-image文档中）。默认 为 5。
      - "max_objects"— （可选）用于按最高置信度分数过滤计算机视觉模型检测到的对象。默认值为 3。
      - "iou_threshold"— （可选）用于根据原始检测值评估预测值的最小交集 (IOU) 量度。使用是因为在掩蔽期间检测框将移动。默认值为 0.5。
      - "context"— （可选）在运行 SHAP 时掩盖检测到对象的边界框周围的区域。有效值为0掩盖所有东西，或者 1 来掩盖任何东西。默认值是 1。

    - "save_local_shap_values"— (可选) 布尔值，用于指示是否将本地 SHAP 值保存在输出位置。使用true拯救他们;false不救他们。默认为 false。
  - "pdp"— (可选) 用于配置部分依赖性图 (PDP) 的部分。显示目标响应对一组感兴趣的输入要素的依赖关系。边缘化所有其他输入要素的值。
    - "features": ["pdp_feature_1", "pdp_feature_2"…] — (可选) 要计算和绘制部分依赖性图的要素名称或索引的列表。如果没有请求 SHAP，则必须提供这些功能。
    - "grid_resolution" — (必填) 用于数字要素。表示将数值范围划分到的存储桶数量。这指定 PDP 图的网格粒度。
    - "top_k_features" — (可选) 如果features未提供参数，而且shap已提供，澄清选择顶部k基于 SHAP 归因的功能。您可以设置此值以指定 PDP 图必须使用多少顶级要素。默认值为 10。
- "**predictor**"— (可选) 关于模型参数的部分，如果需要"shap"和"post_training_bias"存在部分。
  - "model_name"— 由创建的模型名称CreateModelAPI，容器模式为SingleModel.
  - "instance_type"— 影子终端节点的实例类型。
  - "initial_instance_count"— 影子终端节点的实例计数。
  - "content_type"— (可选) 用于获取影子端点推断的模型输入格式。有效值为"text/csv"对于 CSV，"application/jsonlines"对于 JSON 行，application/x-parquet对于 Apache 实木复合地板，以及application/x-image以启用计算机视觉的可解释性。默认值与dataset_type格式的日期和时间。
  - "accept_type"— (可选) 模型输出用于获取影子端点推断的格式。有效值为"text/csv"对于 CSV，"application/jsonlines"对于 JSON 行，application/x-parquet对于 Apache 实木复合地板，以及application/x-image以启用计算机视觉的可解释性。默认值与content_type.
  - "accelerator_type"— (可选) 要附加到实例的 Elastic Inference (EI) 加速器类型。例如：ml.eia1.medium (有关更多信息，请参阅 使用 Amazon SageMaker Elastic Inference (EI) (p. 2387)。)
  - "custom_attributes"— (可选) 提供有关向亚马逊托管模型提交的推理请求的其他信息 SageMaker 终端节点。该信息是逐字转发的不透明值。例如，您可以使用此值提供跟踪请求的 ID。您还可以使用此值提供服务端点编程处理的元数据。该值必须包含不超过 1024 个可见的 US-ASCII 字符，如第 3.3.6 节。字段值组件超文本传输协议 (HTTP/1.1)。
  - "label"— (可选) 偏差指标使用的目标属性的模型输出中的索引或 jsonPath 位置。如果labelCSV 中没有提供accept_typecase，然后澄清假定模型输出是与分数或概率对应的单个数值。
  - "probability"— (可选) 模型输出中的索引或 jsonPath 位置，用于解释性的概率或分数。例如，如果模型输出是带有标签和概率列表的 JSON Line，则会选择与最大概率对应的标签进行偏差计算。对于可解释性方法，目前对所有概率进行了解释。
  - "endpoint_name-prefix"— (可选) 为临时终端节点的名称提供自定义前缀。
  - "endpoint_name"— (可选) 指定用于获取预测的现有终端节点，使用现有终端节点可缩短引导时间，但可能会导致现有终端节点的负载显著增加。指定现有生产终端节点时要谨慎。
  - "target_model"— (可选) 在使用多模型端点时设置目标模型名称。有关多模型终端节点的更多信息，请参阅Request Syntax.
  - "label_headers"— (可选) "label"获取数据集。将模型端点返回的分数与相应的标签值相关联。用于提取分数最高作为预测标签的标签值。
  - "content_template"— (可选) 用于从数据集实例构造模型输入的模板字符串。它只在以下情况下使用"content_type"是"application/jsonlines". 模板应仅有一个占位符，$features，它在运行时由功能列表替换。例如，给定"content_template":"{\"myfeatures\":$features}"，如果实例 (没有标签) 1,2,3，然后模型输入变为 JSON Line'{"myfeatures":[1,2,3]}'.
- "**report**"— (可选) 关于报告参数的部分。如果存在此部分，则会生成报告。
  - "name"— (可选) 报表笔记本和 PDF 文件的文件名前缀。默认名称为 "report"。
  - "title"— (可选) 报表笔记本和 PDF 文件的标题字符串。默认标题是"SageMaker Analysis Report".

示例 JSON 配置文件

以下是 CSV、JSON 行、图像和文本数据集的分析配置 JSON 文件的示例。

主题

- CSV 数据集的分析配置 JSON 文件 (p. 2052)
- JSON 线数据集的分析配置 JSON 文件 (p. 2053)
- 影像数据集的分析配置 JSON 文件 (p. 2054)
- 文本数据集的 NLP 分析配置 JSON 文件 (p. 2055)

CSV 数据集的分析配置 JSON 文件

以下代码示例显示如何为 CSV 数据集配置分析。

```
{
    "dataset_type": "text/csv",
    "headers": [
        "feature_0",
        "feature_1",
        "feature_2",
        "feature_3",
        "target"
    ],
    "label": "target",
    "label_values_or_threshold": [1],
    "probability_threshold" : 0.7,
    "facet": [
        {
         "name_or_index" : "feature_1",
         "value_or_threshold": [1]
        },
        {
         "name_or_index" : "feature_2",
         "value_or_threshold": [0.7]
        }
    ],
    "group_variable": "feature_3",
    "joinsource_name_or_index":'column_name',
    "methods": {
        "shap": {
            "baseline": [
                [
                    "yes",
                    3,
                    0.9,
                    1
                ]
            ],
            "num_samples": 1000,
            "agg_method": "mean",
            "use_logit": true,
            "save_local_shap_values": true,
            "num_clusters": 5,
            "seed": 1234
        },
        "pre_training_bias": {
            "methods": "all"
        },
        "post_training_bias": {
            "methods": "all"
```

```
        },
        "report": {
            "name": "report",
            "title": "Analysis Report"
        }
    },
    "predictor": {
        "model_name": "my_model",
        "instance_type": "ml.m5.xlarge",
        "initial_instance_count": 1,
        "content_type": "text/csv",
        "accept_type": "text/csv",
        "accelerator_type":"ml.eia1.medium",
        "custom_attributes": "c000b4f9-df62-4c85-a0bf-7c525f9104a4",
        "label": 0,
        "probability": 1,
        "endpoint_name_prefix": "myendpointprefix",
        "target_model": ""
    }
}
```

CSV 格式的模型输出如下：

```
Current,"[0.028986845165491104, 0.8253824710845947, 0.028993206098675728,
 0.02898673340678215, 0.029557107016444206, 0.0290389321744442, 0.02905467338860035]"
```

相应的预测变量配置如下：

```
    "predictor": {
        ...,
        "accept_type":  "text/csv",
        "label": 0,
        "probability": 1,
        ...
    }
```

JSON 线数据集的分析配置 JSON 文件

以下代码示例演示了如何为 JSON Line 数据集配置分析。

```
{
    "dataset_type": "application/jsonlines",
    "dataset_uri": "s3://my_bucket/my_folder/dataset.jsonl",
    "headers": ["Label", "Feature1", "Feature2"],
    "label": "data.label",
    "features": "data.features.values",
    "facet":[
        {
            "name_or_index": "Feature1",
            "value_or_threshold": [1,5]
        },
        {
            "name_or_index": "Feature2",
            "value_or_threshold": [2,6]
        }
    ],
    "methods": {
        "shap": {
            "baseline": [
                {"data":{"features":{"values":[9,10]},"label":0}},
                {"data":{"features":{"values":[11,12]},"label":1}}
```

```
            ]
        }
    },
    "predictor": {
        "model_name": "my_jsonl_model",
        "instance_type": "ml.m5.xlarge",
        "initial_instance_count": 1
    }
}
```

作为 S3 前缀的数据集如下：

```
"dataset_uri": "s3://my_bucket/my_folder"
```

作为 S3 对象的数据集如下：

```
"dataset_uri": "s3://my_bucket/my_folder/train.csv"
```

作为 S3 对象的基线如下：

```
"baseline": "s3://my_bucket/my_folder/baseline.csv"
```

以 JSON 行形式输出模型如下：

```
{"predicted_label": "Current", "score": "[0.028986845165491104, 0.8253824710845947,
 0.028993206098675728, 0.02898673340678215, 0.029557107016444206, 0.0290389321744442,
 0.02905467338860035]"}
```

相应的预测变量配置如下：

```
    "predictor": {
        ...,
        "accept_type":  "application/jsonlines",
        "label": "predicted_label",
        "probability": "score",
        ...
    }
```

影像数据集的分析配置 JSON 文件

以下代码示例演示了如何使用计算机视觉为对象检测配置分析。

```
"dataset_type": "application/x-image",
"dataset_uri": "s3://<BUCKET/KEY">
"probability_threshold": 0.7,
"methods": {
    "shap": {
        "num_samples": 500,
        "baseline": "s3://path/to/baseline/image/noise_rgb.png",
        "image_config": {
            "model_type": "(OBJECT_DETECTION|IMAGE_CLASSIFICATION)",
            "num_segments": 20,
            "segment_compactness": (5|10|100),
            "max_objects" : 3,
            "overlap": 0.5,
            "context": 1.0
```

```
        }
    }
},

"predictor": {
    "endpoint_name": "sagemaker-endpoint-name",
    "content_type": "(image/jpeg | image/png | application/x-npy)",
    "label_headers": [...] # Required for CV
}
```

文本数据集的 NLP 分析配置 JSON 文件

以下代码示例演示了如何为自然配置分析。语言正在处理。

```
{
  "dataset_type":...
  .
  .
  "methods": {
      "shap" : {
         "baseline": ".."
         "num_samples": 100
         *"text_config": {
            "granularity": "(token|sentence|paragraph)"
            "language": ".."
          }*
       }
   .
   .
  }
```

运行 SageMaker 澄清处理工作以进行偏见分析和可解释性

你使用 SageMaker 澄清处理作业，以分析训练数据中的潜在偏见来源，并检查训练模型是否存在偏见。对于分析中数据的过程 SageMaker 工作室，请参阅为训练前数据中的偏差生成报告 SageMaker 工作室 (p. 551). 这里的重点是训练后的偏差指标和 SHAP 值，以确保可解释性。模型预测可能是偏见的根源（例如，如果它们作出的预测更频繁地为一个群体产生负面结果而不是另一组）。 SageMaker 澄清与 SageMaker 实验以便在模型经过训练后，您可以识别要检查偏差的属性（例如收入）。 SageMaker 运行一组算法来检查训练过的模型，并为您提供关于每个属性不同类型的偏见的直观报告，例如高收入者与低收入收入者相比是否获得更积极的预测。

主题

- 计算所需资源 SageMaker 澄清处理任务 (p. 2055)
- 运行澄清处理 Job (p. 2056)
- 使用 Spark 运行澄清处理 Job (p. 2057)
- 获取分析结果 (p. 2057)

计算所需资源 SageMaker 澄清处理任务

在确定运行所需的计算资源时，请注意以下事项： SageMaker 澄清处理任务：

- 处理作业可能需要几分钟或更长时间才能完成。
- 计算解释可能比实际推断更加耗时。这包括启动计算资源的时间。

- 计算解释可能比实际推断更密集计算。查看和监控使用 SageMaker 资源可能产生的费用。有关更多信息，请参阅 。亚马逊 SageMaker 定价.

运行澄清处理 Job

有关运行说明的示例笔记本 SageMaker 澄清 Studio 中的处理作业以检测训练后模型偏差，请参阅利用亚马逊进行可解释性和偏见检测 SageMaker 澄清.

如果您需要有关如何在亚马逊打开笔记本电脑的说明 SageMaker 工作室，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86). 以下代码示例来自之前列出的示例笔记本。

在训练模型之后，实例化 SageMaker 使用以下命令澄清处理器：

```
from sagemaker import clarify
clarify_processor = clarify.SageMakerClarifyProcessor(
    role=role,
    instance_count=1,
    instance_type='ml.c4.xlarge',
    sagemaker_session=session)
```

接下来，配置输入数据集、存储输出的位置、以DataConfig对象，请使用指定有关训练模型的信息ModelConfig，并使用提供有关预测格式的信息ModelPredictedLabelConfig.

```
bias_report_output_path = 's3://{}/{}/clarify-bias'.format(bucket, prefix)
bias_data_config = clarify.DataConfig(
    s3_data_input_path=train_uri,
    s3_output_path=bias_report_output_path,
    label='Target',
    headers=training_data.columns.to_list(),
    dataset_type='text/csv')

model_config = clarify.ModelConfig(
    model_name=model_name,
    instance_type='ml.c5.xlarge',
    instance_count=1,
    accept_type='text/csv')

predictions_config = clarify.ModelPredictedLabelConfig(probability_threshold=0.8)
```

使用BiasConfig以提供有关哪些列包含方面的信息（敏感组，Sex），什么敏感特征（facet_values_or_threshold）可能是，理想的结果是什么（label_values_or_threshold)。

```
bias_config = clarify.BiasConfig(
                label_values_or_threshold=[1],
                facet_name='Sex',
                facet_values_or_threshold=[0])
```

您可以在处理作业中同时运行训前和训练后分析run_bias().

```
clarify_processor.run_bias(
                data_config=bias_data_config,
                bias_config=bias_config,
                model_config=model_config,
                model_predicted_label_config=predictions_config,
                pre_training_methods='all',
                post_training_methods='all')
```

在 Studio 中查看结果或从bias_report_output_pathS3 存储桶。

使用 Spark 运行澄清处理 Job

Apache Spark是用于大规模数据处理的统一分析引擎。处理大型数据集时，可以使用 Spark 处理功能 SageMaker 澄清以使您的 Clarify 处理作业能够更快地运行。要对澄清任务使用 Spark 处理，请将实例计数设置为大于 1 的数字。当每个 Clarify 处理器有多个实例时，澄清使用 Spark 分布式计算。

以下示例演示了如何使用。SageMakerClarify 处理器创建一个包含 5 个实例的 Clarify 处理器。

澄清使用 Spark 分布式处理器运行与此处理器相关联的任务。

```
from sagemaker import clarify
clarify_processor = clarify.SageMakerClarifyProcessor(
    role=role,
    instance_count=5,
    instance_type='ml.c5.xlarge',
    max_runtime_in_seconds=1200,
    volume_size_in_gb=100)
```

如果您将 Clarine 作业配置为将本地 SHAP 值保存在ShapConfig类，Spark 将本地 SHAP 值 parallel 保存为多个零件文件。

如果您添加更多实例，我们建议您还增加模型配置中的实例数量ModelConfig对于影子端点。这是为了防止处理实例受到影子端点的瓶颈。具体来说，建议您使用 one-to-one 终端节点与处理实例的比率。

获取分析结果

在处理作业完成后，可以下载输出文件以在 Studio 中检查或可视化结果。输出目录包含以下文件：

- `analysis.json`— JSON 格式的偏差指标和 SHAP 值。
- `report.ipynb`— 静态笔记本可视化偏差指标和 SHAP 值。
- `explanations_shap/out.csv`— 数据集中每行的本地（每个实例）SHAAP 值，格式与输入数据集相同。在每行上，输出文件都包含每个要素的 SHAP 值和预测的标注。

在分析 JSON 文件中，偏差指标和 SHAP 值分为三个单独的部分。

```
{
    "explanations": { . . . }
    "pre_training_bias_metrics": { . . . }
    "post_training_bias_metrics": { . . . }
}
```

SHAP 值在“`explanations`”部分。值对应于每个要素列的全局 SHAP 值。

```
    "explanations": {
        "kernel_shap": {
            "label0": {
                "global_shap_values": {
                    "feature_0": 0.022486410860333206,
                    "feature_1": 0.007381025261958729,
                    "feature_2": 0.006843906804137847
                },
                "expected_value": 0.508233428001
            }
        }
    }
```

偏差指标位于训练前和训练后偏差指标部分。

```
{
    "post_training_bias_metrics": {
        "label": "target",
        "label_value_or_threshold": "1",
        "facets": {
            "feature_2": [
                {
                    "value_or_threshold": "1",
                    "metrics": [
                        {
                            "name": "DI",
                            "description": "Disparate Impact (DI)",
                            "value": 0.711340206185567
                        },
                        {
                            "name": "DCR",
                            "description": "Difference in Conditional Rejections (DCR)",
                            "value": -0.34782608695652184
                        }
                        { . . . }
                    ]
                }
            ]
        }
    }
}
```

有关偏差指标和 SHAP 值以及如何对其进行解析的更多信息，请参阅亚马逊 AI 公平性和可解释性白皮书.

报告笔记本中提供了最高 SHAP 值的条形图和带有偏差指标的表格。

使用亚马逊检测训练后数据和模型偏差 SageMaker 澄清

训练后的偏差分析可以帮助揭示可能源于数据中的偏见或分类和预测算法引入的偏见。这些分析考虑了包括标签在内的数据以及模型的预测。您可以通过分析预测标签或将预测值与数据中针对具有不同属性的组的观测目标值进行比较来评估绩效。有不同的公平概念，每个概念都需要不同的偏差指标来衡量。

有一些公平的法律概念可能不容易掌握，因为它们很难发现。例如，美国关于不同影响的概念，这种概念发生在一个被称为不太受欢迎的方面的群体时D, 即使所采取的办法似乎是公平的, 也会受到不利影响. 这种偏差可能不是由于机器学习模型造成的，但训练后偏差分析仍然可以检测到。

亚马逊 SageMaker 澄清试图确保术语的一致使用。有关术语及其定义的列表，请参阅亚马逊 SageMaker 澄清偏见和公平性条款 (p. 542).

有关训练后偏差指标的其他信息，请参阅金融 Machine Learning 的公平性措施.

示例笔记本

亚马逊 SageMaker 澄清为训练后偏差检测提供了以下示例笔记本：

- 亚马逊 SageMaker 澄清处理使用 — 使用 SageMaker 澄清以创建处理作业，以检测偏差并使用功能归因解释模型预测。示例包括使用 CSV 和 JSON Line 数据格式、自带容器以及使用 Spark 运行处理作业。

这款笔记本已经验证可在亚马逊上运行 SageMaker 仅限 Studio。如果您需要有关如何在亚马逊打开笔记本电脑的说明 SageMaker 工作室，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86). 如果系统会提示您选择内核，请选择Python 3（数据科学）.

衡量训练后数据和模型偏差

亚马逊 SageMaker 澄清提供了 11 个培训后数据和模型偏差指标，以帮助量化各种公平概念。这些概念不能同时满足，选择取决于所分析的潜在偏见的案例的具体情况。其中大多数指标是从不同人口群体的二元分类混淆矩阵中获取的数字的组合。由于公平性和偏见可以通过广泛的指标来定义，因此需要人工判断才能理解和选择哪些指标与个人使用案例相关，客户应咨询适当的利益相关者，以确定适当的应用程序的公平度量标准。

我们使用以下符号来讨论偏差指标。此处描述的概念模型用于二进制分类，其中事件被标记为样本空间中只有两种可能的结果，称为正（值为 1）和负（值为 0）。这个框架通常可以直接扩展到多类别分类，或者在需要时可扩展到涉及持续有价值结果的情况。在二进制分类案例中，正标签和负面标签将分配给原始数据集中记录的结果，以获取受欢迎的方面一个而且对于一个不喜欢的方面D. 这些标签 y 被称为观察标签将它们与预测的标签'是由机器学习模型在机器学习生命周期的训练或推断阶段分配的。这些标签用于定义概率分布 $P_a(y)$ 和 $P_d(y)$ 他们各自的方面结果。

- 标签：
  - y 表示训练数据集中事件结果的 n 个观测标签。
  - y 表示经过训练的模型对数据集中 n 个观测到的标签的预测标签。
- 结果：
  - 样本的积极结果（值为 1），例如申请接受。
    - $n^{(1)}$是观察到的积极结果（接受）标签的数量。
    - $n'^{(1)}$是积极结果（接受）的预计标签数量。
  - 样本的负面结果（值为 0），例如申请拒绝。
    - $n^{(0)}$是观察到的负面结果（拒绝）标签的数量。
    - $n'^{(0)}$是负面结果（拒绝）的预计标签数量。
- 分面值：
  - 特点一个— 定义偏见偏向的人口群体的功能值。
    - n_a是偏爱的小面值观察到的标签数量：$n_a=a^{(1)}+ n_a^{(0)}$价值方面的正面和负面观察到的标签的总和一个.
    - n'_a是偏爱的刻面值的预测标签数量：$n'_an'_a^{(1)}+ n'_a^{(0)}$面值的正面和负面预测结果标签的总和一个. $n'_a=a$.
  - 特点D— 定义偏向不利的人口群体的功能值。
    - n_d是不喜欢的小面值观察到的标签数量：$n_d=d^{(1)}+ n_d^{(0)}$小平面值观测到的正值和负面标签的总和D.
    - n'_d是不喜欢的平面值的预测标签数量：$n'_dn'_d^{(1)}+ n'_d^{(0)}$小面值的正值和负面预测标签的总和D. $n'_d=d$.
- 标记的小面数据结果的概率分布：
  - $P_a(y)$ 是刻面观测到的标签的概率分布一个. 对于二进制标记的数据，此分布由刻面样本数量的比率给出一个对总数标记为积极结果，$P_a(y)^1n) =a^{(1)}/n_a$，以及结果为负的样本数量与总数之比，$P_a(y)^0n) =a^{(0)}/n_a$.
  - $P_d(y)$ 是刻面观测到的标签的概率分布D. 对于二进制标记的数据，此分布由平面中的样本数量给出D对总数标记为积极结果，$P_d(y)^1n) =d^{(1)}/n_d$，以及结果为负的样本数量与总数之比，$P_d(y)^0n) =d^{(0)}/n_d$.

下表包含快速指导的备忘单以及训练后偏差指标的链接。

训练后偏差指标

训练后偏差指标	描述	例如问题	解释指标值
预测标签中的正比例差异 (DPPL) (p. 2063)	衡量偏爱方面之间积极预测比例的差异一个和不喜欢的方面D.	在预测的积极结果中，人口群体之间是否存在不平衡，这可能表明偏见？	标准化二进制和多类别平面标签的范围：[-1, +1] 连续标签的范围：(-∞, +∞) 解释：

训练后偏差指标	描述	例如问题	解释指标值
			• 正值表示偏爱的方面一个预计的积极结果的比例较高。 • 接近零的值表示各个方面之间的预测积极结果的比例更为相等。 • 负值表示不喜欢的方面D预计的积极结果的比例较高。
不同的影响力 (DI) (p. 2064)	衡量预测标签在受欢迎的平面中的比例一个和不喜欢的方面D.	在预测的积极结果中，人口群体之间是否存在不平衡，这可能表明偏见？	标准化二进制、多类别平面和连续标签的范围：[0, ∞) 解释： • 值小于 1 表示偏爱的方面一个预计的积极结果的比例较高。 • 值为 1 表示我们有人口统计平等。 • 值大于 1 表示不喜欢的方面D预计的积极结果的比例较高。
有条件接受的差异（DCCAC）(p. 2065)	将观察到的标签与模型预测的标签进行比较，并评估预测的积极结果（接受率）的各个方面是否相同。	与基于资格的另一个年龄组相比，一个年龄组接受贷款申请的人数是否超过预期的多或少？	二进制、多类别平面和连续标签的范围：(-∞, +∞)。 • 正值表明从不受欢迎的方面可能对合格申请人存在偏见D. • 接近零的值表明，两个方面的合格申请人也以类似的方式被接受。 • 负值表明从受欢迎的方面可能存在对合格申请人的偏见一个.

训练后偏差指标	描述	例如问题	解释指标值
有条件拒绝（DCR）的差异 (p. 2066)	将观察到的标签与模型预测的标签进行比较，并评估负面结果（拒绝）的各个方面是否相同。	与基于资格的另一个年龄组相比，一个年龄组的贷款申请被拒绝是多或少于预期的？	二进制、多类别平面和连续标签的范围：(-∞, +∞)。 • 正值表明从不受欢迎的方面可能对合格申请人存在偏见D. • 接近零的值表明，两个方面的合格申请人也以类似的方式被拒绝。 • 负值表明从受欢迎的方面可能存在对合格申请人的偏见一个.
召回差异 (RD) (p. 2067)	比较模型的回忆以获得青睐和不受欢迎的方面。	由于模型与另一个年龄组相比，一个年龄组的召回率更高，是否存在基于年龄的偏见？	二进制和多类别分类的范围：[-1, +1]。 • 积极的价值表明该模型发现了更多的真正积极因素一个而且对不受欢迎的方面有偏见D. • 接近零的值表明，该模型在两个方面都发现了大约相同数量的真正阳性，并且没有偏见。 • 负值表明该模型发现了更多的真正积极因素D而且对偏爱的方面有偏见一个.
接受率的差异 (DAR) (p. 2068)	衡量受欢迎和不受欢迎的方面之间观察到的积极结果（TP）与预测阳性（TP + FP）的比率的差异。	在预测所有年龄组合格申请人的贷款接受情况时，模型是否具有同等的精确度？	二进制、多类别平面和连续标签的范围为 [-1, +1]。 • 正值表示可能存在对方面的偏差D原因是在不受欢迎的方面发生了相对更多的误报D. • 接近零的值表示模型正在以同等的精确度预测两个方面的积极结果（接受率）的观测标签。 • 负值表示对平面可能存在偏差一个是由于偏爱的方面出现了相对更多的误报一个.

训练后偏差指标	描述	例如问题	解释指标值
拒绝率的差异 (DRR) (p. 2068)	衡量观察到的负面结果（TN）与预测的负面因素（TN + FN）之间的比率之间的差异。	在预测所有年龄组的不合格申请人的贷款拒绝时，模型是否具有同等的精确度？	二进制、多类别平面和连续标签的范围为 [-1, +1]。 • 正值表示偏差可能是由于偏爱的方面出现相对较多的虚假负面而造成的偏见一个. • 接近零的值表示模型对两个方面的负面结果（拒绝）的观察标签进行了同等精确的预测。 • 负值表示可能是由于不受欢迎的方面出现相对较多的虚假负面引起的偏见D.
准确度差异 (AD) (p. 2069)	衡量偏好和不受欢迎的方面的预测准确度之间的差异。	该模型是否准确地预测所有人口群体的应用程序的标签？	二进制和多类平面标签的范围为 [-1, +1]。 • 正值表示该方面D更多的是误报（I 类错误）或假底片（II 类错误）的组合。这意味着对不受欢迎的方面存在潜在的偏见D. • 小平面的预测准确度时，将出现接近零的值一个类似于 FacetD. • 负值表示该方面一个更多的是误报（I 类错误）或假底片（II 类错误）的组合。这意味着对偏爱的方面存在偏见一个.
待遇平等（TE）(p. 2070)	衡量偏爱和不受欢迎的方面之间的误报与假负面之间的差异。	在贷款申请中，在所有年龄段的人口统计数据中，误报与虚假负面的相对比是否相同？	二进制和多类平面标签的范围：(-∞, +∞)。 • 当面的误报与假负面的比率时，会出现正值一个比方面还要大D. • 当面的误报与假负面的比率时，将出现接近零的值一个类似于 FacetD. • 当面的误报与假负面的比率时，会出现负值一个对于方面来说还不到这一点D.

训练后偏差指标	描述	例如问题	解释指标值
预测标签中的条件人口统计差异 (CDDPL) (p. 2070)	衡量各个方面之间的预测标签之间的差异，但也可以按子组进行衡量。	一些人口群体对贷款申请结果的拒绝比例是否超过接受贷款的比例？	二进制结果、多类别和连续结果的 CDDPL 值范围：[-1, +1] • 正值表示一种结果，其中面D被拒绝超过接受。 • 接近零表示平均来说没有人口差异。 • 负值表示一种结果一个被拒绝超过接受。
反事实翻转（FT）(p. 2071)	检查小平面的每个成员D并评估是否有类似的FACET 成员一个有不同的模型预测。	一组特定年龄人口群组与另一个年龄组的所有功能密切匹配，是否平均支付比其他年龄组高？”	二进制和多类平面标签的范围为 [-1, +1]。 • 当针对不受欢迎的方面作出不利的反事实翻转决定的数量时，会出现正值D超过了有利的。 • 当不利和有利的反事实翻转测试决定的数量平衡时，将出现接近零的值。 • 当不受欢迎的方面作出不利的反事实翻转决定的数量时，会出现负值D低于有利的。

有关训练后偏差指标的其他信息，请参阅金融 Machine Learning 的公平措施系列.

主题

- 预测标签中的正比例差异 (DPPL) (p. 2063)
- 不同的影响力 (DI) (p. 2064)
- 有条件接受的差异（DCCAC）(p. 2065)
- 有条件拒绝（DCR）的差异 (p. 2066)
- 召回差异 (RD) (p. 2067)
- 接受率的差异 (DAR) (p. 2068)
- 拒绝率的差异 (DRR) (p. 2068)
- 准确度差异 (AD) (p. 2069)
- 待遇平等（TE）(p. 2070)
- 预测标签中的条件人口统计差异 (CDDPL) (p. 2070)
- 反事实翻转（FT）(p. 2071)

预测标签中的正比例差异 (DPPL)

预测标签 (DPPL) 指标中正比例的差异决定了模型是否对每个方面预测结果的不同。它被定义为平面积极预测比例（Y=1）之间的差异一个以及平面积极预测的比例 (Y=1)D. 例如，如果模型预测向 60% 的中年群体提供贷款（方面）一个）和 50% 的其他年龄组（方面D)，它可能会对方面有偏见D. 在此示例中，您需要确定

10% 的差异对于偏见的情况来说是否重要。将 DPL 与 DPL 的比较可评估训练后最初存在于数据集中的偏差是增加还是减少模型预测。

预测标签比例差异的公式：

$$DPL = q'_a - q'_d$$

其中：

- $Q'_a n'^{(1)}_a / n_a$是平面的预测比例一个他们得到了价值 1 的积极结果。在我们的例子中，预计将获得贷款的中年方面的比例。$n'^{(1)}_a$表示 facet 的成员数一个谁得到了值 1 和 n 的积极预测结果$_a$是 Facet 的成员数量一个.
- $Q'_d n'^{(1)}_d / n_d$是平面的预测比例D他们得到了价值 1 的积极结果。在我们的例子中，预计将获得贷款的老年人和年轻人的一个方面。$n'^{(1)}_d$表示 facet 的成员数D谁得到了积极的预测结果和 n_d是 Facet 的成员数量D.

如果 DPPL 足够接近 0，这意味着训练后人口平等已经实现。

对于二进制和多类平面标签，标准化的 DPL 值在区间 [-1, 1] 范围内。对于连续标签，值随时间间隔（-∞、+∞）而变化。

- DPPL 值正表示该方面一个与方面相比，预计积极结果的比例更高D.

  这被称为正偏差.
- DPPL 值接近零表示各个方面之间预计积极结果的比例更相等一个和D值为零表示完美的人口平等。
- 负 DPPL 值表示该方面D与方面相比，预计积极结果的比例更高一个. 这被称为负偏差.

不同的影响力 (DI)

可以以比率的形式评估预测标签指标中的正比例差异。

预测标签指标中正比例的比较可以以比率的形式进行评估，而不是像与预测标签中的正比例差异 (DPPL) (p. 2063). 不同影响 (DI) 指标的定义是平面积极预测的比例 (y=1)D超过了平面积极预测的比例（Y=1）一个. 例如，如果模型预测向 60% 的中年群体提供贷款（方面）一个）和 50% 的其他年龄组（方面）D)，然后是 DI = .5/.6 = 0.8，这表明对以方面为代表的另一个老年人群有积极的偏见和不利影响D.

预测标签比例的公式：

$$DI = q'_d / q'_a$$

其中：

- $Q'_a n'^{(1)}_a / n_a$是平面的预测比例一个他们得到了价值 1 的积极结果。在我们的例子中，预计将获得贷款的中年方面的比例。$n'^{(1)}_a$表示 facet 的成员数一个谁得到了积极的预测结果和 n_a是 Facet 的成员数量一个.
- $Q'_d n'^{(1)}_d / n_d$是平面的预测比例Da 谁得到了价值 1 的积极结果。在我们的例子中，预计将获得贷款的老年人和年轻人的一个方面。$n'^{(1)}_d$表示 facet 的成员数D谁得到了积极的预测结果和 n_d是 Facet 的成员数量D.

对于二进制、多类别平面和连续标签，DI 值在区间 [0, ∞) 范围内。

- 值小于 1 表示该小面一个预计积极结果的比例高于方面D. 这被称为正偏差.
- 值为 1 表示人口均等。
- 值大于 1 表示该小平面D预计积极结果的比例高于方面一个. 这被称为负偏差.

有条件接受的差异（DCCAC）

该指标将观察到的标签与模型预测的标签进行比较，并评估各个方面的标签是否相同，以获得预测的积极结果。该指标接近模拟人的偏见，因为它量化了与训练数据集（标签 y）中观察到的结果（标签 y）相比，模型在某个方面预测的积极结果（标签是）多少。例如，如果在培训数据集中观察到更多的接受（积极结果），中年人群的贷款申请（方面）一个）比模型基于资格的预测与包含其他年龄组的方面相比（facet）D），这可能表明有利于中年人群的贷款批准方式存在潜在偏见。

有条件接受差异的公式：

$$DCAC = {}_{a}-c_{d}$$

其中：

- $c_{a}n={}_{a}^{(1)}/n'_{a}{}^{(1)}$是平面值 1（接受）的观察到的积极结果数量的比率一个到平面的预计积极结果（接受）数一个.
- $c_{d}n={}_{d}^{(1)}/n'_{d}{}^{(1)}$是平面值 1（接受）的观察到的积极结果数量的比率D预计的积极结果（接受率）的数量D.

DCACC 指标可以反映出基于资格的优惠待遇的积极和负面偏见。考虑以下基于年龄的偏见对贷款接受的情况。

示例 1：正偏差

假设我们有 100 个中年人的数据集（方面）一个）和来自其他年龄组的 50 人（方面）D) 谁申请贷款，该模型建议从 Facet 开始计算 60一个还有 30 个从方面开始D获得贷款。因此，对于 DPPL 指标而言，预测的比例是公正的，但观察到的标签显示，从平面来看，70一个还有 20 个从方面D获得了贷款。换句话说，该模型向中年人提供的贷款比培训数据中观察到的标签（70/60 = 1.17）少 17%，并向其他年龄组提供的贷款比观察到的标签（20/30 = 0.67）多 33%。DCACC 值的计算得出以下结果：

$$DCCAC = 70/60-20/30 = 1/2$$

正值表明存在对中年层面的潜在偏见一个与另一方面相比，接受率较低D比观察到的数据（被视为无偏见）表明的情况是这种情况。

示例 2：负偏差

假设我们有 100 个中年人的数据集（方面）一个）和来自其他年龄组的 50 人（方面）D) 谁申请贷款，该模型建议从 Facet 开始计算 60一个还有 30 个从方面开始D获得贷款。因此，对于 DPPL 指标而言，预测的比例是公正的，但观察到的标签显示，从平面来看 50一个还有 40 个方面D获得了贷款。换句话说，该模型向中年人提供的贷款比培训数据中观察到的标签（50/60 = 0.83）少 17%，并向其他年龄组提供的贷款比观察到的标签（40/30 = 1.33）多 33%。DCACC 值的计算得出以下结果：

$$DCCAC = 50/60-40/30 = -1/2$$

负值表明存在对方面的潜在偏见D与中年方面相比，接受率较低一个比观察到的数据（被视为无偏见）表明的情况是这种情况。

请注意，您可以使用 DCACC 来帮助您检测人类在循环环环境中监督模型预测的潜在（无意的）偏见。例如，假设模型的预测是公正的，但最终的决定是由一个可以改变模型预测以生成新的最终版本的人（可能有额外的功能）做出的。人类的额外处理可能会无意中从一个方面拒绝不成比例的贷款。DCAC 可以帮助发现这种潜在的偏见。

二进制、多类别平面和连续标签的条件接受差异的值范围为 $(-\infty, +\infty)$。

- 如果观察到的接受人数与预测接受量的比率相对于小平面的预测接受率，则出现正值一个高于平面的相同比率D. 这些值表明从方面来看可能存在对合格申请人的偏见一个. 比率的差异越大，明显的偏差就越极端。
- 当观察到的接受人数与预测接受量的比率相比，将出现接近零的值一个与小平面的比率相似D. 这些值表明，预测的接受率与标签数据中观察到的值一致，而且两个方面的合格申请人也以类似的方式被接受。

- 如果观察到的接受人数与预测接受量的比率相对于小平面的预测接受率，则会出现负值一个小于平面的比率D. 这些值表明从方面来看可能存在对合格申请人的偏见D. 比率的差异越为负面，明显的偏差就越极端。

有条件拒绝（DCR）的差异

此指标将观察到的标签与模型预测的标签进行比较，并评估负面结果（拒绝）的各个方面是否相同。该指标接近模拟人的偏见，因为它量化了与训练数据集中标签建议的建议相比，模型赋予某个方面的负面结果（预测标签 y）的数量（预测标签是）多少。例如，如果对中年人群的贷款申请有更多的拒绝（负面结果）（方面）一个）比模型基于资格的预测与包含其他年龄组的方面相比（facet）D），这可能表明，拒绝贷款的方式可能存在偏见，而这种贷款偏向于中年人群而不是其他群体。

有条件接受差异的公式：

$$DCR = {}_{d}\text{-}r_{a}$$

其中：

- $r_{d}n = {}_{d}^{(0)}/n'{}_{d}^{(0)}$是平面值 0（拒绝）的观察到的负面结果数的比率D到预计的负面结果（拒绝）数D.
- $r_{a}n = {}_{a}^{(0)}/n'{}_{a}^{(0)}$是平面值 0（拒绝）的观察到的负面结果数的比率一个转换为平面值 0（拒绝）的预计负面结果数一个.

DCR 指标可以反映出基于资格的优惠待遇的积极和负面偏见。考虑以下基于年龄的偏见对拒绝贷款的情况。

示例 1：正偏差

假设我们有 100 个中年人的数据集（方面）一个）和来自其他年龄组的 50 人（方面）D) 谁申请贷款，该模型建议从 Facet 开始计算 60一个还有 30 个从方面开始D被拒绝贷款。因此，预测的比例不受 DPPL 指标的偏差，但观察到的标签显示，从平面来看，50一个还有 40 个方面D被拒绝。换句话说，该模型拒绝了来自中年层面的贷款，比培训数据中观察到的标签（50/60 = 0.83）多 17%，拒绝其他年龄组的贷款比观察到的标签（40/30 = 1.33）少 33%。DCR 值量化了各个方面之间观察到的拒绝率与预测拒绝率之间的比率的这种差异。正值表明，与其他群体相比，与观察到的数据（视为无偏见）表明的情况相比，有利于拒绝率较低的中年人群的潜在偏见。

DCR = 40/30-50/60 = 1/2

示例 2：负偏差

假设我们有 100 个中年人的数据集（方面）一个）和来自其他年龄组的 50 人（方面）D) 谁申请贷款，该模型建议从 Facet 开始计算 60一个还有 30 个从方面开始D被拒绝贷款。因此，预测的比例不受 DPPL 指标的偏差，但观察到的标签显示，从平面开始显示 70一个还有 20 个从方面D被拒绝。换句话说，该模型拒绝的中年贷款比培训数据中观察到的标签（70/60 = 1.17）减少 17%，拒绝了其他年龄组的贷款比观察到的标签（20/30 = 0.67）多 33%。负值表示存在有利于小面积的潜在偏见一个与中年方面相比，拒绝率更低一个比观察到的数据（被视为无偏见）表明的情况是这种情况。

DCR = 20/30-70/60 = -1/2

二进制、多类别方面和连续标签的条件拒绝差异的值范围为 (-∞, +∞)。

- 如果观察到的拒绝次数与预计拒绝的比率为准，则会出现正值D对于小平面的比率大于该比率一个. 这些值表明从方面来看可能存在对合格申请人的偏见一个. DCR 指标的值越大，明显的偏差就越极端。
- 如果观察到的拒绝次数与预测接受量的比率相比，将出现接近零的值一个与小平面的比率相似D. 这些值表明，预计的拒绝率与标签数据中观察到的值一致，两个方面的合格申请人也以类似的方式被拒绝。
- 如果观察到的拒绝次数与预计拒绝的比率对小平面的预测拒绝的比率，则会出现负值D小于该比率方面一个. 这些值表明从方面来看可能存在对合格申请人的偏见D. 负 DCR 指标的幅度越大，明显的偏差就越极端。

召回差异 (RD)

召回差异 (RD) 指标是偏爱的方面之间模型召回的差异一个而且不喜欢的方面D. 这些召回中的任何区别都是一种潜在的偏见形式。召回是真正的正值率（TPR），它衡量模型正确预测应获得积极结果的病例的频率。如果所有 y=1 的病例都是正确的预测，那么召回对于那个方面来说是完美的。当模型最大限度地减少被称为 II 型错误的假负面情况时，召回会更大。例如，两个不同群体（方面）中有多少人一个和D）模型可以正确检测到应该有资格获得贷款的？ 如果贷款的召回率很高一个，但贷款对方面来说很低D, 差异提供了对属于 FACET 的群体的这种偏见的一种衡量标准D.

方面召回率差异的公式一个和D：

$$RD = TP_a/(TP_a + FN_a) - TP_d/(TP_d + FN_d) = TPR_a - TPR_d$$

其中：

- TP_a对于方面来说，真正的积极预测是什么？一个.
- FN_a是预计的错误的负面一个.
- TP_d对于方面来说，真正的积极预测是什么？D.
- FN_d是预计的错误的负面D.
- $TPR_a = TP_a/(TP_a + FN_a)$ 是小平面的召回一个。或者它的真正的积极率。
- $TPR_d TP_d/(TP_d + FN_d)$ 是小平面的召回D。或者它的真正的积极率。

例如，考虑以下小面的混淆矩阵：一个和D.

青睐方面的混淆矩阵 a

预测 A 类	实际结果 0	实际结果 1	总计
0	20	5	25
1	10	65	75
总计	30	70	100

不喜欢的方面的混淆矩阵 d

D 类预测	实际结果 0	实际结果 1	总计
0	18	7	25
1	5	20	25
总计	23	27	50

召回差额的值为 RD = 65/70-20/27 = 0.93-0.74 = 0.19，这表示对平面存在偏见D.

面之间召回差异的值范围一个和D对于二进制和多类别分类为 [-1, +1]。此指标不适用于连续标签的情况。

- 当小平面的召回率较高时，将获得正值一个而不是分面D. 这表明该模型发现了更多的真正积极因素的方面一个而不是分面D，这是一种偏见的形式。
- 接近零的值表明正在比较的小面的召回是类似的。这表明该模型在这两个方面都发现了大约相同数量的真正阳性，而且没有偏见。
- 当小平面的召回率较高时，将获得负值D而不是分面一个. 这表明该模型发现了更多的真正积极因素的方面D而不是分面一个，这是一种偏见的形式。

接受率的差异 (DAR)

接受率 (DAR) 指标的差异是各方面的真正正值 (TP) 预测与观察到的阳性 (TP + FP) 的比率的差异一个和D. 该指标衡量了从这两个方面预测接受情况的模型的精度差异。精确度衡量模型确定的合格候选人库中合格候选人的比例。如果预测合格申请人的模型精度在各个方面之间存在差异，那么这是一种偏差，其规模由 DAR 来衡量。

各个方面之间接受率差异的公式一个和D：

$$DAR = {}_{a}/(TP_a + FP_a) - TP_d/(TP_d + FP_d)$$

其中：

- TP_a对于方面来说，真正的积极预测是什么？一个.
- FP_a对于方面来说，预计会出现误报吗？一个.
- TP_d对于方面来说，真正的积极预测是什么？D.
- FP_d对于方面来说，预计会出现误报吗？D.

例如，假设该模型接受 70 名中年申请人（方面）一个）用于贷款（预计正面标签），其中只有 35 个实际接受（观察到的正面标签）。还假设该模型接受来自其他年龄人口统计数据的 100 名申请人（方面D）用于贷款（预计正面标签），其中只有 40 个实际接受（观察到的正面标签）。然后 DAR = 35/70-40/100 = 0.10，这表明对第二个年龄组合格人士存在潜在偏见（方面）D)。

二进制、多类别平面和连续标签的 DAR 值范围为 [-1, +1]。

- 如果预测的阳性（接受率）与观察到的积极结果（合格申请人）的比率（合格申请人）的比率，则会出现正值一个大于平面的相同比率D. 这些值表明可能对不受欢迎的方面存在偏见D是因为在方面出现了相对更多的误报D. 比率的差异越大，明显的偏差就越极端。
- 当各方面的预测正值（接受率）与观察到的积极结果（合格申请人）的比率时，将出现接近零的值一个和D具有类似的值，表明模型正在以同等精确的方式预测积极结果的观察标签。
- 如果预测的正面（接受率）与观察到的积极结果（合格申请人）的比率（合格申请人）的比率，则会出现负值D大于比率方面一个. 这些值表明可能存在对偏爱的方面的偏见一个是因为在方面出现了相对更多的误报一个. 比率的差异越为负面，明显的偏差就越极端。

拒绝率的差异 (DRR)

拒绝率 (DRR) 指标的差异是方面的真正负值 (TN) 预测与观察到的负面值 (TN + FN) 的比率的差异一个和D. 该指标衡量了预测这两个方面的拒绝情况的模型的精度差异。精确度衡量模型确定的不合格候选人库中不合格候选人的比例。如果预测不合格申请人的模型精度在各个方面之间存在差异，那么这是一种偏差，其规模由 DRR 来衡量。

不同方面之间拒绝率差异的公式一个和D：

$$DRR = TN_d/(TN_d + FN_d) - TN_a/(TN_a + FN_a)$$

其中：

- TN_d是预测的真正负面D.
- FN_d是预计的错误的负面D.
- TP_a是预测的真正负面一个.
- FN_a是预计的错误的负面一个.

例如，假设该模型拒绝了 100 名中年申请人（方面）一个）贷款（预计的负面标签），其中 80 个实际上是不合格的（观察到的负面标签）。还假设该模型拒绝了 50 名来自其他年龄人口统计的申请人（方面）D）贷款（预计的负面标签），其中只有 40 个实际上不合格（观察到的负面标签）。然后 DRR = 40/50-80/100 = 0，所以没有表示偏见。

二进制、多类别平面和连续标签的 DRR 值范围为 [-1, +1]。

- 当预测负面（拒绝）与观察到的负面结果（不合格申请人）的比率时，会出现正值D大于平面的相同比率一个. 这些值表明可能存在对偏爱的方面的偏见一个原因是在平面上发生了相对更多的虚假负面一个. 比率的差异越大，明显的偏差就越极端。
- 当各方面的预测负面（拒绝）与观察到的负面结果（不合格申请人）的比率时，将出现接近零的值一个和D具有类似的值，表明模型正在以同等精确的方式预测消极结果的观察标签。
- 如果预测的负面（拒绝）与观察到的负面结果（不合格申请人）的比率（不合格的申请人）的比率，则会出现负值一个大于比率方面D. 这些值表明可能对不受欢迎的方面存在偏见D是因为在方面出现了相对更多的误报D. 比率的差异越为负面，明显的偏差就越极端。

准确度差异 (AD)

准确度差 (AD) 指标是不同方面的预测准确度之间的差异。此指标决定了模型的分类对于一个方面而言是否比另一个方面更准确。AD 表示一个方面是否引发了更大比例的 I 和 II 类错误。但是它不能区分 I 类和 II 类错误。例如，该模型对于不同年龄的人口统计数据可能具有相同的准确度，但是对于一个基于年龄的组而言，错误可能主要是误报（I 类错误），另一个基于年龄组的大部分错误（类型 II 错误）。

此外，如果贷款批准对于中年人口的准确性要高得多（方面）一个) 而不是另一个基于年龄的人口统计数据（面D），要么在第二组中，有更多的合格申请人被拒绝贷款（FN），要么该组的不合格申请人中有更大比例的不符合条件的申请人获得贷款（FP），或者两者兼而有。这可能会导致第二组内部的不公平，即使两个基于年龄的群体发放的贷款比例几乎相同，这表示为接近零的 DPPL 值。

AD 指标的公式是小平面预测准确度之间的差异一个ACC_a，减去那个对于方面DACC_d：

$$AD = ACC_aACC\text{-}ACC_d$$

其中：

- $ACC_a = (TP_a + TN_a)/(TP_a + TN_a + FP_a + FN_a)$
  - TP_a对于方面来说，真正的积极预测是什么？一个
  - TN_a是预测的真正负面一个
  - FP_a对于方面来说，预计会出现误报吗？一个
  - FN_a是预计的错误的负面一个
- $ACC_d = (TP_d + TN_d)/(TP_d + TN_d + FP_d + FN_d)$
  - TP_d对于方面来说，真正的积极预测是什么？D
  - TN_d是预测的真正负面D
  - FP_d对于方面来说，预计会出现误报吗？D
  - FN_d是预计的错误的负面D

例如，假设一个模型批准向 70 名申请人提供贷款来自方面一个的 100 个，拒绝了另外 30 个。10 个不应该获得贷款（FP_a）和 60 个已获批准，应该是（$TP)_a$）。20 个拒绝应该已经获得批准（FN_a）和 10 被正确拒绝（$TN)_a$)。分面的准确性一个如下所示：

$$ACC_a = (60 + 10)/(60 + 10 + 20 + 10) = 0.7$$

接下来，假设一个模型批准向 50 名申请人提供贷款来自方面D的 100 个，拒绝了另外 50 个。10 个不应该获得贷款（FP_a）和 40 个被批准了应该是（$TP)_a$）。40 个拒绝应该已经获得批准（FN_a）和 10 被正确拒绝（$TN)_a$)。分面的准确性一个确定如下所示：

$$ACC_d = (40 + 10)/(40 + 10 + 40 + 10) = 0.5$$

因此，准确度差异是 $AD = ACC_aACC\text{-}ACC_d = 0.7\text{-}0.5 = 0.2$。这表明存在着对方面的偏见D因为该指标是积极的。

二进制和多类别平面标签的 AD 值范围为 [-1, +1]。

- 小平面的预测准确度时出现正值一个比方面还要大D. 这意味着那个方面D更多的是误报（I 类错误）或假底片（II 类错误）的组合。这意味着对不受欢迎的方面存在潜在的偏见D.
- 小平面的预测准确度时，将出现接近零的值一个类似于 FacetD.
- 小平面的预测准确度时，会出现负值D比方面还要大一个t. 这意味着那个方面一个更多的是误报（I 类错误）或假底片（II 类错误）的组合。这意味着对偏爱的方面存在偏见一个.

待遇平等（TE）

治疗平等（TE）是指各个方面之间的假阴性与误报之间的比率的差异一个和D. 该指标的主要思想是评估，即使各组的准确性相同，错误对一个组的伤害是否比另一个组更有害？ 错误率来自误报和假负面的总数，但这两者的细分可能在各个方面有很大不同。TE 衡量错误是以类似方式还是不同的方式跨方面补偿错误。

治疗平等的公式：

$$TE = FN_d/FP_d\text{-}FN_a/FP_a$$

其中：

- FN_d是预计的错误的负面D.
- FP_d对于方面来说，预计会出现误报吗？D.
- FN_a是预计的错误的负面一个.
- FP_a对于方面来说，预计会出现误报吗？一个.

请注意，如果 FP，指标将变为无限$_a$或 FP_d为 0。

例如，假设有来自 Facet 的贷款申请人。一个还有 50 个方面D. 对于分面一个, 8 个被错误地拒绝了贷款 $(FN)_a$）又有 6 个被错误批准（FP_a)。剩下的预测是真的，所以 TP_a+ TN_a= 86。对于分面D，5 被错误地拒绝了（FN_d）和 2 被错误批准（FP_d)。剩下的预测是真的，所以 TP_d+ TN_d=43。对于方面而言，假阴性与误报的比率等于 8/6 = 1.33一个和 5/2 = 2.5 表示刻面D. 因此，TE = 2.5-1.33 = 1.167，尽管两个方面都具有相同的精度：

$$ACC_a = (86)/(86+ 8 +6) = 0.86$$

$$ACC_d = (43)/(43 + 5 + 2) = 0.86$$

二进制和多类别方面标签的条件拒绝差异的值范围为 (-∞, +∞)。没有为连续标签定义 TE 指标。对此指标的解释取决于误报（I 类错误）和假底片（类型 II 错误）的相对重要性。

- 当面的假负面与误报的比率时，会出现正值D比方面还要大一个.
- 当方面的假阴性与误报的比率时，将出现接近零的值一个类似于 FacetD.
- 当面的假负面与误报的比率时，会出现负值D对于方面来说还不到这一点一个.

> **Note**
>
> 之前的版本指出，治疗平等度量计算为 $FP_a/FN_a\text{-}FP_d/FN_d$而不是 $FN_d/FP_d\text{-}FN_a/FP_a$. 虽然可以使用其中任何一个版本。有关更多信息，请参阅 Fairness measures for Machine Learning in Finance。

预测标签中的条件人口统计差异 (CDDPL)

人口差异指标 (DDPL) 决定了是否面D预计被拒的标签比预计可接受的标签的比例要大。它可以对预测的拒绝比例和预测接受率的差异进行比较。此指标与预训练 CDD 指标完全相同，不同之处在于它是根据预测标签而不是观察到的标签计算的。此指标位于范围 (-1, +1)。

平面标签的人口差异预测公式D如下所示：

$$DDPL_dn=_d^{(0)}/n'^{(0)}n'_d^{(1)}/n'^{(1)}P =_d^R(y)^0P)-P_d^{-个}(y)^1)$$

其中：

- $n'^{(0)}n=_a^{(0)}+ n'_d^{(0)}$是刻面的预计拒绝标签的数量一个和D.
- $n'^{(1)}n=_a^{(1)}+ n'_d^{(1)}$是刻面的预计可接受标签的数量一个和D.
- $P_d^R(y)^0)$ 是平面中预测被拒的标签（值 0）的比例D.
- $P_d^{-个}(y)^1)$ 是平面中预测的可接受标签（值 1）的比例D.

为了排除辛普森的悖论，需要预测标签 (CDDPL) 指标中的有条件人口统计差异，该指标将 DDPL 限制在定义数据集上的子组层的属性上。重组可以让人们深入了解较不受欢迎的方面造成明显的人口差异的原因。典型的例子出现在伯克利入院的案例中，男子被接受的总体比率高于女性。但是，当对部门分组进行检查时，各部门的女性入学率都高于男性。解释是，女性向接受率低于男性的部门提出申请。审查分组接受率表明，在接受率较低的部门，实际上接受女性的比率高于男子。

CDDPL 指标通过平均数据集属性定义的子组中发现的所有差异提供了一个单一的度量。它被定义为预测标签（DDPL）中人口差异的加权平均值$_i$）对于每个子组，每个子组差异按包含中的观测值数成比例加权。预测标签中有条件的人口差异的公式如下：

$$CDDPL = (1/n)*\Sigma_i n_i *DDPL_i$$

其中：

- $\Sigma_i n_i$= n 是观测值的总数，n_i是每个子组的观测值数。
- $DDPL_in=_i^{(0)}/n^{(0)}n'_i^{(1)}/n^{(1)}P =_i^R(y)^0P)-P_i^{-个}(y)^1$ ）是该子组的预测标签中的人口差异。

因此，预测标签（DDPL）中一个子组的人口统计差异$_i$) 是预测拒绝的标签比例与每个子组的预测可接受标签的比例之间的差异。

二进制结果、多类别和连续结果的 DDPL 值范围为 [-1, +1]。

- +1：当平面没有预测的拒绝标签时一个或子组，而且没有预计接受小面D或子组。
- 正值表示作为方面的预测标签存在人口差异D或子组的预测拒绝标签的比例大于预测的可接受标签的比例。该值越高，差异越大。
- 接近零的值表明平均来说没有人口差异。
- 负值表示作为方面的预测标签存在人口差异一个或子组的预测拒绝标签的比例大于预测的可接受标签的比例。值越低，差异就越大。
- -1：当面面没有预测的拒绝翻领时D或子组，而且没有预计接受小面一个或子组。

反事实翻转（FT）

翻转的是一种看平面每个成员的方法D并评估是否有类似的 FACET 成员一个有不同的模型预测。Facet 的成员一个被选为从平面看观测值的 k 最近邻D. 我们评估了相对群体中有多少最近的邻居收到不同的预测，翻转的预测可以从正向变为负，反之亦然。

反事实翻转的公式是两套基数之间的差异除以平面成员数D：

$$FT = (F^+-F^-n) /n_d$$

其中：

- F^+= 是不喜欢的方面的数量D结果不利的成员，其最近的邻居处于偏爱方面一个获得了有利的结果。

- F^{-}= 是不喜欢的方面的数量D结果有利的成员，其最近的邻居处于受欢迎的方面一个收到了不利的结果。
- n_d是刻面的样本大小D.

二进制和多类别平面标签的反事实翻转测试值范围为 [-1, +1]。对于连续标签，我们设置了将标签折叠为二进制标签的阈值。

- 当针对不受欢迎的方面作出不利的反事实翻转决定的数量时，会出现正值D超过了有利的。
- 当不利和有利的反事实翻转测试决定的数量平衡时，将出现接近零的值。
- 当不受欢迎的方面作出不利的反事实翻转决定的数量时，会出现负值D低于有利的。

亚马逊 SageMaker 澄清模型的可解释性

亚马逊 SageMaker Explain 提供了有助于解释机器学习 (ML) 模型如何作出预测的工具。这些工具可以帮助机器学习建模者和开发人员以及其他内部利益相关者在部署之前将模型特征作为一个整体了解，并在部署后调试模型提供的预测。机器学习模型如何实现预测的透明度对消费者和监管机构也至关重要。如果他们要接受基于模型的决定，他们需要信任模型预测。 SageMaker 澄清使用与模型无关的功能归因方法。你可以使用它来了解模型为什么在训练后进行预测，并在推断过程中提供每个实例的解释。该实施包括可扩展且高效的实施SHAP. 这是基于来自合作博弈论领域的 Shapley 值的概念，该概念为每个特征赋予特定预测的重要性值。

澄清生成部分依赖图（PDP），显示边际影响功能对机器学习模型的预测结果产生的影响。给定一组输入要素，部分依赖性有助于解释目标响应。它还支持计算机视觉 (CV) 和自然语言处理 (NLP) 可解释性，使用与用于表格数据解释相同的 Shapley Value (SHAP) 算法。

在机器学习上下文中解释的作用是什么？ 一个解释可以被视为对为什么提问这有助于人类理解预测的原因。在机器学习模型的背景下，您可能有兴趣回答以下问题：

- 该模型为什么预测某个申请人的负面结果，例如拒绝贷款？
- 模型如何做出预测？
- 为什么模型做出了错误的预测？
- 哪些功能对模型的行为影响最大？

您可以使用解释来审计和满足监管要求、建立对模型的信任并支持人类决策，以及调试和提高模型性能。

需要满足人类理解 ML 推理的性质和结果的需求，这是所需解释的关键。来自哲学和认知科学学科的研究表明，人们尤其关心对比性的解释，或者解释为什么事件 X 发生，而不是其他一些没有发生的事件 Y。在这里，X 可能是发生的意想不到或令人惊讶的事件，Y 对应于基于他们现有心理模型的期望，被称为基线. 请注意，对于同一事件 X，不同的人可能会根据他们的观点或心理模型 Y 寻求不同的解释。在可解释的人工智能上下文中，你可以将 X 视为正在解释的例子，Y 作为基线，通常选择表示一个信息性或普通示例的基线在数据集中。有时，例如在图像的机器学习建模的情况下，基线可能是隐式的，其中像素都是相同颜色的图像可以作为基线。

示例笔记本

亚马逊 SageMaker 澄清为模型可解释性提供了以下示例笔记本：

- 亚马逊 SageMaker 澄清处理— 使用 SageMaker 澄清以创建处理作业，以检测偏差并使用功能归因解释模型预测。示例包括使用 CSV 和 jsonLine 数据格式、自带容器以及使用 Spark 运行处理作业。
- 使用来解释图像分类 SageMaker 澄清– SageMaker 澄清为您提供了关于计算机视觉模型如何分类图像的见解。
- 用解释对象检测模型 SageMaker 澄清– SageMaker 澄清为您提供了关于计算机视觉模型如何检测物体的见解。

这款笔记本已经验证可在亚马逊上运行 SageMaker 仅限 Studio。如果您需要有关如何在亚马逊打开笔记本电脑的说明 SageMaker 工作室，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86). 如果系统提示您选择内核，请选择Python 3（数据科学）.

主题

- 使用 Shapley 值的功能归因 (p. 2073)
- SHAP 基准的可解释性 (p. 2073)
- 部分依赖图：分析配置和输出 (p. 2074)
- SageMaker 澄清计算机视觉 (p. 2076)
- 创建要素属性基线和可解释性报告 (p. 2077)
- NLP 模型的可解释性报告 (p. 2077)

使用 Shapley 值的功能归因

SageMaker 澄清基于以下概念提供功能归因Shapley Value. 您可以使用 Shapley 值来确定每个功能对模型预测的贡献。这些属性可以用于具体的预测，也可以在全球层面针对整个模型提供。例如，如果你使用 ML 模型进行大学录取，那么解释可以帮助确定 GPA 还是 SAT 分数是对模型的预测负最大责任的功能，然后你可以确定每个功能对确定特定入学决定负有多大责任学生。

SageMaker Cline 从博弈论中采用了 Shapley 价值观的概念，并将其部署在机器学习环境中。Shapley 值提供了一种量化每位玩家对游戏贡献的方法，因此也是根据玩家的贡献将游戏产生的总收益分配给玩家的方法。在这种机器学习背景下， SageMaker 澄清将给定实例上模型的预测视为比赛以及模型中包含的功能作为玩家们. 对于第一个近似值，您可能会想通过量化任一要素的结果来确定每个要素的边际贡献或效果落下来自模型的那个功能或落下模型中的所有其他功能。但是，这种方法没有考虑到模型中包含的特征往往彼此不独立。例如，如果两个要素高度相关，则删除其中一个要素可能不会显著改变模型预测。

为了解决这些潜在的依赖关系，Shapley 值要求必须考虑每个可能的要素组合（或联合）的结果，以确定每个要素的重要性。鉴于d功能，有 2 个^{d}这种可能的功能组合，每种组合都与潜在模型相对应。确定给定功能的归因f，考虑包括在内的边际贡献f在所有不包含的功能组合（和关联模型）中f，然后取平均值。可以表明，Shapley 价值是分配满足某些理想属性的每个功能的贡献或重要性的独特方式。特别是，每个要素的 Shapley 值的总和对应于模型的预测与没有功能的虚拟模型之间的差异。但是，即使是为了合理的价值d，比方说 50 个功能，训练 2 是计算过于望而且不切实际^{d}可能的模型。因此， SageMaker 澄清需要利用各种近似技术。为此目的， SageMaker 澄清使用 SHAPLEY 加法解释 (SHAP)，其中包含了这种近似值，并通过额外的优化设计了可扩展且高效的内核 SHAP 算法的实施。

有关 Shapley 值的其他信息，请参阅解释模型预测的统一方法.

SHAP 基准的可解释性

如前所述，解释通常是截然不同的（也就是说，它们解释了偏离基线的原因）。因此，对于同一模型预测，您可以期望获得对不同基线的不同解释，因此选择基线至关重要。在机器学习上下文中，基线对应于一个假设实例，该实例可以是没有信息要么信息丰富. 在计算 Shapley 值的过程中， SageMaker 澄清在基线和给定实例之间生成几个新实例，其中，通过将要素值设置为基线值来建模不存在要素，并通过将要素值设置为给定实例的值来建模要素的存在。因此，缺少所有要素与基线相对应，所有要素的存在对应于给定的实例。

你怎样才能选择良好的基准？ 通常最好选择信息内容非常低的基线。例如，您可以通过使用数值要素的中位数或平均值以及分类要素的模式，从训练数据集构建平均实例。对于大学录取示例，您可能有兴趣解释为什么特定申请人与基于普通申请人的基准录取率相比，为什么特定申请人被接受。如果未提供，则基线将自动计算 SageMaker 在输入数据集中使用 K-均值或 K-原型进行澄清

或者，您也可以选择生成有关信息性基线的说明。对于大学入学情况，您可能想解释为什么特定申请人与来自相似人口背景的其他申请人相比，被拒绝。在这种情况下，你可以选择一个代表感兴趣的申请人的基线，即那些来自类似人口背景的申请人。因此，您可以使用信息性基线将分析集中于特定模型预测的具体方面。您可以通过将人口统计属性和其他无法处理的要素设置为与给定实例中相同的值来隔离要素进行评估。

部分依赖图：分析配置和输出

部分依赖图 (PDP) 显示了预测的目标响应对对一组感兴趣的输入要素的依赖性。它们在所有其他输入要素的值上被边缘化，被称为补充功能. 直观地，您可以将部分依赖性解释为目标响应，这是每个感兴趣的输入要素的函数。

主题

- 部分依赖性图分析配置 (p. 2074)
- 部分依赖性图分析输出 (p. 2074)

部分依赖性图分析配置

要创建部分依赖图 (PDP)，亚马逊 SageMaker 澄清最初会查找在的 JSON 数组中指定的功能列`analysis_config.json`. 必须在此 JSON 文件中提供用于配置处理作业分析的其他参数。有关配置 PDP 和分析的其他方面的更多信息，请参阅配置分析 (p. 2049).

以下代码包含 JSON 示例`"pdp"`中的对象`"methods"`一个对象`analysis_config.json`。配置文件。

```
{
  "dataset_type":...
  "baseline": [[..]]
  .
  .
  "methods": {
      "shap" : {
         "baseline": ".."
         "num_samples": 100
       },
       "pdp": {
         "features": ["Age", "MaturityMonths"] // The features for which we need to plot
 PDP.
         "grid_resolution": 20, //Required for numerical columns only.
                                //The number of buckets into which the range of values is
 divided.
         "top_k_features": 10,  //Specifies how many of the top features must be used for
 PDP plots. The default is 10.
      },
       .
       .
   }
   .
   .
}
```

Note

如果`"features"`没有在`"pdp"`对象但`"shap"`提供了配置， SageMaker 澄清利用全球 SHAP 结果中的十大功能来绘制 PDP 可视化效果。

部分依赖性图分析输出

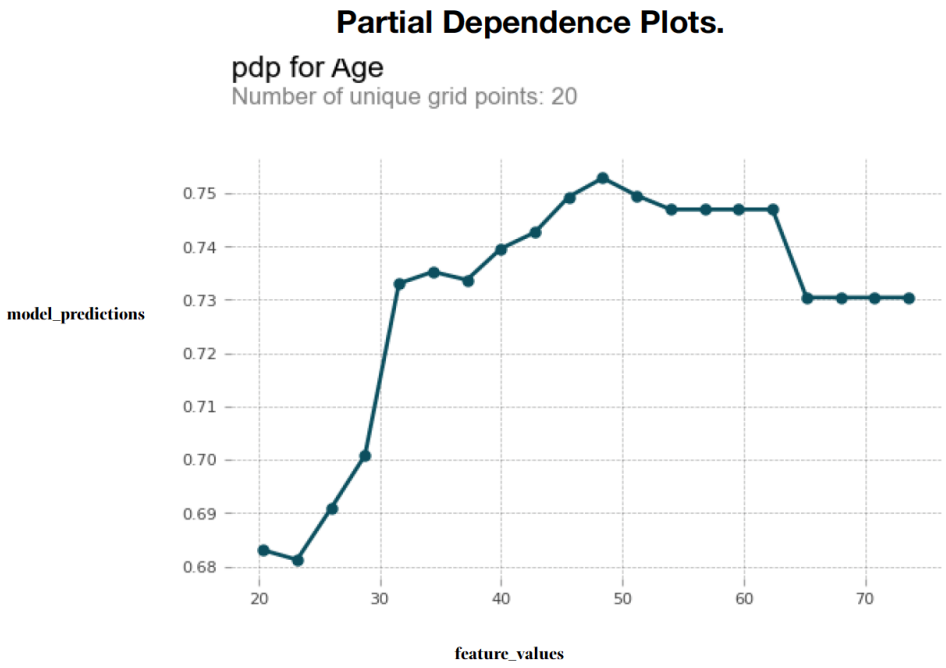
下面的代码显示了在分析 .json 结果文件中返回的部分依赖图 (PDP) 架构的示例。这些区域有：`"pdp"`此分析输出文件中的部分包含生成 PDP 图所需的信息。列表中的每个字典都包含由`feature_name`.

这些区域有：`data_type`指示数据是否`numerical`要么`categorical`. 这些区域有：`feature_values`字段包含要素中存在的值。如果`data_type`澄清推断`categorical`、`feature_values`包含该功能可以假设的所有唯一值。如果`data_type`澄清推断`numerical`，它包含每个`grid_resolution`Clarify 生成的存储桶数量。

如果为特定要素计算部分依赖图，则feature_values、model_predictions，和data_distributions字段被替换为error包含错误消息的字段。

```
{
    "version": "1.0",
    "explanations":{
      "kernel_shap":{
         .
         .
         .
       },
        "pdp": [
            {
                "feature_name": "Age",
                "data_type": "numerical"
                "feature_values": [
                    20.4,
                    23.2,
                    26.0,
                    28.799999999999997,
                    31.599999999999998,
                    34.4,
                    70.8,
                    73.6
                ],
                "model_predictions": [
                    [
                        0.6830344458296895,
                        0.6812452118471265,
                        0.6908621763065458,
                        0.7008252082392573,
                        0.733054383918643,
                        0.7352442337572574,
                        0.7337257475033403,
                        0.7395857129991055,
                    ],

                ],
                "data_distribution": [
                    0.13,
                    0.25,
                    0.15,
                    0.35.
                    0.17
                 ]
            },
            {
                "feature_name": "text_column",
                "data_type": "free_text"
                "error": "Detected data type is not supported for PDP. PDP can only be
 computed for numerical or categorical columns"
            }
        ]
    }
}
```

此 PDP 架构为 “年龄” 功能生成以下部分依赖关系图。PDP 绘制feature_values沿 x 轴。y 轴包含中的值model_predictions 字段中返回的子位置类型。中的每个清单model_predictions字段对应于模型输出中的一个类。



您可以在提供的分析输出路径中查看 report.pdf 文件中的绘图。

SageMaker 澄清计算机视觉

亚马逊 SageMaker 澄清为突出显示正在分析的要素的图像生成热图。特别是，热图可以深入了解您的计算机视觉模型如何对图像进行分类，以及它们如何检测这些图像中的物体。

主题

- 向亚马逊解释图片分类 SageMaker 澄清 (p. 2076)
- 用亚马逊解释对象检测 SageMaker 澄清 (p. 2076)

向亚马逊解释图片分类 SageMaker 澄清

SageMaker Clary 处理作业支持使用Kernelshap 算法. 该算法将图像视为超级像素的集合。给定由图像组成的数据集，处理作业会输出一个图像数据集，其中每个图像显示相关超级像素的热图。

对于使用的示例笔记本 SageMaker 澄清对图像进行分类并解释其分类，请参阅使用来解释图像分类 SageMaker 澄清.

用亚马逊解释对象检测 SageMaker 澄清

亚马逊 SageMaker 澄清可以检测图像中的物体并对其进行分类，然后对预测的检测提供解释。首先，这些对象被归类到指定集合中的一个分类。 SageMaker 澄清为属于该类的每个对象生成置信度分数，以及用于分隔该对象的边界框的坐标。对于有足够概率识别的每个检测物体， SageMaker 使用澄清提取功能简单线性迭代聚类 (SLIC)来自 cikit-learn 库的方法进行图像分割。

SageMaker 然后，澄清可以为在图像场景中检测物体提供解释。它使用 Shapley 值来确定每个要素对模型预测的贡献。它会生成一个热图，显示图像场景中的每个要素对其检测物体的重要性。对象的上下文对于正确识别物体通常非常重要，因此重要的是不仅仅使用裁剪的图像进行分类。

对于使用的示例笔记本 SageMaker 请参阅以及检测图像中的对象并解释其预测，请参阅使用 Amazon SageMaker 澄清解释对象检测模型.

创建要素属性基线和可解释性报告

有关运行方法说明的示例笔记本 SageMaker 澄清 Studio 中的处理作业，为其相对于基线的预测创建解释，请参阅利用亚马逊进行可解释性和偏见检测 SageMaker 澄清.

如果您需要有关如何在亚马逊打开笔记本电脑的说明 SageMaker 工作室，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86). 以下代码示例来自以前列出的示例笔记本。本节讨论了与使用 Shapley 值相关的代码，以提供比较每个功能所作预测的相对贡献的报告。

使用`SHAPConfig`来创建基线。在此示例中，`mean_abs`是指定为基线的所有实例的绝对 SHAP 值的平均值。你用`DataConfig`以配置目标变量、数据输入和输出路径及其格式。

```
shap_config = clarify.SHAPConfig(baseline=[test_features.iloc[0].values.tolist()],
                                 num_samples=15,
                                 agg_method='mean_abs')

explainability_output_path = 's3://{}/{}/clarify-explainability'.format(bucket, prefix)
explainability_data_config = clarify.DataConfig(s3_data_input_path=train_uri,
                                s3_output_path=explainability_output_path,
                                label='Target',
                                headers=training_data.columns.to_list(),
                                dataset_type='text/csv')
```

> **Note**
>
> 中的 SHAP 内核 SageMaker 澄清支持省略 “基线” 参数。在这种情况下，将自动生成基于输入数据集聚类的基线。

然后运行可解释性作业。

```
clarify_processor.run_explainability(data_config=explainability_data_config,
                                     model_config=model_config,
                                     explainability_config=shap_config)
```

在 Studio 中查看结果或从`explainability_output_path`S3 存储桶。

NLP 模型的可解释性报告

亚马逊 SageMaker Explain 支持对自然语言处理 (NLP) 模型的说明。NLP 模型可帮助您了解哪些文本部分对于模型所做的预测最为重要。此功能可用于解释单个局部预测或模型对整体全局解释的预期预测。您可以定义文本段（标记、短语、句子、段落）的长度，以了解和可视化模型在多个粒度级别的行为。

SageMaker 澄清 NLP 可解释性与分类和回归模型兼容。还可以使用澄清来解释模型在包含文本、分类或数字要素的多模态数据集上的行为。多模态数据集的 NLP 可解释性可以帮助您了解每个要素对模型输出的重要性。 SageMaker 澄清支持 62 种语言，可以处理包含多种语言的文本。

要获取输入文本部分的功能重要性，请创建`TextConfig`指定`granularity`文本的各部分和`language`. 然后澄清，然后根据你的选择将文本分解为令牌、句子或段落`granularity`. 它用来自`baseline`. 这些区域

有：baseline默认为空字符串，意思是 SageMaker 澄清删除输入文本的一部分。 SageMaker 然后澄清替换标记（分别是句子和段落），以确定这些替代品如何影响模型的预测，以评估它们的重要性。

```
from sagemaker import clarify

text_config = clarify.TextConfig(
    # Specify the language of your text or use "xx" for multi-language.
    language="en",
    # Choose the granularity of your explanations as tokens, sentences or paragraphs.
    granularity="token"
)
shap_config = clarify.SHAPConfig(
    # For now, we don't support global aggregation,
    # so make sure you retrieve local explanations.
    save_local_shap_values=True,
    text_config=text_config,
    # Determine with what to replace tokens/sentences/paragraphs (based on your choice of
 granularity) with.
    baseline=[["<UNK>"]],
    # You can further specify num_samples and agg_method.
)
```

可解释性工作的其余部分像往常一样编码。

```
explainability_output_path = 's3://{}/{}/clarify-explainability'.format(bucket, prefix)
explainability_data_config = clarify.DataConfig(
    s3_data_input_path=train_uri,
    s3_output_path=explainability_output_path,
    # With text data sometimes detection of headers fails,
    # so we recommend explicitly setting it in the config.
    headers=["text"],
    dataset_type="text/csv")

model_config = clarify.ModelConfig(
    model_name=model_name,
    # Specialized hardware is well-suited for NLP models.
    instance_type="ml.p2.xlarge",
    instance_count=1
)
clarify_processor = clarify.SageMakerClarifyProcessor(
    role=role, instance_count=1, instance_type="ml.m5.xlarge", sagemaker_session=session
)

clarify_processor.run_explainability(
    data_config=explainability_data_config,
    model_config=model_config,
    explainability_config=shap_config)
```

查看本地说明explainability_output_pathS3 存储桶。 SageMaker 澄清并没有总结个别令牌/句子/段落的重要性。

：扩展名

- 多模态数据-如果您的数据包含的不止是文本，例如分类或数字要素，则可以在标题和基线中添加更多列以容纳这些要素。
- JSON-如果模型输入或输出数据采用 JSON 线格式，请指定dataset_type中的DataConfig或者accept_type和content_type中的ModelConfig.
- 多类别模型输出-如果你的模型进行了多级或多标签预测， SageMaker Clplain 计算每个课程的重要性。
- 模型输出-某些模型输出包含超过一个分数。对三个示例的预测返回一个形状矩阵[3, 1]例如[[0.1], [-1.1], [10.2], [0.0], [3.2]]. 您可以指定model_scores在run_explainability. 例如，如

果你的模型预测了五个类别的概率并另外输出获胜类别，`["c1", [0.1, 0.4, 0.3, 0.2]]`，然后你可以设置`model_scores=1`.

快速结果的提示：

- 在开发阶段，使用一小部分示例进行快速迭代。
- 要向上扩展以进行部署，请增加`instance_count`用于处理作业和预测变量。这指示 SageMaker 澄清如何对数据进行分区 parallel 处理多台计算机。对于 parallel 处理，我们建议使用镶木地板数据格式，因为 Spark 可能会遇到 CSV 文件中的特殊字符。
- 选择合适的`instance_type`为您的模型。GPU 实例通常适用于 NLP 模型。
- 要保存不必要的计算，请考虑截断文本。许多模型都使用最大文本长度。通过在预处理步骤中应用截断，SageMaker 澄清不会浪费资源来确定模型忽视的文本部分的功能重要性。

故障排除 SageMaker 澄清处理任务

如果遇到与 SageMaker 澄清处理作业，请参阅以下方案以帮助确定问题。

> Note
>
> 失败原因和退出消息旨在包含描述性消息和例外（如果遇到）在运行期间。一个常见的原因是参数无效或缺少参数。如果您遇到不清楚、混乱或误导性的消息或无法找到解决方案，请提交反馈。

主题

- 处理作业未能完成 (p. 2079)
- 处理作业没有结果就完成了你会得到 CloudWatch 警告消息 (p. 2079)
- 分析配置无效的错误消息 (p. 2080)
- 几个或所有指标的偏差指标计算失败 (p. 2080)
- 分析配置与数据/模型输入/输出之间不匹配 (p. 2080)
- 模型返回 500 内部服务器错误或容器由于模型错误而退回到每条记录的预测 (p. 2080)
- 执行角色无效 (p. 2080)
- 无法下载数据 (p. 2080)
- 无法连接到 SageMaker (p. 2080)

处理作业未能完成

如果处理作业未能完成，可以尝试以下操作：

- 直接在运行作业的笔记本中检查作业日志。作业日志位于启动运行的笔记本电脑单元的输出中。
- 在 CloudWatch 中检查作业日志。
- 在笔记本中添加以下行以描述最后的处理作业并查找失败原因并退出消息：
  - `clarify_processor.jobs[-1].describe()`
- 执行下面的操作Amazon用于描述处理作业并查找失败原因并退出消息的 CLI 命令：
  - `aws sagemaker describe-processing-job –processing-job-name <processing-job-id>`

处理作业没有结果就完成了你会得到 CloudWatch 警告消息

如果处理作业已完成但没有找到结果并且在 CloudWatch 日志上显示 “信号 15 收到，正在清理”，这表明该作业已停止，原因是客户请求调用`StopProcessingJob`API 或者该作业已经用完完了分配的时间了。在后面的情况下，请检查作业配置中的最长运行时间 (`max_runtime_in_seconds`) 并根据需要增加它。

分析配置无效的错误消息

- 如果你收到错误消息 “无法将分析配置加载为 JSON。 “，这意味着处理作业的分析配置输入文件不包含有效的 JSON 对象。使用 JSON linter 检查 JSON 对象的有效性。
- 如果你收到错误消息 “分析配置模式验证错误。 “，这意味着处理作业的分析配置输入文件包含某些字段值的未知字段或无效类型。查看文件中的配置参数，然后使用配置规范文件。

几个或所有指标的偏差指标计算失败

如果您收到以下错误消息之一：“预测的标签列中没有标签值，则预测指数正数系列包含所有 False 值。” 或者 “预测标签列” 系列数据类型与标签列系列不同。 “，请尝试以下操作：

- 检查是否使用了正确的数据集。
- 检查数据集大小是否太小；例如，它是否只包含几行。这可能会导致模型输出具有相同的值，或者数据类型推断错误。
- 检查标签或刻面是被视为连续还是分类的。 SageMaker 澄清使用启发式算法来确定`DataType`. 对于训练后的偏差指标，模型返回的数据类型可能与数据集中的数据类型不匹配，或 SageMaker 澄清可能无法正确转换它。
  - 在偏差报告中，您应该看到分类列的单个值或连续列的间隔。
  - 例如，如果一列的值为 0.0 和 1.0 作为浮点数，那么即使唯一值太少也会被视为连续值。

分析配置与数据/模型输入/输出之间不匹配

- 检查分析配置中的基线格式是否与数据集格式相同。
- 如果你收到错误消息 “无法将字符串转换为浮点数。 “，请检查格式是否正确指定。它还可能表明模型预测的格式与标签列不同，也可能表明标签或概率的配置不正确。
- 如果你收到错误消息 “找不到小面”。或者 “标题必须包含标签。” 或者 “配置中的标题与数据集中的列数不匹配。” 或 “找不到功能名称。 “，检查标题是否与列匹配。
- 如果收到错误消息 “数据必须包含要素。 “，请检查 JSON Line 的内容模板，并将其与数据集示例进行比较（如果可用）。

模型返回 500 内部服务器错误或容器由于模型错误而退回到每条记录的预测

如果您收到错误消息 “由于模型错误而回退到每条记录预测。 “，这可能表明模型无法处理批量大小或受到限制，或者由于序列化问题，只是不接受容器传递的输入。你应该查看 CloudWatch SageMaker 终端节点的日志并查找错误消息或追踪信息。对于模型限制情况，使用不同的实例类型或增加终端节点的实例数量可能有所帮助。

执行角色无效

这表示提供的角色不正确或缺少所需的权限。检查用于配置处理作业的角色及其权限，并验证角色的权限和信任策略。

无法下载数据

这表示无法下载作业输入以启动作业。检查存储桶名称和数据集的权限以及配置输入。

无法连接到 SageMaker

这表明工作无法到达 SageMaker 服务终端节点。检查处理作业的网络配置设置并验证 VPC 配置。

Amazon SageMaker 中的增量训练

随着时间的推移，您可能会发现模型生成的推理不如以前那么好。通过增量训练，您可以使用来自现有模型的构件，并使用扩展的数据集 来训练新模型。增量训练可节省时间和资源。

使用增量训练可以：

- 使用扩展的数据集训练新模型，该数据集包含以前训练中未考虑在内的基础模式，使得以前训练得到的模型性能不佳。
- 在训练作业中使用来自热门公开模型中的模型构件或部分模型构件。您无需从头开始训练新的模型。
- 恢复已停止的训练作业。
- 使用不同超参数设置或使用不同数据集，训练模型的多个变体。

有关训练作业的更多信息，请参阅使用 Amazon SageMaker 训练模型 (p. 7)。

您可以使用 SageMaker 控制台或Amazon SageMaker Python 开发工具包.

> **Important**
>
> 目前只有三种内置算法支持增量训练：对象检测算法 (p. 1691)、图像分类算法 (p. 1620) 和 语义分割算法 (p. 1711)。

主题

- 执行增量训练（控制台） (p. 2081)
- 执行增量训练 (API) (p. 2083)

执行增量训练（控制台）

要完成此过程，您需要：

- 存储训练的 Amazon Simple Storage Service (Amazon S3) 存储桶 URI数据.
- 您要存储作业输出的 S3 存储桶 URI。
- 存储训练代码的 Amazon Elastic 容器注册表路径。有关更多信息，请参阅 Docker 注册表路径和示例代码 (p. 793)。
- S3 存储桶的 URL，您将要在增量训练中使用的模型构件存储在该存储桶。要查找模型构件的 URL，请查看创建模型所用训练作业的详细信息页面。要查找详细信息页面，请在 SageMaker 控制台中，选择推理，选择模型，然后选择模型。

要重启已停止的训练作业，请在详细信息页面中使用存储的模型构件的 URL，就像您对模型或者已完成的训练作业那样。

执行增量训练（控制台）

1. 打开位于 https://console.aws.amazon.com/sagemaker/ 的 Amazon SageMaker 控制台。
2. 在导航窗格中，选择 Training (训练)，然后选择 Training jobs (训练作业)。
3. 选择 Create training job (创建训练作业)。
4. 为训练作业提供一个名称。该名称必须是唯一的Amazon中的地区Amazonaccount. 训练作业名称必须为 1 到 63 个字符。有效字符：a-z、A-Z、0-9 和 . : + = @ _ % - （连字符）。
5. 选择要使用的算法。有关算法的信息，请参阅使用 Amazon SageMaker 内置算法或预训练模型 (p. 787)。
6. （可选）对于 Resource configuration (资源配置)，保留默认值或增加资源消耗，以减少计算时间。

a. （可选）对于 Instance type (实例类型)，选择您要使用的 ML 计算实例类型。在大多数情况下 ml.m4.xlarge 便足够。
b. 对于 Instance count (实例计数)，请使用默认值 1。
c. （可选）对于 Additional volume per instance (GB) (每个实例的附加卷 (GB))，请选择您要预配置的 ML 存储卷的大小。在大部分情况下，您可以使用默认值 1。如果您使用大数据集，请使用较大的大小。

7. 为训练数据集提供有关输入数据的信息。

a. 对于 Channel name (通道名称)，请保留默认值 (**`train`**) 或为训练数据集输入更有意义的名称，例如 **`expanded-training-dataset`**。
b. F要么InputMode，选择File (文件). 对于增量训练，您需要使用文件输入模式。
c. 对于 S3 data distribution type (S3 数据分配类型)，请选择 FullyReplicated。这会导致每个 ML 计算实例使用扩展数据集以增量方式进行训练时。
d. 如果扩展数据集未压缩，请将 Compression type (压缩类型) 设置为 None (无)。如果使用 Gzip 压缩了扩展数据集，请将其设置为 Gzip。
e. （可选）如果您使用的是文件输入模式，请将 Content type (内容类型) 留空。对于管道输入模式，请指定相应的 MIME 类型。内容类型 是多用途 Internet 邮件扩展 (MIME) 类型的数据。
f. 对于 Record wrapper (记录包装程序)，如果数据集以 RecordIO 格式保存，请选择 RecordIO。如果您的数据集未另存为 RecordIO 格式的文件，请选择无.
g. 适用于S3 数据类型，如果将数据集作为单个文件存储，请选择S3Prefix. 如果数据集作为文件夹中的多个文件存储，请选择 Manifest (清单)。
h. 对于 S3 location (S3 位置)，请提供存储扩展数据集位置的路径的 URL。
i. 选择完成。

8. 要在训练作业中使用模型构件，您需要添加新通道并提供有关模型构件所需的信息。

a. 对于 Input data configuration (输入数据配置)，选择 Add channel (添加通道)。
b. 对于 Channel name (通道名称)，输入 **`model`** 以标识此通道作为模型构件的源。
c. 对于 InputMode，选择 File (文件)。模型构件存储为文件。
d. 对于 S3 data distribution type (S3 数据分配类型)，请选择 FullyReplicated。这指示各个 ML 计算实例应使用所有模型构件进行训练。
e. 对于 Compression type (压缩类型)，选择 None (无)，因为我们为通道使用模型。
f. 将 Content type (内容类型) 留空。内容类型 是多用途 Internet 邮件扩展 (MIME) 类型的数据。对于模型构件，我们将其留空。
g. 将 Record wrapper (记录包装程序) 设置为 None (无)，因为模型构件未以 RecordIO 格式存储。
h. 对于 S3 data type (S3 数据类型)，如果您使用内置算法或者将模型存储为单个文件的算法，请选择 S3Prefix。如果您使用将模型存储为多个文件的算法，请选择 Manifest (清单)。
i. 对于 S3 location (S3 位置)，请提供存储模型构件位置的路径的 URL。通常，模型使用名称 `model.tar.gz` 存储。要查找模型构件的 URL，请在导航窗格中，选择 Inference (推理)，然后选择 Models (模型)。从模型列表中，选择要显示其详细信息页的模型。模型构件的 URL 在 Primary container (主容器) 下方列出。
j. 选择完成。

9. 对于 Output data configuration (输出数据配置)，请提供以下信息：

a. 对于 S3 location (S3 位置)，键入存储输出数据的 S3 存储桶的路径。
b. （可选）对于 Encryption key (加密密钥)，您可以添加 Amazon Key Management Service (Amazon KMS) 加密密钥来加密静态的输出数据。提供密钥 ID 或其 Amazon 资源编号 (ARN)。有关更多信息，请参阅 KMS 托管的加密密钥。

10. （可选）对于 Tags (标签)，向训练作业添加一个或多个标签。一个标签您可以定义并分配到的元数据。Amazon资源的费用。在这种情况下，您可以使用标签来帮助您管理训练作业。标签包含由您定义的键和值。例如，您可能希望创建一个标签，其键为 **`Project`**，其值引用与训练作业相关的项目，例如 **`Home value forecasts`**。

11. 选择 Create training job (创建训练作业)。SageMaker 创建并运行训练作业.

训练作业完成后，新训练的模型构件存储在 S3 output path (S3 输出路径) 下，这是您在 Output data configuration (输出数据配置) 字段中提供的位置。要部署模型以获取预测，请参阅第 5 步：将模型部署到 Amazon EC2 (p. 68)。

执行增量训练 (API)

此示例演示如何使用 SageMaker API，通过 SageMaker 图像分类算法和加州理工 256 影像数据集，然后使用第一个模型训练一个新模型。它使用 Amazon S3 作为输入源和输出源。有关使用增量训练的更多信息，请参阅增量训练示例笔记本。

> **Note**
>
> 对于此示例，我们在增量训练中使用原始数据集，不过您可以使用不同的数据集，例如包含新增示例的数据集。将新数据集上传到 S3，并对用于训练新模型的 `data_channels` 变量进行调整。

获取Amazon Identity and Access Management授予所需权限并初始化环境变量的 IAM 角色：

```
import sagemaker
from sagemaker import get_execution_role

role = get_execution_role()
print(role)

sess = sagemaker.Session()

bucket=sess.default_bucket()
print(bucket)
prefix = 'ic-incr-training'
```

获取图像分类算法的训练图像：

```
from sagemaker.amazon.amazon_estimator import get_image_uri

training_image = get_image_uri(sess.boto_region_name, 'image-classification',
 repo_version="latest")
#Display the training image
print (training_image)
```

下载训练和验证数据集，然后将其上载到 Amazon Simple Storage Service (Amazon S3)：

```
import os
import urllib.request
import boto3

# Define a download function
def download(url):
    filename = url.split("/")[-1]
    if not os.path.exists(filename):
        urllib.request.urlretrieve(url, filename)

# Download the caltech-256 training and validation datasets
download('http://data.mxnet.io/data/caltech-256/caltech-256-60-train.rec')
download('http://data.mxnet.io/data/caltech-256/caltech-256-60-val.rec')

# Create four channels: train, validation, train_lst, and validation_lst
s3train = 's3://{}/{}/train/'.format(bucket, prefix)
s3validation = 's3://{}/{}/validation/'.format(bucket, prefix)

# Upload the first files to the train and validation channels
```

```
!aws s3 cp caltech-256-60-train.rec $s3train --quiet
!aws s3 cp caltech-256-60-val.rec $s3validation --quiet
```

定义训练超参数：

```
# Define hyperparameters for the estimator
hyperparams = { "num_layers": "18",
                "resize": "32",
                "num_training_samples": "50000",
                "num_classes": "10",
                "image_shape": "3,28,28",
                "mini_batch_size": "128",
                "epochs": "3",
                "learning_rate": "0.1",
                "lr_scheduler_step": "2,3",
                "lr_scheduler_factor": "0.1",
                "augmentation_type": "crop_color",
                "optimizer": "sgd",
                "momentum": "0.9",
                "weight_decay": "0.0001",
                "beta_1": "0.9",
                "beta_2": "0.999",
                "gamma": "0.9",
                "eps": "1e-8",
                "top_k": "5",
                "checkpoint_frequency": "1",
                "use_pretrained_model": "0",
                "model_prefix": "" }
```

创建评估程序对象并使用训练和验证数据集训练第一个模型：

```
# Fit the base estimator
s3_output_location = 's3://{}/{}/output'.format(bucket, prefix)
ic = sagemaker.estimator.Estimator(training_image,
                                   role,
                                   instance_count=1,
                                   instance_type='ml.p2.xlarge',
                                   volume_size=50,
                                   max_run=360000,
                                   input_mode='File',
                                   output_path=s3_output_location,
                                   sagemaker_session=sess,
                                   hyperparameters=hyperparams)

train_data = sagemaker.inputs.TrainingInput(s3train, distribution='FullyReplicated',
                                        content_type='application/x-recordio',
 s3_data_type='S3Prefix')
validation_data = sagemaker.inputs.TrainingInput(s3validation,
 distribution='FullyReplicated',
                                             content_type='application/x-recordio',
 s3_data_type='S3Prefix')

data_channels = {'train': train_data, 'validation': validation_data}

ic.fit(inputs=data_channels, logs=True)
```

要使用模型来增量训练另一个模型，请创建新的估算器对象并为 `model_uri` 输入参数使用模型构件（在此例中为 `ic.model_data`）：

```
# Given the base estimator, create a new one for incremental training
incr_ic = sagemaker.estimator.Estimator(training_image,
                                        role,
```

```
                                        instance_count=1,
                                        instance_type='ml.p2.xlarge',
                                        volume_size=50,
                                        max_run=360000,
                                        input_mode='File',
                                        output_path=s3_output_location,
                                        sagemaker_session=sess,
                                        hyperparameters=hyperparams,
                                        model_uri=ic.model_data) # This parameter will
 ingest the previous job's model as a new channel
incr_ic.fit(inputs=data_channels, logs=True)
```

训练作业完成后，新训练的模型构件存储在 `S3 output path` 下，这是您在 `Output_path` 中提供的位置。要部署模型以获取预测，请参阅第 5 步：将模型部署到 Amazon EC2 (p. 68)。

访问训练数据

创建训练作业时，您可以指定训练数据集的位置以及用于访问该数据的输入模式。对于数据位置，根据输入模式，Amazon SageMaker 支持 Amazon Simple Storage Service (Amazon S3)、Amazon Elastic File System (Amazon EFS) 和 Amazon FSx for Lustre。输入模式决定在培训作业开始时是流式传输还是下载数据。

输入模式

- File (文件)模式向训练容器显示数据集的文件系统视图。数据源可以是 Amazon S3 或 Amazon EFS 和 Amazon FSx 远程文件系统。

  File (文件)模式将训练数据从存储位置下载到 Docker 容器的本地目录。训练在完整数据集下载完毕后开始。
- 管道模式直接从 Amazon S3 数据源流式传输数据。流式传输可以提供比启动时间和更好的吞吐量`File`模式。

  直接流式传输数据时，您可以减小培训实例使用的 Amazon EBS 卷的大小。`Pipe`模式只需要足够的磁盘空间来存储最终模型构件。
- 快速文件模式允许文件系统访问 Amazon S3 数据源，同时利用`Pipe`模式。在训练开始时，`FastFile`模式标识数据文件但不下载它们。训练可以在不等待整个数据集下载的情况下开始。如果提供的 Amazon S3 前缀中的文件较少，启动时间会缩短。

  与此相反`Pipe`模式，`FastFile`模式适用于随机访问数据。但是，按顺序读取数据时效果最佳。`FastFile`模式不支持增强清单文件。

有关更多信息，请参阅 。Amazon SageMaker 如何提供训练信息 (p. 2430)，`CreateTrainingJob`API 和`TrainingInputMode`在`AlgorithmSpecification`.

Amazon SageMaker 中的托管的 Spot

利用托管的 Amazon EC2 Amazon SageMaker pot 实例，您可以轻松训练机器学习模型。托管的 Spot 训练可以针对按需实例优化训练模型的成本高达 90%。SageMaker 代表您管理竞价型中断。

托管的 Spot 训练使用 Amazon EC2 Spot 实例来运行训练作业而不是按需实例。您可以指定哪些训练作业使用 Spot 实例，以及指定使用 Amazon EC2 SageMaker pot 实例等待作业运行多长时间的停止条件。CloudWatch 中提供了在训练期间生成的指标和日志。

Amazon SageMaker 自动模型优化（也称为超参数优化）可以使用托管的 Spot 训练。有关自动模型优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939).

Spot 实例可能会中断，导致作业开始或结束所花的时间更长。您可以将托管的 Spot 训练作业配置为使用检查点。SageMaker 会将检查点数据从本地路径复制到 Amazon S3。重新启动作业时，SageMaker 会将来自 Amazon S3 的数据复制回本地路径。然后，训练作业可以从最后一个检查点恢复，而不是重新启动。有关检查点操作的更多信息，请参阅 在 Amazon SageMaker 中使用检查点 (p. 2087)。

> Note
>
> 除非您的训练作业能够快速完成，否则我们建议您对托管的 Spot 训练使用检查点。不执行检查点操作的 SageMaker 内置算法和市场算法目前将 `MaxWaitTimeInSeconds` 限制为 3600 秒（60 分钟）。

主题

- 使用托管的 Spot 训练 (p. 2086)
- 托管的 Spot 训练生命周期 (p. 2086)

使用托管的 Spot 训练

要使用托管的 Spot 训练，请创建一个训练作业。将 `EnableManagedSpotTraining` 设置为 `True` 并指定 `MaxWaitTimeInSeconds`。`MaxWaitTimeInSeconds` 必须大于 `MaxRuntimeInSeconds`。有关创建训练作业的更多信息，请参阅 DescribeTrainingJob。

您可以使用公式 `(1 - (BillableTimeInSeconds / TrainingTimeInSeconds)) * 100` 计算使用托管的 Spot 训练时节省的时间。例如，如果 `BillableTimeInSeconds` 为 100，`TrainingTimeInSeconds` 为 500，则表示时间节省 80%。

要了解如何在 Amazon SageMaker 竞价型实例上运行培训作业以及托管竞价培训如何工作并缩短计费时间，请参阅以下示例笔记本：

- 利用 TensorFlow 托管的 Spot 训练
- 利用 xGBoost 托管的 Spot 训练
- 利用 MXNet 托管的 Spot 训练
- Amazon SageMaker 托管式 Spot 训练示例 GitHub 存

托管的 Spot 训练生命周期

你可以使用`TrainingJobStatus`和`SecondaryStatus`返回方DescribeTrainingJob. 下面的列表显示了 `TrainingJobStatus` 和 `SecondaryStatus` 值如何根据训练场景而变化：

- 在训练期间不间断地获得的 Spot 实例
  1. `InProgress`: `Starting`→ `Downloading` → `Training` → `Uploading`
- Spot 实例中断一次。之后，获得了足够的 Spot 实例来完成训练工作。
  1. `InProgress`: `Starting` → `Downloading` → `Training` → `Interrupted` → `Starting` → `Downloading` → `Training` → `Uploading`
- Spot 实例中断两次并且超出了 **`MaxWaitTimeInSeconds`**。
  1. `InProgress`: `Starting` → `Downloading` → `Training` → `Interrupted` → `Starting` → `Downloading` → `Training` → `Interrupted` → `Downloading` → `Training`
  2. `Stopping`: `Stopping`
  3. `Stopped`: `MaxWaitTimeExceeded`
- Spot 实例从未启动。
  1. `InProgress`: `Starting`
  2. `Stopping`: `Stopping`
  3. `Stopped`: `MaxWaitTimeExceeded`

在 Amazon SageMaker 中使用检查点

在训练期间使用 Amazon SageMaker 中的检查点来保存机器学习 (ML) 模型的状态。检查点是模型的快照，可以通过 ML 框架的回调函数进行配置。您可以使用保存的检查点从上次保存的检查点重新启动训练作业。

SageMaker 培训机制使用 Amazon EC2 实例上的训练容器，检查点文件保存在容器的本地目录下。SageMaker 提供了将检查点从本地路径复制到 Amazon S3 的功能。使用检查点，您可以执行以下操作：

- 由于训练作业或实例意外中断，请在训练中保存模型快照。
- 将来从检查点恢复训练模型。
- 在训练的中间阶段分析模型。
- 将检查点与 SageMaker 托管现场培训结合使用，以节省培训成本

如果您在 SageMaker 托管现场训练中使用检查点，则 SageMaker 会管理在竞价型实例上检查模型训练并在下一个竞价型实例上恢复训练作业。借助 SageMaker 托管现场训练，您可以显著减少训练机器学习模型的计费时间。有关更多信息，请参阅 Amazon SageMaker 中的托管的 Spot (p. 2085)。

主题

- SageMaker 中框架和算法的检查点 (p. 2087)
- 启用检查点 (p. 2088)
- 浏览检查点文件 (p. 2089)
- 从检查点恢复训练 (p. 2089)
- 检查点的注意事项 (p. 2090)

SageMaker 中框架和算法的检查点

使用检查点来保存 SageMaker 中基于首选框架构建的机器学习模型的快照。

支持检查点的 SageMaker 框架和算法

SageMaker 支持 Deep Learning Containers 的检查点和内置算法的子集，而无需更改训练脚本。SageMaker 将检查点保存到默认的本地路径`'/opt/ml/checkpoints'`并将它们复制到 Amazon S3。

- Deep Learning Containers：TensorFlow、PyTorch、MXNet, 和HuggingFace

  Note

  如果您使用的是 HuggingFace 框架估计器，则需要通过超参数指定检查点输出路径。有关更多信息，请参阅 。在 Amazon SageMaker 上运行培训中的HuggingFace 文档.
- 内置算法：图像分类、对象检测、语义分割, 和XGBoost（0.90-1 或更高版本）

  Note

  如果您在框架模式（脚本模式）中使用 xgBoost 算法，则需要带有手动配置的检查点的 xgBoost 训练脚本。有关保存模型快照的 XGBoost 训练方法的更多信息，请参阅训练 XGBoost在xgBoost Python SDK 文档.

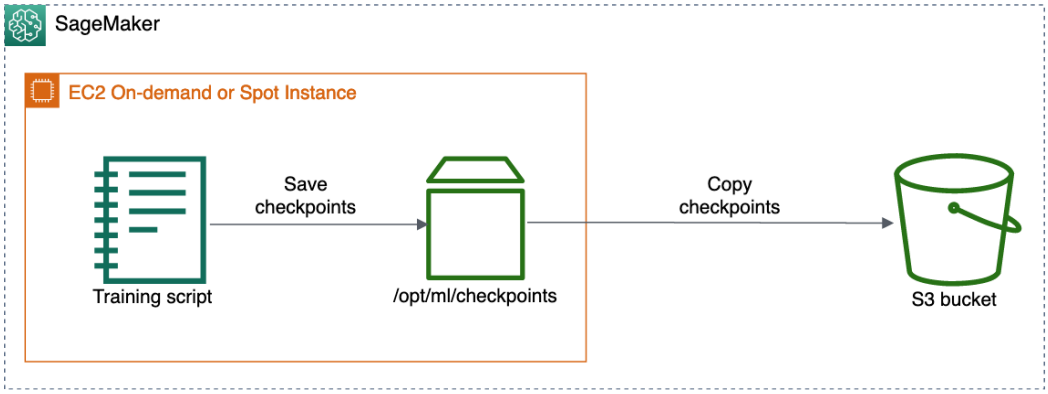
如果在托管现场训练作业中使用了不支持检查点的预构建算法，则 SageMaker 不允许作业的最长等待时间超过一小时，以限制因中断而浪费的训练时间。

用于自定义培训容器和其他框架

如果您使用自己的训练容器、训练脚本或上一节中未列出的其他框架，则必须使用回调或训练 API 正确设置训练脚本，以将检查点保存到本地路径（`'/opt/ml/checkpoints'`）然后从训练脚本中的本地路径加载。SageMaker 估算器可以与本地路径同步并将检查点保存到 Amazon S3。

启用检查点

启用检查点后，SageMaker 将检查点保存到 Amazon S3，并将训练作业与检查点 S3 存储桶同步。



以下示例演示在构建 SageMaker 估计程序时如何配置检查点路径。要启用检查点，请添加`checkpoint_s3_uri`和`checkpoint_local_path`估计器的参数。

以下示例模板演示如何创建通用 SageMaker 估计程序并启用检查点。您可以将此模板用于支持的算法，方法是指定`image_uri`参数。要查找 SageMaker 支持检查点的算法的 Docker 图像 URI，请参阅Docker 注册表路径和示例代码 (p. 793). 您还可以更换`estimator`和`Estimator`与其他 SageMaker 框架的估计器父类和估计器类一起使用，例如`TensorFlow`、`PyTorch`、`MXNet`、`HuggingFace`和`XGBoost`.

```
import sagemaker
from sagemaker.estimator import Estimator

bucket=sagemaker.Session().default_bucket()
base_job_name="sagemaker-checkpoint-test"
checkpoint_in_bucket="checkpoints"

# The S3 URI to store the checkpoints
checkpoint_s3_bucket="s3://{}/{}/{}".format(bucket, base_job_name, checkpoint_in_bucket)

# The local path where the model will save its checkpoints in the training container
checkpoint_local_path="/opt/ml/checkpoints"

estimator = Estimator(
    ...
    image_uri="<ecr_path>/<algorithm-name>:<tag>" # Specify to use built-in algorithms
    output_path=bucket,
    base_job_name=base_job_name,

    # Parameters required to enable checkpointing
    checkpoint_s3_uri=checkpoint_s3_bucket,
    checkpoint_local_path=checkpoint_local_path
)
```

以下两个参数指定检查点的路径：

- `checkpoint_local_path`— 指定模型在训练容器中定期保存检查点的本地路径。默认路径设置为`'/opt/ml/checkpoints'`. 如果您正在使用其他框架或自带训练容器，请确保训练脚本的检查点配置指定了`'/opt/ml/checkpoints'`.

Note

我们建议将本地路径指定为'/opt/ml/checkpoints'以与默认的 SageMaker 检查点设置保持一致。如果您更喜欢指定自己的本地路径，请确保与训练脚本中的检查点保存路径和checkpoint_local_pathSageMaker 估计器的参数。

- checkpoint_s3_uri— 实时存储检查点的 S3 存储桶的 URI。

要查找 SageMaker 估计器参数的完整列表，请参阅估计器 API中的Amazon SageMaker Python 开发工具包文档.

浏览检查点文件

使用 SageMaker Python SDK 和 Amazon S3 控制台查找检查点文件。

以编程方式查找检查点文件

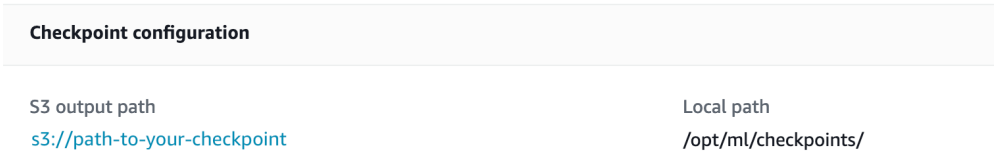
要检索保存检查点的 S3 存储桶 URI，请检查以下估算器属性：

```
estimator.checkpoint_s3_uri
```

这将返回请求时配置的检查点的 Amazon S3 输出路径CreateTrainingJob请求. 要使用 Amazon S3 控制台查找保存的检查点文件，请按以下步骤操作。

从 Amazon S3 控制台查找检查点文件

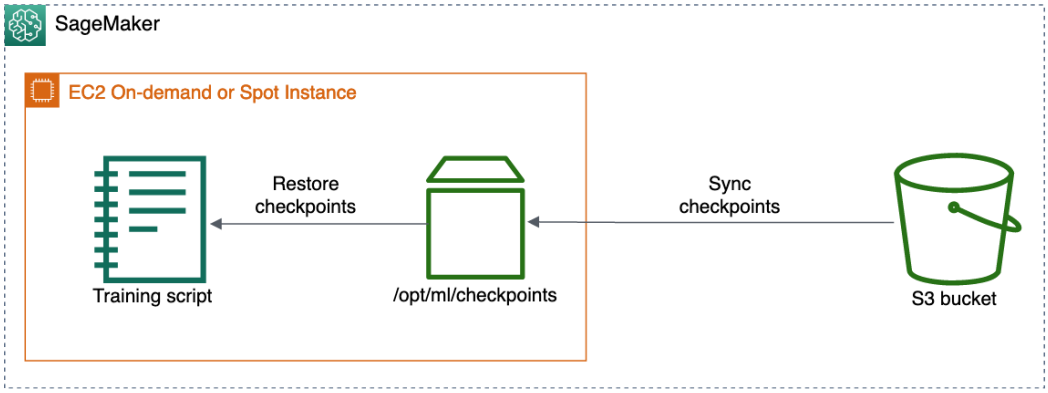
1. 登录到Amazon Web Services Management Console在处打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在左侧导航窗格中，选择训练任务.
3. 选择启用检查点的训练作业的链接以打开Job 设置.
4. 在存储库的Job 设置培训作业的页面，找到检查点配置部分。



5. 使用 S3 存储桶的链接访问检查点文件。

从检查点恢复训练

要从检查点恢复训练作业，请使用相同的checkpoint_s3_uri你在启用检查点 (p. 2088)部分。训练恢复后，此 S3 存储桶中的检查点将恢复到checkpoint_local_path在新培训工作的每个实例中。确保 S3 存储桶与当前 SageMaker 会话位于同一区域。



检查点的注意事项

在 SageMaker 中使用检查点时注意以下事项。

- 为避免在使用多个实例的分布式训练中覆盖，必须在训练脚本中手动配置检查点文件名和路径。高级 SageMaker 检查点配置指定了单个 Amazon S3 位置，而无需额外的后缀或前缀来标记多个实例的检查点。
- SageMaker Python SDK 不支持检查点频率的高级配置。要控制检查点频率，请使用框架的模型保存函数或检查点回调来修改训练脚本。
- 如果您使用 SageMaker 调试器和 SageMaker 分布的 SageMaker 检查点并且面临问题，请参阅以下页面了解故障排除和注意事项。
  - Amazon SageMaker 调试程序的注意事项 (p. 1936)
  - 数据并行排查 (p. 1996)
  - 模型并行疑难解 (p. 2040)

通过增强清单文件，将数据集元数据提供给训练作业

要在训练作业中随您的数据集包含元数据，请使用增强清单文件。使用增强清单文件时，您的数据集必须存储在 Amazon Simple Storage Service (Amazon S3) 中，并且必须配置训练作业以使用存储在该位置的数据集。您可以为一个或多个 `Channel` 指定此数据集的位置和格式。增强清单只能支持管道输入模式。请参阅部分InputMode在`Channel`了解更多关于管道输入模式的信息。

指定频道的参数时，需要指定文件的路径，称为`S3Uri`. 亚马逊 SageMaker 根据指定的`S3DataType`在`S3DataSource`. `AugmentedManifestFile` 选项定义随输入数据包括元数据的清单格式。在您有标记过的数据时，使用增强清单文件是一种替代的预处理方法。对于使用标记过数据的训练作业，您通常需要预处理数据集，从而在训练前将输入数据与元数据组合起来。如果您的训练数据集较大，预处理的时间可能会较长，成本高昂。

增强清单文件格式

增强清单文件必须采用 JSON 行格式。在 JSON 行格式中，文件的每一行是一个完整的 JSON 对象，后跟换行分隔符。

在训练过程中， SageMaker 分析每行 JSON 行，并将其部分或全部属性发送到训练算法。您可以使用 `CreateTrainingJob` API 的 `AttributeNames` 参数指定要传递哪些属性内容以及传递的顺序。这些区域有：`AttributeNames`参数是属性名称的有序列表 SageMaker 在 JSON 对象中查找用作训练输入。

例如，如果您为 `AttributeNames` 列出 `["line", "book"]`，则输入数据必须按指定顺序包括 `line` 和 `book` 的属性名称。对于本示例，以下增强清单文件内容有效：

```
{"author": "Herman Melville", "line": "Call me Ishmael", "book": "Moby Dick"}
{"line": "It was love at first sight.", "author": "Joseph Heller", "book": "Catch-22"}
```

SageMaker 会忽略未列出的属性名称，即使这些属性位于列出的属性之前、之后或之间。

使用增强清单文件时，请遵循以下准则：

- 在 AttributeNames 参数中列出的属性顺序确定了在训练作业中传递到算法的属性顺序。
- 列出的AttributeNames可以是 JSON 行中的所有属性的子集。 SageMaker 忽略文件中未列出的属性。
- 您可在 AttributeNames 中指定 JSON 格式允许的任意数据类型，包括文本、数字、数组或对象。
- 要包括 S3 URI 作为属性名称，请将后缀 -ref 添加到其中。

如果属性名称包含后缀 -ref，则属性的值必须是指向数据文件的 S3 URI 并且训练作业可以访问该 URI。例如，如果AttributeNames包含["image-ref", "is-a-cat"]中，以下示例显示有效的增强清单文件：

```
{"image-ref": "s3://mybucket/sample01/image1.jpg", "is-a-cat": 1}
{"image-ref": "s3://mybucket/sample02/image2.jpg", "is-a-cat": 0}
```

如果是此清单文件的第一行 JSON， SageMaker 检索image1.jpg来自的文件s3://mybucket/sample01/以及的字符串表示形式is-a-cat属性"1"对于图像分类。

> Tip
>
> 要创建增强清单文件，请使用 Amazon SageMaker Ground Truth 并创建标记作业。有关标记作业的输出的更多信息，请参阅输出数据 (p. 414).

流式处理增强清单文件数据

增强清单格式使您可以使用文件在管道模式下进行训练，而无需创建 RecordIO 文件。您需要将训练通道和验证通道指定为 CreateTrainingJob 请求的 InputDataConfig 参数的值。只有使用管道输入模式的通道才支持增强清单文件。对于每个通道，数据提取自其增强清单文件，并通过通道的指定管道（按顺序）流式传输到算法。管道模式使用先进先出 (FIFO) 方法，因此记录按照排队的顺序处理。有关管道输入模式的信息，请参阅Input Mode.

具有 "-ref" 后缀的属性名称指向预先格式化的二进制数据。在某些情况下，算法知道如何解析数据。在另一些情况下，您可能需要包装数据，针对算法来分隔记录。如果算法与 RecordIO 格式数据兼容，请为 RecordWrapperType 指定 RecordIO 以解决此问题。如果算法不与 RecordIO 格式兼容，请为 RecordWrapperType 指定 None 并确保针对您的算法正确解析了数据。

使用 ["image-ref", "is-a-cat"] 示例，如果您使用 RecordIO 包装，则将以下数据流发送到队列：

```
recordio_formatted(s3://mybucket/foo/
image1.jpg)recordio_formatted("1")recordio_formatted(s3://mybucket/bar/
image2.jpg)recordio_formatted("0")
```

未使用 RecordIO 格式包装的图像将通过相应的流式传输is-a-cat属性值作为一条记录。由于算法可能没有正确的分隔图像和属性，这可能会导致问题。有关使用增强清单文件进行映像分类的更多信息，请参阅使用扩增清单图像格式进行训练.

通常使用增强清单文件和管道模式时，EBS 卷的大小限制不适用。这包括原本必须位于 EBS 卷大小限制内的设置，例如：S3DataDistributionType .有关管道模式以及如何使用该模式的更多信息，请参阅使用您自己的训练算法-输入数据配置.

使用增强清单文件（控制台）

要完成此过程，您需要：

- 存储了增强清单文件的 S3 存储桶的 URL。
- 将在增强清单文件中列出的数据存储到 S3 存储桶中。
- 您要存储作业输出的 S3 存储桶的 URL。

在训练作业中使用增强清单文件（控制台）

1. 打开亚马逊 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中，选择 Training (训练)，然后选择 Training jobs (训练作业)。
3. 选择 Create training job (创建训练作业)。
4. 为训练作业提供一个名称。该名称在Amazon中的地区Amazonaccount. 名称可以包括 1 到 63 个字符。有效字符：a-z、A-Z、0-9 和 . : + = @ _ % - （连字符）。
5. 选择要使用的算法。有关支持的内置算法的信息，请参阅使用 Amazon SageMaker 内置算法或预训练模型 (p. 787)。如果您希望使用自定义算法，请确保该算法与管道模式兼容。
6. （可选）对于 Resource configuration (资源配置)，请接受默认值，如果要减少计算时间，请增加资源消耗。
   a. （可选）对于 Instance type (实例类型)，选择您要使用的 ML 计算实例类型。在大多数情况下 ml.m4.xlarge 便足够。
   b. 对于 Instance count (实例计数)，请使用默认值 `1`。
   c. （可选）对于 Additional volume per instance (GB) (每个实例的附加卷 (GB))，请选择您要预配置的 ML 存储卷的大小。在大部分情况下，您可以使用默认值 `1`。如果您使用大数据集，请使用较大的大小。
7. 为训练数据集提供有关输入数据的信息。
   a. 对于 Channel name (通道名称)，请接受默认值 (**`train`**) 或者输入更有意义的名称，例如 **`training-augmented-manifest-file`**。
   b. 对于 InputMode (输入模式)，请选择 Pipe (管道)。
   c. 对于 S3 data distribution type (S3 数据分配类型)，请选择 FullyReplicated。进行增量训练时，完全复制会导致各个 ML 计算实例使用扩展数据集的完整副本。对于基于神经的算法，例如 神经主题模型 (NTM) 算法 (p. 1671)，请选择 `ShardedByS3Key`。
   d. 如果在增强清单文件中指定的数据未压缩，请将 Compression type (压缩类型) 设置为 None (无)。如果使用 gzip 压缩了数据，请将其设置为 Gzip。
   e. （可选）对于 Content type (内容类型)，请指定相应的 MIME 类型。内容类型 是多用途 Internet 邮件扩展 (MIME) 类型的数据。
   f. 对于 Record wrapper (记录包装程序)，如果在增强清单文件中指定的数据集以 RecordIO 格式保存，则选择 RecordIO。如果您的数据集未另存为 RecordIO 格式的文件，请选择 None (无)。
   g. 对于 S3 data type (S3 数据类型)，选择 AugmentedManifestFile。
   h. 对于 S3 location (S3 位置)，提供您存储增强清单文件的存储桶的路径。
   i. 对于 AugmentedManifestFile attribute names (AugmentedManifestFile 属性名称)，请指定您要使用的属性的名称。属性名称必须存在于增强清单文件中，并且区分大小写。
   j. （可选）要添加更多属性，请选择 Add row（添加行）并为各个属性指定其他属性名称。
   k. （可选）要调整属性名称的顺序，请选择名称旁边的上移或下移按钮。使用增强清单文件时，指定属性名称的顺序非常重要。
   l. 选择完成。
8. 对于 Output data configuration (输出数据配置)，请提供以下信息：
   a. 对于 S3 location (S3 位置)，键入存储输出数据的 S3 存储桶的路径。
   b. （可选）您可以使用您的 Amazon Key Management Service (Amazon KMS) 加密密钥来加密静态输出数据。对于 Encryption key (加密密钥)，请提供密钥 ID 或其 Amazon 资源编号 (ARN)。有关更多信息，请参阅 KMS 托管的加密密钥。

9. （可选）对于 Tags (标签)，向训练作业添加一个或多个标签。一个标签您可以定义并分配到的元数据Amazon资源的费用。在这种情况下，您可以使用标签来帮助您管理训练作业。标签包含由您定义的键和值。例如，您可能希望创建一个标签，使用 **`Project`** 作为键，其值引用与训练作业相关的项目，例如 **`Home value forecasts`**。
10. 选择创建训练作业. SageMaker 创建并运行训练作业。

在培训工作完成之后 SageMaker 将模型工件存储在您提供的路径的存储桶中输出路径中的输出数据配置字段中返回的子位置类型。要部署模型以获取预测，请参阅第 5 步：将模型部署到 Amazon EC2 (p. 68)。

使用增强清单文件 (API)

以下显示了如何使用增强清单文件训练模型。使用 SageMaker 高级 Python 库：

```
import sagemaker

# Create a model object set to using "Pipe" mode.
model = sagemaker.estimator.Estimator(
    training_image,
    role,
    instance_count=1,
    instance_type='ml.p3.2xlarge',
    volume_size = 50,
    max_run = 360000,
    input_mode = 'Pipe',
    output_path=s3_output_location,
    sagemaker_session=session
)

# Create a train data channel with S3_data_type as 'AugmentedManifestFile' and attribute
 names.
train_data = sagemaker.inputs.TrainingInput(
    your_augmented_manifest_file,
    distribution='FullyReplicated',
    content_type='application/x-recordio',
    s3_data_type='AugmentedManifestFile',
    attribute_names=['source-ref', 'annotations'],
    input_mode='Pipe',
    record_wrapping='RecordIO'
)

data_channels = {'train': train_data}

# Train a model.
model.fit(inputs=data_channels, logs=True)
```

在培训工作完成之后 SageMaker 将模型工件存储在您提供的路径的存储桶中输出路径中的输出数据配置字段中返回的子位置类型。要部署模型以获取预测，请参阅第 5 步：将模型部署到 Amazon EC2 (p. 68)。

使用 Amazon CloudWatch 指标监控和分析训练作业

Amazon SageMaker 训练作业是一个迭代过程，此过程通过提供训练数据集中的示例来指导模型进行预测。通常情况下，训练算法计算几个指标，例如训练错误和预测准确度。这些指标有助于诊断模型的学习效果是否良好，以及针对未看到的数据进行预测是否将实现很好的泛化。训练算法将这些指标的值写入到日志中，而 SageMaker 将这些值实时监控这些值并将它们发送到 Amazon CloudWatch。要分析训练作业的绩效，您可以在 CloudWatch 中查看这些指标的图表。当训练作业已完成时，您还可以获得它通过调用 `DescribeTrainingJob` 操作在其最终迭代中计算的度量值的列表。

Note

Amazon CloudWatch 支持高精度定制指标，其最好的分辨率是 1 秒钟。但是，分辨率越好，CloudWatch 指标的使用寿命就越短。对于 1 秒的频率分辨率，CloudWatch 指标可用期为 3 小时。有关 CloudWatch 指标的分辨率和使用期限的更多信息，请参阅GetMetricStatistics中的Amazon CloudWatch API 参考.

Tip

如果您想以更精细的分辨率分析训练工作，最小至 100 毫秒（0.1 秒）的粒度，并随时将训练指标无限期存储在 Amazon S3 中以进行自定义分析，请考虑使用Amazon SageMaker 调试程序. SageMaker Debug 提供了自动检测常见训练问题的内置规则；它可以检测硬件资源利用率问题（例如 CPU、GPU 和 I/O 瓶颈）和非融合模型问题（例如过度拟合、消失渐变和爆炸张量）。SageMaker 调试器还通过 Studio 及其性能分析报告提供可视化效果。要探索调试器的可视化效果，请参阅SageMaker 调试器见解仪表板演练、调试器性能分析报告演练, 和使用 smDebug 客户端库分析数据.

主题

- 定义训练指标 (p. 2094)
- 监视训练 Job 指标（CloudWatch 控制台）(p. 2096)
- 监视训练 Job 指标（SageMaker 控制台）(p. 2097)
- 例如：查看训练和验证曲线 (p. 2099)

定义训练指标

SageMaker 会自动解析训练作业日志并将训练指标发送到 CloudWatch。默认情况下，SageMaker 会发送中列出的系统资源利用率SageMaker 作业和端点指标. 如果您希望 SageMaker 解析日志并将自己算法的训练作业中的自定义指标发送到 CloudWatch，则需要在配置 SageMaker 培训任务请求时传递指标名称和正则表达式来指定指标定义。

您可以使用 SageMaker 控制台指定要跟踪的指标：Python 开发工具包或低级别 SageMaker API。

如果您使用的是自己的算法，请执行以下操作：

- 确保算法将要捕获的指标写入日志中。
- 定义一个正则表达式，以准确地搜索日志来捕获要发送到 CloudWatch 的指标值。

例如，假设您的算法发出以下指标来发出训练错误和验证错误指标：

```
Train_error=0.138318;  Valid_error=0.324557;
```

如果您想在 CloudWatch 中监视这两个指标，则指标定义的字典应类似如下：

```
[
    {
        "Name": "train:error",
        "Regex": "Train_error=(.*?);"
    },
    {
        "Name": "validation:error",
        "Regex": "Valid_error=(.*?);"
    }
]
```

在正则表达式中`train:error`在上面示例中定义的指标，正则表达式的第一部分找到确切的文本“Train_error=”，表达式找到确切的文本“Train_error=”`(.*?);`捕获任何字符，直到出现第一个分号字符。在此表达式中，括号告诉正则表达式捕获括号内的内容，`.`表示任何字符，`*`表示零或更多个字符，`?`表示仅捕获到遇到`;`字符的第一个实例为止。

使用 SageMaker Python 开发工具包定义指标

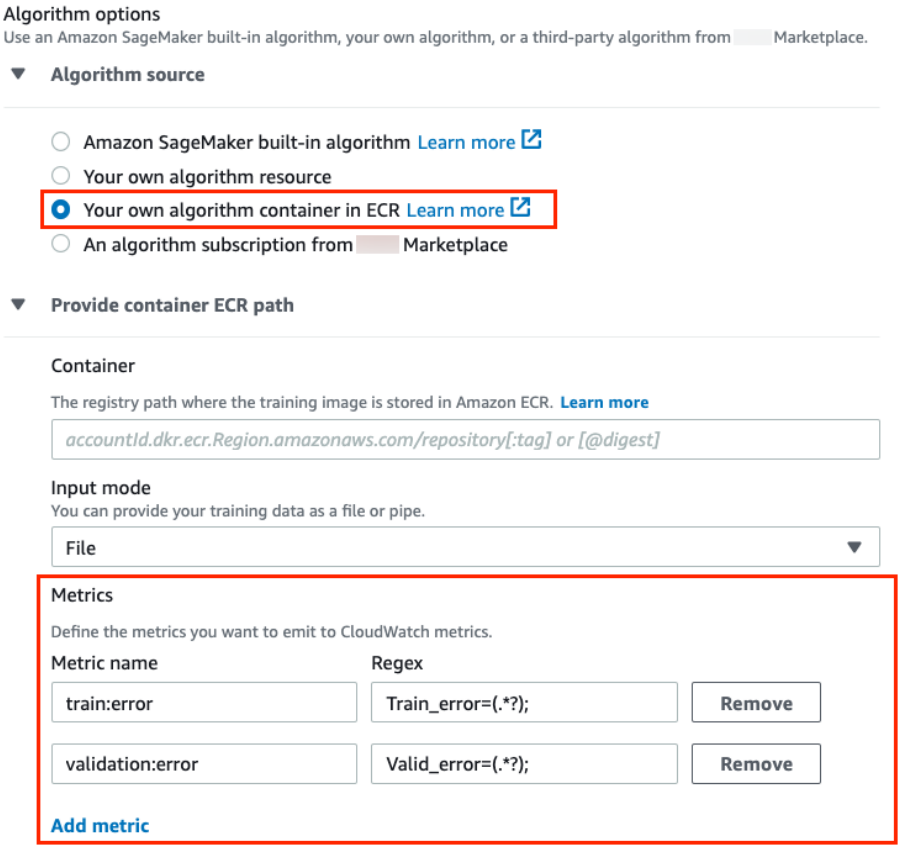
通过将指标名称和正则 CloudWatch 达式的列表指定为`metric_definitions`初始化时的参数`Estimator`对象。例如，如果您要监视`train:error`和`validation:error`CloudWatch 中的指标，`Estimator`初始化将类似于以下示例：

```
import sagemaker
from sagemaker.estimator import Estimator

estimator = Estimator(
    image_uri="your-own-image-uri",
    role=sagemaker.get_execution_role(),
    sagemaker_session=sagemaker.Session(),
    instance_count=1,
    instance_type='ml.c4.xlarge',
    metric_definitions=[
       {'Name': 'train:error', 'Regex': 'Train_error=(.*?);'},
       {'Name': 'validation:error', 'Regex': 'Valid_error=(.*?);'}
    ]
)
```

有关使用训练的更多信息Amazon SageMaker Python 开发工具包估计器，请参阅Sageaker Python 开发工具包（位于 GitHub 上）。

使用 SageMaker 控制台定义指标

如果选择ECR 中你自己的算法容器选项作为 SageMaker 控制台中的算法源，当您创建训练作业时，将量度定义添加到指标部分。下面的屏幕截图显示了添加示例指标名称和相应的正则表达式后应该如何看待。



使用低级别 SageMaker API 定义指标

通过在`MetricDefinitions`字段中的`AlgorithmSpecification`你传递给的输入参数`CreateTrainingJob`operation. 例如，如果您要监视`train:error`和`validation:error`CloudWatch中的指标，`AlgorithmSpecification`将类似于以下示例：

```
"AlgorithmSpecification": {
    "TrainingImage": your-own-image-uri,
    "TrainingInputMode": "File",
    "MetricDefinitions" : [
        {
            "Name": "train:error",
            "Regex": "Train_error=(.*?);"
        },
        {
            "Name": "validation:error",
            "Regex": "Valid_error=(.*?);"
        }
    ]
}
```

有关使用低级别 SageMaker API 定义和运行训练作业的更多信息，请参阅`CreateTrainingJob`.

监视训练 Job 指标（CloudWatch 控制台）

您可以在 CloudWatch 控制台中实时监视训练作业发出的指标。

监视训练作业指标（CloudWatch 控制台）

1. 从打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch.
2. 选择 Metrics (指标)，然后选择 /aws/sagemaker/TrainingJobs。
3. 选择 TrainingJobName。
4. 在 All metrics (所有指标) 选项卡上，选择您要监控的训练指标的名称。
5. 在绘成图表的指标选项卡上，配置图表选项。有关使用 CloudWatch 图形的更多信息，请参阅绘制指标图形中的Amazon CloudWatch 用户指南.

监视训练 Job 指标（SageMaker 控制台）

您可以使用 SageMaker 控制台实时监视训练作业发出的指标。

监视训练作业指标（SageMaker 控制台）

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker.
2. 选择 Training jobs (训练作业)，然后选择您要查看其指标的训练作业。
3. 选择 TrainingJobName。
4. 在 Monitor (监视) 部分，您可以查看实例使用率和算法指标图。

Monitor
Access logs for debugging and progress reporting. View metrics to set alarms, send notifications, or take actions. Learn more
View algorithm metrics
View logs
View instance metrics
2019-01-24 (10:33:57) -
MemoryUtilization
CPUUtilization
DiskUtilization
GPUUtilization
GPUMemoryUtilization
validation:accuracy
train:progress
train:throughput
train:accuracy
validation:cross_entropy
train:cross_entropy

例如：查看训练和验证曲线

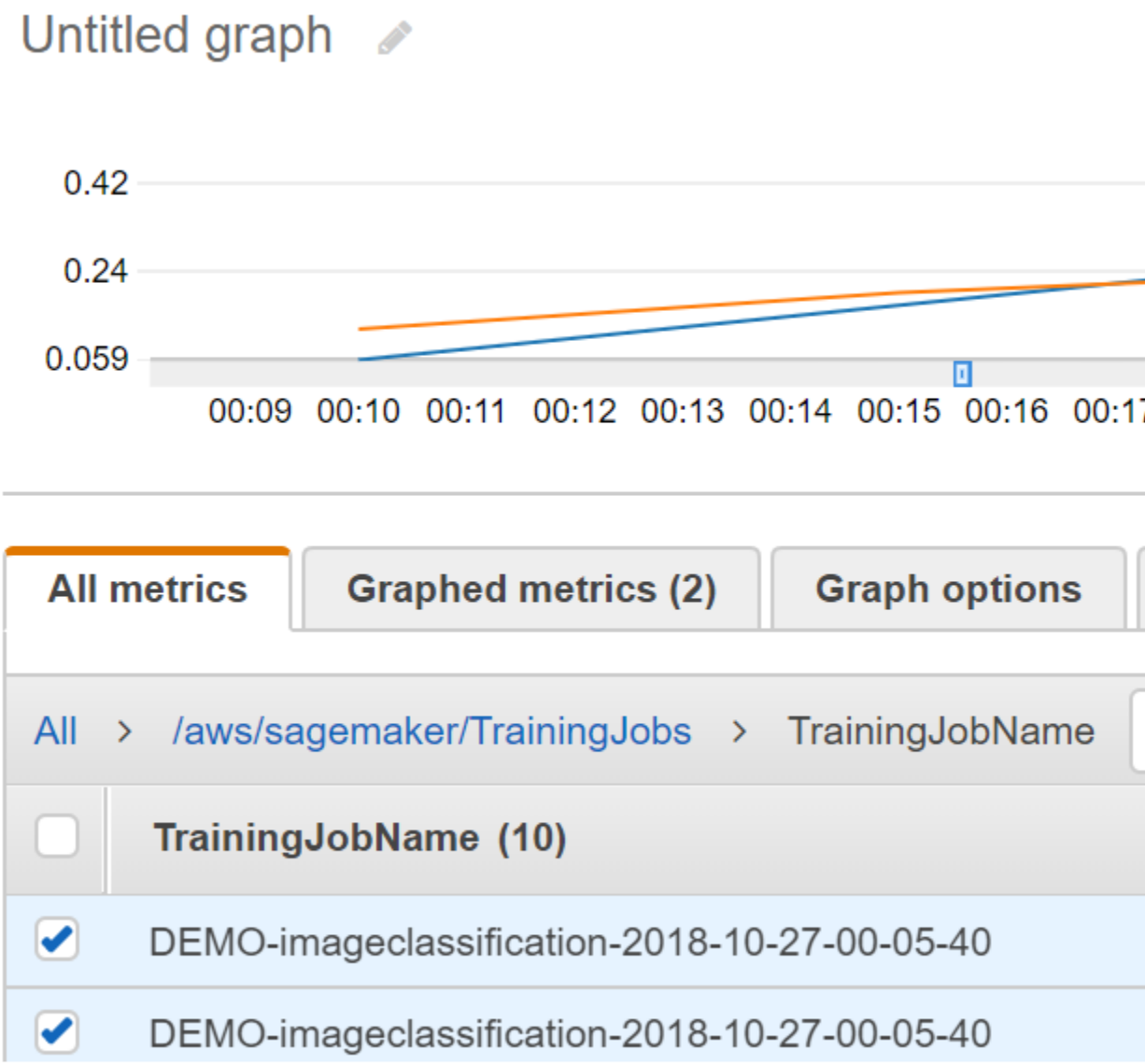
通常情况下，您将将训练模型所依据的数据拆分为训练和验证数据集。您使用训练集来训练用于对训练数据集进行预测的模型参数。然后，通过计算验证集的预测结果来测试模型预测的效果。一种常见的分析训练作业性能的方法是对照验证曲线绘制训练曲线。

查看训练集和验证集的准确性随时间变化的图形可帮助您改进模型性能。例如，如果训练准确性随时间持续提高，但在某个时间点验证准确性开始下降，则您可能过拟合您的模型。要解决此问题，可以对模型进行调整，例如提高正则化水平。

在此示例中，您可以使用图片分类-全面培训示例中的示例笔记本您的 SageMaker 笔记本实例的部分。如果您没有 SageMaker 笔记本实例，请通过执行以下操作：第 1 步：创建 Amazon SageMaker 笔记本实例 (p. 59). 如果您愿意，可以遵循 GitHub 上示例笔记本中的端到端多类图像分类示例。您还需要 Amazon S3 存储桶来存储训练数据和输出模型。

查看训练和验证错误曲线

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker.
2. 选择 Notebooks (笔记本)，然后选择 Notebook instances (笔记本实例)。
3. 选择要使用的笔记本实例，然后选择 Open (打开)。
4. 在笔记本实例的控制面板上，选择 SageMaker Examples (SageMaker 示例)。
5. 展开Amazon 算法简介部分，然后选择使用旁边图像分类-全训练 .iPynb.
6. 选择创建副本. SageMaker 创建一个可编辑的副本图像分类-全训练 .iPynb笔记本实例中的笔记本。
7. 将笔记本中的所有单元格运行到推理部分。您不需要部署终端节点或获取此示例的推论。
8. 在训练作业开始后，从打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch.
9. 选择 Metrics (指标)，然后选择 /aws/sagemaker/TrainingJobs。
10. 选择 TrainingJobName。
11. 在 All metrics (所有指标) 选项卡上，为您在笔记本中创建的训练作业选择 train:accuracy 和 validation:accuracy 指标。
12. 在图表中，选择一个区域来放大指标值。您应看到类似于以下示例的信息。


Untitled graph
0.42
0.24
0.059
00:09
00:10
00:11
00:12
00:13
00:14
00:15
00:16
All metrics
Graphed metrics (2)
Graph options
All
/aws/sagemaker/TrainingJobs
TrainingJobName
TrainingJobName (10)
DEMO-imageclassification-2018-10-27-00-05-40
DEMO-imageclassification-2018-10-27-00-05-40

部署推理模型

构建并训练模型之后，您可以部署这些模型，以便通过两种方式之一来获取预测：

- 要设置持久性终端节点以便从模型获取预测，请使用 Amazon SageMaker 托管服务。有关如何将模型部署到的示例 SageMaker 托管服务，请参阅创建终端节点并部署模型 (p. 2103).

  或者，如果您愿意，请观看以下视频教程：

  使用 Amazon SageMaker 将 ML 模型批量部署到生产环境
- 要获取整个数据集的预测，请使用 SageMaker 批量转换。有关使用部署模型的概述 SageMaker 批量转换，请参阅使用批量转换 (p. 2229).

  有关如何使用批量转换部署模型的示例，请参阅 （可选）使用 Batch 变换进行预测 (p. 69)。

  或者，如果您愿意，请观看以下视频教程：

  使用 Amazon SageMaker 将 ML 模型批量部署到生产环境

先决条件

以下主题假设您已构建和训练了一个或多个机器学习模型，并已准备好部署它们。如果您是新手 SageMaker 并且尚未完成这些先决条件任务，请完成开始使用 Amazon SageMaker (p. 30)教程让自己熟悉一个例子 SageMaker 管理数据科学过程以及如何处理模型部署。有关训练模型的更多信息，请参阅。训练模型 (p. 783).

您需要做什么？

在部署机器学习模型时，SageMaker 提供了用于管理资源和优化推理性能的功能。有关使用推理管道、使用 Neo 编译和部署模型、Elastic Inference 以及自动模型伸缩的指南，请参阅以下主题。

- 要管理数据处理和实时预测，或者在管道中处理批量转换，请参阅主机模型以及预处理逻辑作为一个端点后面的串行推理管道 (p. 2131)。
- 如果要在 inf1 实例上部署模型，请参阅使用 Neo 优化模型性能 (p. 2325)。
- 要训练一次 TensorFlow、Apache MXNet、PyTorch、ONNX 以及 XGBoost 模型并优化它们以部署到 ARM、Intel 和 Nvidia 处理器上，请参阅 使用 Neo 优化模型性能 (p. 2325)。
- 在您不需要持久的终端节点时，要快速预处理整个数据集或者从经训练的模型获取大型数据集的推理，请参阅使用批量转换 (p. 2229)。
- 对部署作为的深度学习模型，有关提升吞吐量并缩短获取实时推理延迟的指南， SageMaker 为终端节点使用 GPU 实例的托管模型，请参阅使用 Amazon SageMaker Elastic Inference (EI) (p. 2387).
- 要动态调整预配置的实例数以响应工作负载更改，请参阅自动扩展 Amazon SageMaker 模型 (p. 2144)。
- 要使用共享服务容器创建可托管多个模型的终端节点，请参阅在一个端点后面的一个容器中托管多个模 (p. 2112)。
- 要在生产中测试多个模型，请参阅在生产中安全更新模型 (p. 2158)。

管理模型部署

有关管理模型部署的指南，包括监控、故障排查和最佳实践，以及有关与推理托管实例关联的存储的信息：

- 有关可用于监控模型部署的工具，请参阅监控 Amazon SageMaker (p. 2810)。
- 有关模型部署故障排查，请参阅Amazon 故障排除 SageMaker 模型部署 (p. 2166)。
- 有关模型部署最佳实践，请参阅最佳实践 (p. 2165)。
- 有关为不同大小托管实例提供的存储卷大小的更多信息，请参阅主机实例存储卷 (p. 2158)。

部署自己的推理代码

对于需要有关如何运行自己的推理代码的更高级指导的开发人员：

- 要运行您自己的推理代码托管服务，请参阅将您自己的推理代码用于托管服务 (p. 2435)。
- 要运行您自己的推理代码用于批量转换，请参阅将您自己的推理代码用于批量转换 (p. 2440)。

实时推理

实时推理非常适合您有实时、交互式、低延迟要求的推理工作负载。你可以将模型部署到 SageMaker 托管服务并获得可用于推断的终端节点。这些端点已完全托管并支持自动缩放（请参阅自动扩展 Amazon SageMaker 模型 (p. 2144)）。

主题

- 托管选项 (p. 2102)
- 自动扩展 Amazon SageMaker 模型 (p. 2144)
- 主机实例存储卷 (p. 2158)
- 在生产中安全更新模型 (p. 2158)
- 最佳实践 (p. 2165)
- 监控模型的数据和模型质量、偏见和可解释性 (p. 2167)
- 调用实时端节点 (p. 2213)

托管选项

以下主题介绍了可用的 SageMaker 实时托管选项以及如何设置、调用和删除每个托管选项。

主题

- 托管单个模型 (p. 2102)
- 在一个端点后面的一个容器中托管多个模 (p. 2112)
- 托管在一个终端后面使用不同容器的多个模型 (p. 2125)
- 主机模型以及预处理逻辑作为一个端点后面的串行推理管道 (p. 2131)

托管单个模型

创建、更新和删除托管单个模型的实时推理端点Amazon SDK for Python (Boto3)， SageMaker Python 开发工具包， SageMaker 控制台，或Amazon CLI.

主题

- 创建终端节点并部署模型 (p. 2103)

- 删除终端节点和资源 (p. 2108)

创建终端节点并部署模型

使用部署模型有多种选择 SageMaker 托管服务。您可以使用以编程方式部署模型Amazon开发工具包（例如，SDK for Python (Boto3)），SageMaker Python 开发工具包，Amazon CLI，或者你可以使用 SageMaker 控制台。这些区域有：AmazonSDK 是一个低级 API，支持 Java、C++、Go、JavaScript、Node.js、PHP、Ruby 和 Python 而 SageMaker Python 开发工具包是一个高级 Python API。以下文档演示了如何使用Amazon SDK for Python (Boto3)和 SageMaker Python 开发工具包。

使用部署模型 SageMaker 如果使用托管服务有三个步骤，Amazon SDK for Python (Boto3)，Amazon CLI，或者 SageMaker 控制台：

1. 创建 SageMaker SageMaker 中的模型。
2. 为 HTTPS 终端节点创建终端节点配置。
3. 创建 HTTPS 终端节点。

使用 SageMaker Python SDK 不要求您创建终端节点配置。因此，这包括两个步骤的过程：

1. 通过创建模型对象`Model`可以部署到 HTTPS 终端节点的类。
2. 使用模型对象的预构建创建 HTTPS 终端节点`deploy()`方法。

要使用继续执行的代码片段，请替换`#######`在示例代码中包含您自己的信息。

Note

- SageMaker 支持运行 (a) 多个模型，(b) 一个模型的多个变体，或 (c) 在同一端点上运行模型和变体的组合。模型变体可以引用同一模型推理容器（即运行相同的算法），但使用不同的模型构件（例如，基于其他超参数配置的不同模型权重值）。相比之下，两种不同的模型可能使用相同的算法，但专注于不同的业务问题或基本目标，并可能在不同的数据集上运行。
- 当你创建终端节点时，SageMaker 将 Amazon EBS 存储卷附加到托管该终端节点的每个 ML 计算实例。存储卷的大小取决于实例类型。有关实例类型列表 SageMaker 托管服务支持，请参阅Amazon服务限制. 有关存储卷的大小列表 SageMaker 附加到每个实例，请参阅主机实例存储卷 (p. 2158).
- 终端节点的作用域限于单个 Amazon 账户，并且终端节点不是公有的。该 URL 不包含账户 ID，但 SageMaker 通过调用者提供的身份验证令牌确定账户 ID。

  有关如何使用 Amazon API Gateway 的示例和Amazon Lambda要设置和部署可以从不在帐户范围内的客户端应用程序调用的 Web 服务，请参阅调用 SageMaker 使用 Amazon API Gateway 模型终端节点和Amazon Lambda中的AmazonMachine Learning 博客.

开始前的准备工作

在创建和部署 SageMaker 模型，找到并记下：

- AmazonAmazon S3 存储桶所在的区域
- 存储模型工件的 Amazon S3 URI 路径
- SageMaker 的 IAM 角色
- 包含推理代码的客户映像的 Docker Amazon ECR URI 注册表路径，或者支持和Amazon.

列表Amazon每个都可用的服务Amazon地区，请参阅区域地图和边缘网络. 请参阅创建 IAM 角色有关如何创建 IAM 角色的信息。

Important

存储模型构件的 Amazon S3 存储桶必须与您正在创建的模型位于同一区域中。

以下内容演示了如何以编程方式定义上述参数。

Amazon SDK for Python (Boto3)

导入 Boto3。定义您的Amazon区域。接下来，初始化低级 SageMaker 服务客户端对象使用Client类。这将使发送和接收请求变得容易Amazon服务。此外，定义Amazon赋予的角色 SageMaker 访问权限Amazon代表您提供的服务：

```
import boto3

# Specify your AWS Region
aws_region='<aws_region>'

# Create a low-level SageMaker service client.
sagemaker_client = boto3.client('sagemaker', region_name=aws_region)

# Role to give SageMaker permission to access Amazon services.
sagemaker_role= "arn:aws:iam::<region>:<account>:role/*"
```

前几行定义：

- sagemaker_client: 低级别 SageMaker 客户端对象，可以轻松发送和接收请求Amazon服务。
- sagemaker_role: 一个字符串变量，带 SageMaker IAM 角色 Amazon 资源名称 (ARN)。
- aws_region: 具有你的名字的字符串变量Amazon区域。

导入image_uris中的模块 SageMaker Python 开发工具包。使用retrieve()函数来检索与给定参数匹配的 Docker 映像的 Amazon ECR URI。同时提供框架名称或算法的名称 (framework字段) 以及框架或算法的版本 (version字段)。有关由提供的内置 Docker 映像列表Amazon，请参阅可用的 Deep Learning Containers 映像. 有关如何自带图像的更多信息，请参阅。使用您自己的推理代码 (p. 2435).

```
from sagemaker import image_uris

# Name of the framework or algorithm
framework='<framework>'
#framework='xgboost' # Example

# Version of the framework or algorithm
version = '<version-number>'
#version = '0.90-1' # Example

# Specify an AWS container image.
container = image_uris.retrieve(region=aws_region,
                                framework=framework,
                                version=version)
```

接下来，提供预训练模型的 Amazon S3 URI。此模型将用于创建 SageMaker Python 开发工具包Model.model将在下一步中执行对象。在此示例中，完整的 Amazon S3 URI 存储在字符串变量中model_url：

```
# Create a variable w/ the model S3 URI
# First, provide the name of your S3 bucket
s3_bucket = '<your-bucket-name>'

# Specify what directory within your S3 bucket your model is stored in
```

```
bucket_prefix = '<directory-name>'

# Replace with the name of your model artifact
model_filename = '<model-name>.tar.gz'


# Relative S3 path
model_s3_key = f'{bucket_prefix}/'+model_filename

# Combine bucket name, model file name, and relate S3 path to create S3 model URI
model_url = f's3://{s3_bucket}/{model_s3_key}'
```

SageMaker Python SDK

定义您的Amazon区域。此外，定义Amazon赋予的角色 SageMaker 访问权限Amazon代表您提供的服务：

```
# Specify your AWS Region
aws_region='<aws_region>'

# Role to give SageMaker permission to access Amazon services.
sagemaker_role= "arn:aws:iam::<region>:<account>:role/*"
```

前几行定义：

- `sagemaker_role`: 一个字符串变量，带 SageMaker IAM 角色 Amazon 资源名称 (ARN)。
- `aws_region`: 具有你的名字的字符串变量Amazon区域。

导入`image_uris`中的模块 SageMaker Python 开发工具包。使用`retrieve()`函数来检索与给定参数匹配的 Docker 映像的 Amazon ECR URI。同时提供框架名称或算法的名称 (`framework`字段）以及框架或算法的版本（`version`字段)。有关由提供的内置 Docker 映像列表Amazon，请参阅可用的 Deep Learning Containers 映像. 有关如何自带图像的更多信息，请参阅。使用您自己的推理代码 (p. 2435).

```
from sagemaker import image_uris

# Name of the framework or algorithm
framework='<framework>'
#framework='xgboost' # Example

# Version of the framework or algorithm
version = '<version-number>'
#version = '0.90-1' # Example

# Specify an AWS container image.
container = image_uris.retrieve(region=aws_region,
                                framework=framework,
                                version=version)
```

接下来，提供预训练模型的 Amazon S3 URI。此模型将用于创建 SageMaker Python 开发工具包`Model.model`将在下一步中执行对象。在此示例中，完整的 Amazon S3 URI 存储在字符串变量中`model_url`：

```
# Create a variable w/ the model S3 URI
# First, provide the name of your S3 bucket
s3_bucket = '<your-bucket-name>'

# Specify what directory within your S3 bucket your model is stored in
bucket_prefix = '<directory-name>'
```

```
# Replace with the name of your model artifact
model_filename = '<model-name>.tar.gz'


# Relative S3 path
model_s3_key = f'{bucket_prefix}/'+model_filename

# Combine bucket name, model file name, and relate S3 path to create S3 model URI
model_url = f's3://{s3_bucket}/{model_s3_key}'
```

创建模型

通过创建模型，你告诉 SageMaker 在哪里可以找到模型组件。

Amazon SDK for Python (Boto3)

在 中创建模型 SageMaker 和`CreateModel`. 指定以下内容：

- `ModelName`：模型的名称（在本例中，它存储为一个字符串变量，名为`model_name`)。
- `ExecutionRoleArn`：IAM 角色的 Amazon 资源名称 (ARN)，Amazon SageMaker 可代入此角色以访问模型构件和 Docker 映像以在 ML 计算实例上进行部署或批量转换作业。
- `PrimaryContainer`：主要 Docker 映像的位置，该映像包含推理代码、关联构件和自定义环境贴图（在部署模型以进行预测时，推理代码将使用此贴图）。

对于主容器，指定包含推理代码的 Docker 映像、构件（来自先前的培训）和推理代码在您部署预测模型时使用的自定义环境贴图。

```
model_name = '<The_name_of_the_model>'

#Create model
create_model_response = sagemaker_client.create_model(
    ModelName = model_name,
    ExecutionRoleArn = sagemaker_role,
    PrimaryContainer = {
        'Image': container,
        'ModelDataUrl': model_url,
    })
```

请参阅`CreateModel`中的描述 SageMaker API 参考指南了解 API 参数的完整列表。

SageMaker Python SDK

通过创建模型对象 SageMaker Python 开发工具包`Model`类。在后面的步骤中，您将使用此对象将模型部署到终端节点。

```
from sagemaker.model import Model

model = Model(image_uri=container,
              model_data=model_url,
              role=sagemaker_role)
```

创建终端节点配置

为 HTTPS 终端节点创建终端节点配置。指定生产中的一个或多个模型的名称（变体）和您想要的 ML 计算实例。 SageMaker 启动以托管每个生产变体。

Note

如果您专门使用 SageMaker Python 开发工具包。

在生产环境中托管模型时，您可以将终端节点配置为弹性扩展部署的 ML 计算实例。对于每个生产变体，您指定所要部署的 ML 计算实例数量。当您指定两个或更多实例时， SageMaker 在多个可用区中启动它们。这可确保连续可用性。SageMaker 管理实例的部署。

Amazon SDK for Python (Boto3)

使用创建端点配置`CreateEndpointConfig`. 亚马逊 SageMaker 托管服务使用此配置来部署模型。在配置中，您可以标识使用与创建的一个或多个模型。`CreateModel`，部署您希望 Amazon 的资源 SageMaker 预置。

以下示例演示了如何使用创建终端节点配置。Amazon SDK for Python (Boto3)：

```
import datetime
from time import gmtime, strftime

# Create an endpoint config name. Here we create one based on the date
# so it we can search endpoints based on creation time.
endpoint_config_name = '<endpoint-config-name>'

# The name of the model that you want to host. This is the name that you specified when
 creating the model.
model_name='<The_name_of_your_model>'


instance_type = '<instance-type>'
# instance_type='ml.m5.xlarge' # Example

endpoint_config_response = sagemaker_client.create_endpoint_config(
    EndpointConfigName=endpoint_config_name, # You will specify this name in a
 CreateEndpoint request.
    # List of ProductionVariant objects, one for each model that you want to host at
 this endpoint.
    ProductionVariants=[
        {
            "VariantName": "variant1", # The name of the production variant.
            "ModelName": model_name,
            "InstanceType": instance_type, # Specify the compute instance type.
            "InitialInstanceCount": 1 # Number of instances to launch initially.
        }
    ]
)

print(f"Created EndpointConfig: {endpoint_config_response['EndpointConfigArn']}")
```

在上述示例中，您可以指定以下密钥以下内容以下内容`ProductionVariants`字段：

- `VariantName`：生产变体的名称。
- `ModelName`：要托管的模型的名称。这是在创建模型时指定的名称。
- `InstanceType`：计算实例类型。请参阅`InstanceType`中的字段https://docs.amazonaws.cn/sagemaker/latest/APIReference/API_ProductionVariant.html和SageMaker 定价以获取支持的计算实例类型列表以及每个实例类型的定价。

SageMaker Python SDK

如果使用 SageMaker Python 开发工具包`Model`类。

部署模型

部署模型并创建 HTTPS 终端节点。

Amazon SDK for Python (Boto3)

向 SageMaker 提供端点配置。该服务会启动 ML 计算实例，并按照配置中的规定部署一个或多个模型。

拥有模型和终端节点配置后，请使用CreateEndpoint用于创建终端节点的 API。终端节点名称在Amazon您的区域Amazonaccount.

以下内容使用在请求中指定的终端节点配置创建终端节点。亚马逊 SageMaker 使用终端节点来配置资源和部署模型。

```
# The name of the endpoint. The name must be unique within an AWS Region in your AWS
 account.
endpoint_name = '<endpoint-name>'

# The name of the endpoint configuration associated with this endpoint.
endpoint_config_name='<endpoint-config-name>'

create_endpoint_response = sagemaker_client.create_endpoint(
                                            EndpointName=endpoint_name,
                                            EndpointConfigName=endpoint_config_name)
```

有关更多信息，请参阅 CreateEndpoint API。

SageMaker Python SDK

为终端节点定义名称。此为可选步骤。如果你不提供一个， SageMaker 将为你创建一个唯一的名字：

```
from datetime import datetime

endpoint_name = f"DEMO-{datetime.utcnow():%Y-%m-%d-%H%M}"
print("EndpointName =", endpoint_name)
```

使用模型对象的内置将模型部署到实时的 HTTPS 终端节点deploy()方法。在instance_type字段以及要在其上运行终端节点的实例的初始数量initial_instance_count字段：

```
initial_instance_count=<integer>
# initial_instance_count=1 # Example

instance_type='<instance-type>'
# instance_type='ml.m4.xlarge' # Example

model.deploy(
    initial_instance_count=initial_instance_count,
    instance_type=instance_type,
    endpoint_name=endpoint_name
)
```

删除终端节点和资源

删除终端节点以停止产生费用。

删除端节点

使用以编程方式删除终端节点Amazon SDK for Python (Boto3)，使用Amazon CLI，或者以交互方式使用 SageMaker 控制台。

SageMaker 释放在创建终端节点时部署的所有资源。删除终端节点不会删除终端节点配置或 SageMaker 模型。请参阅删除端点配置 (p. 2109)和删除模型 (p. 2111)了解有关如何删除终端节点配置的信息以及 SageMaker 模型。

Amazon SDK for Python (Boto3)

使用DeleteEndpoint用于删除终端节点的 API。将终端节点的名称指定为EndpointName字段中返回的子位置类型。

```
import boto3

# Specify your AWS Region
aws_region='<aws_region>'

# Specify the name of your endpoint
endpoint_name='<endpoint_name>'

# Create a low-level SageMaker service client.
sagemaker_client = boto3.client('sagemaker', region_name=aws_region)

# Delete endpoint
sagemaker_client.delete_endpoint(EndpointName=endpoint_name)
```

Amazon CLI

使用delete-endpoint命令删除终端节点。将终端节点的名称指定为endpoint-name标记。

```
aws sagemaker delete-endpoint --endpoint-name <endpoint-name>
```

SageMaker Console

使用以交互方式删除终端节点 SageMaker 控制台。

1. 在 SageMaker 控制台https://console.aws.amazon.com/sagemaker/导航菜单，选择推理.
2. 选择终端节点从下拉菜单中。在您中创建的终端节点列表Amazon账户将按名称、Amazon 资源名称 (ARN)、创建时间、状态和终端节点上次更新时间的时间戳显示。
3. 选择要删除的终端节点。
4. SELECT操作右上角的下拉按钮。
5. 请选择 Delete（删除）。

删除端点配置

使用以编程方式删除终端节点配置Amazon SDK for Python (Boto3)，使用Amazon CLI，或者以交互方式使用 SageMaker 控制台。删除终端节点配置不会删除使用此配置创建的终端节点。请参阅删除端节点 (p. 2108)有关如何删除终端节点的信息。

请勿删除处于活动状态的终端节点或正在更新或创建终端节点时使用的终端节点配置。如果删除处于活动状态或正在创建或更新的终端节点的终端节点配置，则可能无法了解终端节点正在使用的实例类型。

Amazon SDK for Python (Boto3)

使用DeleteEndpointConfig用于删除终端节点的 API。指定终端节点配置的名称以指定EndpointConfigName字段中返回的子位置类型。

```
import boto3
```

```
# Specify your AWS Region
aws_region='<aws_region>'

# Specify the name of your endpoint configuration
endpoint_config_name='<endpoint_name>'

# Create a low-level SageMaker service client.
sagemaker_client = boto3.client('sagemaker', region_name=aws_region)

# Delete endpoint configuration
sagemaker_client.delete_endpoint_config(EndpointConfigName=endpoint_config_name)
```

您可以选择使用DescribeEndpointConfig用于返回有关已部署模型（生产变体）名称的信息的API，例如模型的名称以及与该已部署模型关联的终端节点配置的名称。提供终端节点的名称，以供EndpointConfigName字段中返回的子位置类型。

```
# Specify the name of your endpoint
endpoint_name='<endpoint_name>'

# Create a low-level SageMaker service client.
sagemaker_client = boto3.client('sagemaker', region_name=aws_region)

# Store DescribeEndpointConfig response into a variable that we can index in the next
 step.
response = sagemaker_client.describe_endpoint_config(EndpointConfigName=endpoint_name)

# Delete endpoint
endpoint_config_name = response['ProductionVariants'][0]['EndpointConfigName']

# Delete endpoint configuration
sagemaker_client.delete_endpoint_config(EndpointConfigName=endpoint_config_name)
```

有关返回的其他响应元素的更多信息。DescribeEndpointConfig，请参阅DescribeEndpointConfig中的SageMaker API 参考指南.

Amazon CLI

使用delete-endpoint-config命令删除终端节点配置。将终端节点配置的名称指定为endpoint-config-name标记。

```
aws sagemaker delete-endpoint-config \
                        --endpoint-config-name <endpoint-config-name>
```

您可以选择使用describe-endpoint-config命令返回有关已部署模型（生产变体）名称的信息，例如模型的名称以及与该已部署模型关联的终端节点配置的名称。提供终端节点的名称，以供endpoint-config-name标记。

```
aws sagemaker describe-endpoint-config --endpoint-config-name <endpoint-config-name>
```

这将返回 JSON 响应。您可以复制和粘贴、使用 JSON 解析器或使用为 JSON 解析而构建的工具来获取与该终端节点关联的终端节点配置名称。

SageMaker Console

使用以交互方式删除终端节点配置 SageMaker 控制台。

1. 在 SageMaker 控制台https://console.aws.amazon.com/sagemaker/导航菜单，选择推理.

2. 选择端节点配置从下拉菜单中显示。在您中创建的终端节点配置列表Amazon账户将按名称、Amazon 资源名称 (ARN) 和创建时间显示。
3. 选择要删除的终端节点配置。
4. SELECT操作右上角的下拉按钮。
5. 请选择 Delete（删除）。

删除模型

删除您的 SageMaker 使用编程方式使用Amazon SDK for Python (Boto3)，使用Amazon CLI，或者以交互方式使用 SageMaker 控制台。删除集群 SageMaker 模型仅删除在 SageMaker 中创建的模型条目。删除模型不会删除模型构件、推理代码或在创建模型时指定的 IAM 角色。

Amazon SDK for Python (Boto3)

使用DeleteModel用于删除你的 API SageMaker 模型。对于，请指定模型的名称ModelName字段中返回的子位置类型。

```
import boto3

# Specify your AWS Region
aws_region='<aws_region>'

# Specify the name of your endpoint configuration
model_name='<model_name>'

# Create a low-level SageMaker service client.
sagemaker_client = boto3.client('sagemaker', region_name=aws_region)

# Delete model
sagemaker_client.delete_model(ModelName=model_name)
```

您可以选择使用DescribeEndpointConfig用于返回有关已部署模型（生产变体）名称的信息的API，例如模型的名称以及与该已部署模型关联的终端节点配置的名称。提供终端节点的名称，以供EndpointConfigName字段中返回的子位置类型。

```
# Specify the name of your endpoint
endpoint_name='<endpoint_name>'

# Create a low-level SageMaker service client.
sagemaker_client = boto3.client('sagemaker', region_name=aws_region)

# Store DescribeEndpointConfig response into a variable that we can index in the next
 step.
response = sagemaker_client.describe_endpoint_config(EndpointConfigName=endpoint_name)

# Delete endpoint
model_name = response['ProductionVariants'][0]['ModelName']
sagemaker_client.delete_model(ModelName=model_name)
```

有关返回的其他响应元素的更多信息。DescribeEndpointConfig，请参阅DescribeEndpointConfig中的SageMaker API 参考指南.

Amazon CLI

使用delete-model命令删除 SageMaker 模型。对于，请指定模型的名称model-name标记。

```
aws sagemaker delete-model \
```

```
                    --model-name <model-name>
```

您可以选择使用describe-endpoint-config命令返回有关已部署模型（生产变体）名称的信息，例如模型的名称以及与该已部署模型关联的终端节点配置的名称。提供终端节点的名称，以供endpoint-config-name标记。

```
aws sagemaker describe-endpoint-config --endpoint-config-name <endpoint-config-name>
```

这将返回 JSON 响应。您可以复制和粘贴、使用 JSON 解析器或使用为 JSON 解析而构建的工具来获取与该终端节点关联的模型的名称。

SageMaker Console

删除您的 SageMaker 以交互方式建模 SageMaker 控制台。

1. 在 SageMaker 控制台https://console.aws.amazon.com/sagemaker/选择导航菜单，选择推理.
2. 选择模型从下拉菜单中。在你身上创建的模型列表Amazon账户将按名称、Amazon 资源名称 (ARN) 和创建时间显示。
3. 选择要删除的模型。
4. SELECT操作右上角的下拉按钮。
5. 选择 Delete。

在一个端点后面的一个容器中托管多个模

要创建可托管多个模型的终端节点，请使用多模型终端节点。多模型终端节点提供了经济高效的可扩展解决方案，可用于部署数量非常多的模型。它们使用支持的共享服务容器来托管多个模型。与使用单模型终端节点相比，这可以提高终端节点利用率，从而降低托管成本。它还可以减少部署开销，这是因为 Amazon SageMaker 管理在内存中加载模型，并根据其流量模式扩展模型。

多模型终端节点还可在模型之间实现内存资源的时间共享。当模型的大小和调用延迟非常相似时，这种方法效果最佳。在这种情况下，多模型终端节点可以有效地在所有模型中使用实例。如果模型具有明显较高的每秒事务数 (TPS) 或延迟要求，我们建议在专用终端节点上托管这些模型。多模型终端节点还非常适合可容忍在调用不常使用的模型时偶尔发生的冷启动相关延迟的情况。

多模型终端节点支持 A/B 测试。他们与 Auto Scaling 配合使用Amazon PrivateLink. 您可以将启用多模型的容器与串行推理管道一起使用，但推理管道中只能包含一个启用多模型的容器。您不能将启用多模型的容器与 Amazon Elastic Inference 结合使用。

您可以使用Amazon SDK for Python (Boto)或者使用 SageMaker 控制台来创建多模型终端节点。通过集成多模型服务器库，可以使用多模型终端节点和自定义构建容器。

主题

- 支持的算法和框架 (p. 2113)
- 用于多模型终端节点的示例笔记本 (p. 2113)
- 多模型终端节点的工作原理 (p. 2113)
- 设置 SageMaker 多模型终端节点模型缓存行 (p. 2114)
- 多模型终端节点部署的实例建议 (p. 2114)
- 创建多模型终端节点 (p. 2115)
- 调用多模型终端节点 (p. 2118)
- 添加或删除模型 (p. 2120)
- 使用多模型服务器构建您自己的容器 (p. 2120)

- 多模型终端节点安全性 (p. 2124)
- 多模型终端节点部署的 CloudWatch 指标 (p. 2125)

支持的算法和框架

以下算法和框架的推理容器支持多模型端点：

- XGBoost 算法 (p. 1731)
- K 最近邻 (k-NN) 算法 (p. 1644)
- 线性学习器算法 (p. 1657)
- 随机森林砍伐 (RCF) 算法 (p. 1704)
- 将 TensorFlow 与 Amazon SageMaker 结合使用 (p. 26)
- 将 Scikit-learn 与 Amazon SageMaker 结合使用 (p. 25)
- 将 Apache MXNet 与 Amazon SageMaker 结合使用 (p. 11)
- 将 PyTorch 与 Amazon SageMaker 结合使 (p. 22)

要使用任何其他框架或算法，请使用 SageMaker 推理工具包构建支持多模型终端节点的容器。有关信息，请参阅使用多模型服务器构建您自己的容器 (p. 2120)。

用于多模型终端节点的示例笔记本

有关使用 SageMaker 将多个 XGBoost 模型部署到终端节点的示例笔记本，请参阅多模型终端节点 xgBoost 示例笔记本. 有关演示如何在 SageMaker 中设置和部署支持多模型终端节点的自定义容器的示例笔记本，请参阅多模型终端节点 BYOC 示例笔记本. 有关如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 在您创建笔记本实例并打开该实例之后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。多模型终端节点笔记本位于 ADVANCED FUNCTIONALITY (高级功能) 部分。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

多模型终端节点的工作原理

SageMaker 管理容器内存中托管在多模型终端节点上的模型的生命周期。而不是下载所有模型从 Amazon S3 存储桶创建终端节点时，SageMaker 会在调用终端节点时动态加载它们。当 SageMaker 收到特定模型的调用请求时，它将执行以下操作：

1. 将请求路由到终端节点后面的实例。
2. 将模型从 S3 存储桶下载到该实例的存储卷中。
3. 将模型加载到该实例上的容器内存中。如果模型已加载到容器内存中，则调用速度会更快，因为 SageMaker 不需要下载和加载模型。

SageMaker 继续将模型的请求路由到已将模型加载到的实例。但是，如果模型收到多个调用请求，并且存在多模型终端节点的其他实例，则 SageMaker 会将一些请求路由到另一个实例以容纳流量。如果尚未在第二个实例上加载模型，则模型将下载到该实例的存储卷中，并加载到容器的内存中。

当实例的内存利用率较高，并且 SageMaker 需要将另一个模型加载到内存中时，它会从该实例的容器中卸载未使用的模型，以确保有足够的内存来加载模型。卸载的模型将保留在实例的存储卷上，并且稍后可加载到容器的内存中，而无需再次从 S3 存储桶进行下载。如果实例的存储卷达到其容量，则 SageMaker 将从存储卷中删除所有未使用的模型。

要删除模型，请停止发送请求并将其从 S3 存储桶中删除。SageMaker 在服务容器中提供多模型终端节点功能。在多模型终端节点中添加和删除模型不需要更新终端节点本身。要添加一个模型，请将该模型上传到 S3 存储桶并调用它。无需更改代码即可使用它。

在更新多模型终端节点时，流量将定向到更新后的终端节点中的实例，因此终端节点上的调用请求可能会遇到更高延迟。

设置 SageMaker 多模型终端节点模型缓存行

默认情况下，多模型端点将经常使用的模型缓存在内存和磁盘上，以提供低延迟推断。只有当容器用尽内存或磁盘空间以容纳新目标模型时，才会从磁盘中卸载和/或删除缓存模型。

您可以通过设置参数更改多模型端点的缓存行为，并显式启用或禁用模型缓存`ModelCacheSetting`你打电话的时候create_model.

我们建议设置`ModelCacheSetting`将参数转为`Disabled`适用于无法从模型缓存中受益的使用案例。例如，当需要从端点提供大量模型但每个模型只调用一次（或者很少调用）时。对于此类用例，请设置`ModelCacheSetting`将参数转为`Disabled`允许每秒更高的事务 (TPS)`invoke_endpoint`请求与默认缓存模式进行比较。在这些使用案例中，TPS 较高的原因是 SageMaker 在`invoke_endpoint`请求:

- 从内存中异步卸载模型，并在调用模型后立即将其从磁盘中删除。
- 为在推理容器中下载和加载模型提供了更高的并发性。并发是容器实例 vCPUs 数量的一个因素。

有关为多模型终端节点选择 SageMaker ML 实例类型的指导，请参阅。多模型终端节点部署的实例建议 (p. 2114).

多模型终端节点部署的实例建议

为多模型终端节点选择 SageMaker ML 实例类型时，需要考虑多个项目。为需要服务的所有模型预配置充足的 Amazon Elastic Block Store (Amazon EBS) 容量。平衡性能（最大限度地减少冷启动）和成本（不要过度预配置实例容量）。有关为终端节点和多模型终端节点的每个实例类型附加的存储卷大小的信息，请参阅。主机实例存储卷 (p. 2158). 对于配置为以 `MultiModel` 模式运行的容器，为其实例预配置的存储卷具有更多内存。这样可以在实例存储卷上缓存更多模型。

选择 SageMaker ML 实例类型时，请考虑以下事项：

- GPU 实例类型不支持多模型终端节点。
- 要在多模型终端节点后托管的模型的流量分配（访问模式）以及模型大小（实例上可加载到内存中的模型数）。
  - 将实例上的内存量视为要加载的模型的缓存空间。将 vCPU 数量视为对加载模型执行推理的并发限制（假设调用模型绑定到 CPU）。
  - 更多的实例内存使您可以加载更多模型，并准备好提供推理请求。您不需要浪费时间加载模型。
  - 更多的 vCPUs 使您可以同时调用更多唯一模型（再次假定推理绑定到 CPU）。
  - 请提供一些“松弛”内存以便卸载未使用的模型，特别是具有多个实例的多模型终端节点。如果实例或可用区发生故障，这些实例上的模型将被重新路由到终端节点后的其他实例。
- 容忍加载/下载时间：
  - d 实例类型系列（如 m5d、c5d 或 r5d）附带一个 NVMe（非易失性存储规范）SSD，它具有较高的 I/O 性能，并且可以减少将模型下载到存储卷以及容器从存储卷加载模型所需的时间。
  - 由于 d 实例类型附带 NVMe SSD 存储，因此 SageMaker 不会将 Amazon EBS 存储卷附加到这些托管多模型终端节点的 ML 计算实例。当模型大小相似且同质（也即，它们具有相似的推理延迟和资源要求）时，自动扩展的效果最佳。

在某些情况下，您可以通过选择不能同时将所有目标模型保存在内存中的实例类型来选择降低成本。SageMaker 在内存不足时动态卸载模型，以便为新目标模型腾出空间。对于不经常请求的模型，您需要付出动态负载延迟的代价。在延迟需求更严格的情况下，您可以选择更大的实例类型或更多实例。提前投入时间进行适当的性能测试和分析将非常有利于成功生产部署。

您可以使用 `ModelCacheHit` 指标的 `Average` 统计数据来监控已加载模型的请求比率。您可以使用 `ModelUnloadingTime` 指标的 `SampleCount` 统计数据来监控在一段时间内发送到容器的卸载请求数。如

果模型卸载频率过高（颠簸指示器，由于工作模型集的缓存空间不足，因此将卸载并重新加载模型），请考虑使用具有更多内存的更大实例类型，或者增加多模型终端节点后的实例数。对于具有多个实例的多模型终端节点，请注意可以在多个实例上加载一个模型。

SageMaker 多模型终端节点完全支持 Auto Scaling，它管理模型副本，以确保基于流量模式扩展模型。我们建议您在配置多模型终端节点和实例大小时综合考虑上述所有内容，并为终端节点设置自动扩展。用于触发自动扩展事件的调用率基于终端节点所提供的完整模型集中的聚合预测集。

创建多模型终端节点

您可以使用Amazon SDK for Python (Boto)或者 Amazon SageMaker 创建多模型终端节点。

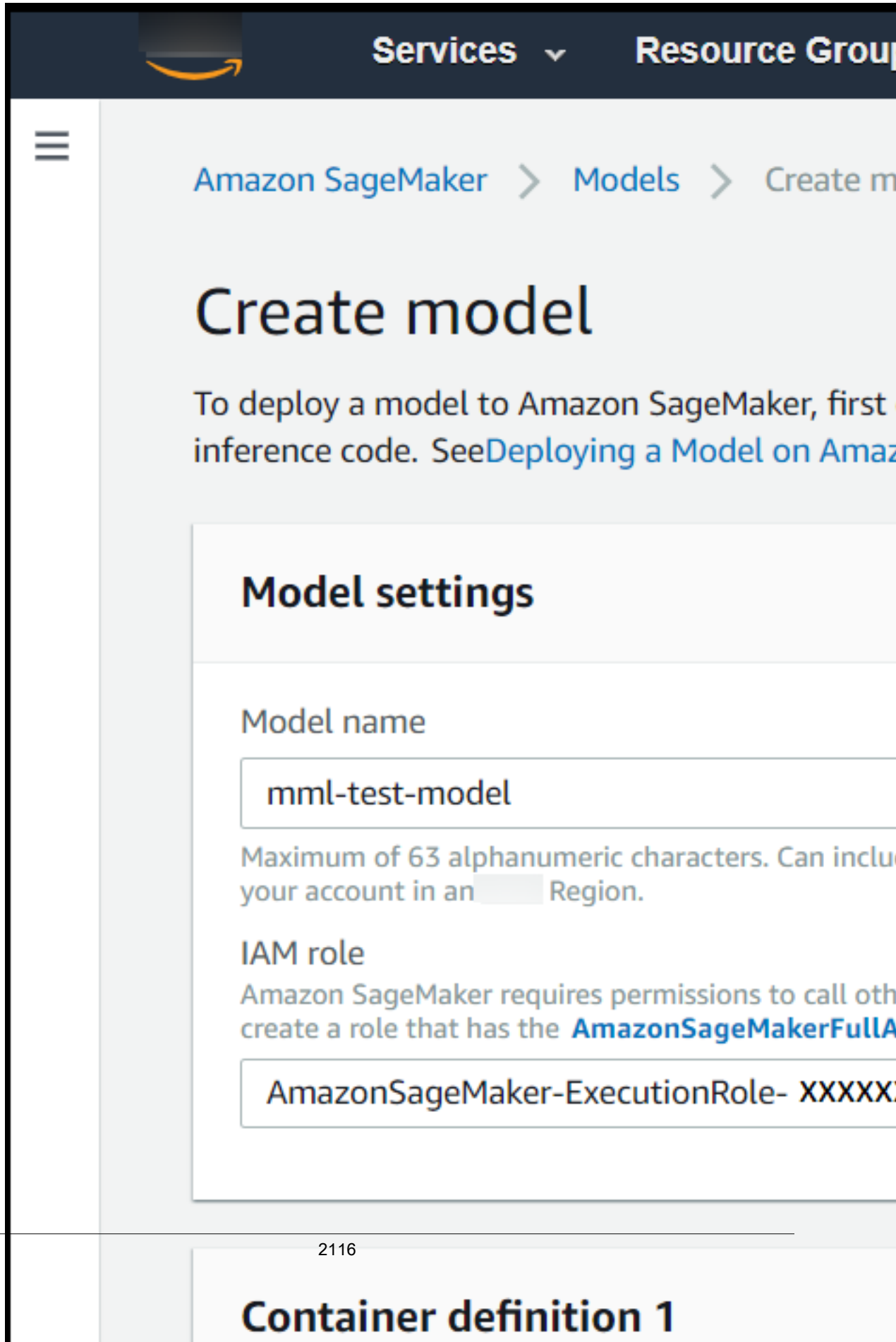
主题

- 创建多模型终端节点（控制台）(p. 2115)
- 创建多模型终端节点Amazon SDK for Python (Boto3) (p. 2117)

创建多模型终端节点（控制台）

创建多模型终端节点（控制台）

1. 打开位于 https://console.aws.amazon.com/sagemaker/ 的 Amazon SageMaker 控制台。
2. 选择 Models (模型)，然后从 Inference (推理) 组选择 Create models (创建模型)。
3. 对于 Model name (模型名称)，输入一个名称。
4. 适用于IAM 角色。选择或创建附加有 AmazonSageMakerFullAccess IAM 策略的 IAM 角色。
5. 在 Container definition (容器定义) 部分中，请为 Provide model artifacts and inference image options (提供模型构件和推理映像选项) 选择 Use multiple models (使用多个模型)。


Services
Resource Group
Amazon SageMaker
Models
Create m
Create model
To deploy a model to Amazon SageMaker, first
inference code. SeeDeploying a Model on Amaz
Model settings
Model name
mml-test-model
Maximum of 63 alphanumeric characters. Can inclu
your account in an Region.
IAM role
Amazon SageMaker requires permissions to call oth
create a role that has the AmazonSageMakerFullA
AmazonSageMaker-ExecutionRole- XXXXX
Container definition 1

6. 选择 Create model (创建模型)。
7. 按照单个模型终端节点的部署方式来部署多模型终端节点。有关说明，请参阅将模型部署到 SageMaker 托管服务 (p. 69)。

创建多模型终端节点Amazon SDK for Python (Boto3)

您可以使用 Amazon SageMaker 创建多模型终端节点。`create_model`、`create_endpoint_config`, 和`create_endpoint`如同您创建单个模型终端节点一样，但具有两处变化。定义模型容器时，您需要传递一个新的模型容器`Mode`参数值，`MultiModel`. 您还需要通过`ModelDataUrl`字段，该字段指定模型构件所在的的 Amazon S3 中的前缀，而不是部署单个模型时指定的单个模型构件的路径。

有关使用 SageMaker 将多个 XGBoost 模型部署到终端节点的示例笔记本，请参阅。多模型终端节点 xgBoost 示例笔记本.

以下过程概述了创建多模型终端节点的示例中所使用的关键步骤。

要部署模型 (Amazon适用于 Python 的开发工具包 (Boto 3))

1. 获取包含支持部署多模型终端节点的映像的容器。有关支持多模型终端节点的内置算法和框架容器的列表，请参阅支持的算法和框架 (p. 2113). 在此示例中，我们使用K 最近邻 (k-NN) 算法 (p. 1644)内置算法。我们将SageMaker Python SDK实用程序`image_uris.retrieve()`以获取 K 最近邻的内置算法映像的地址。

```
import sagemaker
region = sagemaker_session.boto_region_name
image = sagemaker.image_uris.retrieve("knn",region=region)
container = {
              'Image':        image,
              'ModelDataUrl': 's3://<BUCKET_NAME>/<PATH_TO_ARTIFACTS>',
              'Mode':         'MultiModel'
            }
```

2. 获取 Boto 3 SageMaker 客户端，并创建使用此容器的模型。

```
import boto3
sagemaker_client = boto3.client('sagemaker')
response = sagemaker_client.create_model(
              ModelName        = '<MODEL_NAME>',
              ExecutionRoleArn = role,
              Containers       = [container])
```

3. （可选）如果您使用的是串行推理管道，请获取要包含在管道中的其他容器，并将其包含在`CreateModel` 的 `Containers` 参数中：

```
preprocessor_container = {
               'Image':
 '<ACCOUNT_ID>.dkr.ecr.<REGION_NAME>.amazonaws.com/<PREPROCESSOR_IMAGE>:<TAG>'
            }

multi_model_container = {
              'Image':
 '<ACCOUNT_ID>.dkr.ecr.<REGION_NAME>.amazonaws.com/<IMAGE>:<TAG>',
              'ModelDataUrl': 's3://<BUCKET_NAME>/<PATH_TO_ARTIFACTS>',
              'Mode':         'MultiModel'
            }

response = sagemaker_client.create_model(
              ModelName        = '<MODEL_NAME>',
              ExecutionRoleArn = role,
              Containers       = [preprocessor_container, multi_model_container]
```

```
            )
```

4. （可选）如果您的使用案例不能从模型缓存中受益，请将ModelCacheSetting字段中的MultiModelConfig将参数转为Disabled，然后将其包含在Container呼吁的论点create_model. 的价值ModelCacheSetting字段为Enabled默认情况下。

```
container = {
                'Image': image,
                'ModelDataUrl': 's3://<BUCKET_NAME>/<PATH_TO_ARTIFACTS>',
                'Mode': 'MultiModel'
                'MultiModelConfig': {
                        // Default value is 'Enabled'
                        'ModelCacheSetting': 'Disabled'
                }
            }

response = sagemaker_client.create_model(
              ModelName        = '<MODEL_NAME>',
              ExecutionRoleArn = role,
              Containers       = [container]
            )
```

5. 为模型配置多模型终端节点。我们建议您至少为终端节点配置两个实例。这样，SageMaker 可以跨多个可用区为模型提供一组高度可用的预测。

```
response = sagemaker_client.create_endpoint_config(
                EndpointConfigName = '<ENDPOINT_CONFIG_NAME>',
                ProductionVariants=[
                     {
                        'InstanceType':        'ml.m4.xlarge',
                        'InitialInstanceCount': 2,
                        'InitialVariantWeight': 1,
                        'ModelName':            '<MODEL_NAME>',
                        'VariantName':          'AllTraffic'
                      }
                ]
            )
```

Note

在串行推理管道中只能使用一个启用多模型的终端节点。

6. 使用 EndpointName 和 EndpointConfigName 参数创建多模型终端节点。

```
response = sagemaker_client.create_endpoint(
              EndpointName       = '<ENDPOINT_NAME>',
              EndpointConfigName = '<ENDPOINT_CONFIG_NAME>')
```

调用多模型终端节点

要调用多模型终端节点，请使用invoke_endpoint就像调用单个模型端点一样，从 SageMaker 运行时进行一次更改。传递新 TargetModel 参数，该参数指定要定向到终端节点上的哪些模型。SageMaker 运行时间InvokeEndpoint请求支持X-Amzn-SageMaker-Target-Model作为新标头，它采用为调用指定的模型的相对路径。SageMaker 系统通过组合作为一部分提供的前缀来构造模型的绝对路径。CreateModel使用模型的相对路径进行 API 调用。

Amazon SDK for Python (Boto 3)

以下示例预测请求使用Amazon适用于 Python 的开发工具包 (Boto 3)在样本笔记本中。

```
response = runtime_sagemaker_client.invoke_endpoint(
                        EndpointName = "<ENDPOINT_NAME>",
                        ContentType  = "text/csv",
                        TargetModel  = "<MODEL_FILENAME>.tar.gz",
                        Body         = body)
```

Amazon CLI

以下示例演示如何使用以下示例发出具有两行的 CSV 请求：Amazon Command Line Interface(Amazon CLI):

```
aws sagemaker-runtime invoke-endpoint \
  --endpoint-name "<ENDPOINT_NAME>" \
  --body "1.0,2.0,5.0"$'\n'"2.0,3.0,4.0" \
  --content-type "text/csv" \
  --target-model "<MODEL_NAME>.tar.gz"
  output_file.txt
```

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`output_file.txt`如果推理成功，就会提出有关你的推理请求的信息。有关如何使用Amazon CLI请参阅使用Amazon CLI在 SageMaker Python 开发工具包文档中。

多模型终端节点根据需要动态加载目标模型。您可以在运行 MME 示例笔记本时发现这一点，该笔记本会针对单个终端节点后托管的多个目标模型遍历随机调用。针对给定模型的第一个请求需要更长时间，因为必须从 Amazon Simple Storage Service (Amazon S3) 下载模型并将其加载到内存中。（这称为冷启动。）由于加载模型后没有额外开销，因此后续调用完成速度更快。

在模型上重试请求 Notready异常错误

你第一次打电话`invoke_endpoint`对于模型，该模型将从 Amazon Simple Storage Services 下载并加载到推理容器中。这使得第一个电话需要更长的时间才能返回。由于模型已加载，所以对同一模型的后续调用完成速度更快。

SageMaker 返回呼叫的响应`invoke_endpoint`在 60 秒内。有些型号太大，无法在 60 秒内下载。如果模型没有在 60 秒超时限制之前完成加载，请求`invoke_endpoint`使用错误代码返回`ModelNotReadyException`，模型将继续下载并加载到推理容器中，持续长达 360 秒。如果你得到`ModelNotReadyException`的错误代码`invoke_endpoint`请求，请重试请求。默认情况下，Amazon适用于 Python 的开发工具包 (Boto 3)传统重试模式) 和 Java 重试`invoke_endpoint`导致的请求`ModelNotReadyException`错误消息。您可以配置重试策略以继续重试请求长达 360 秒。如果您预计模型下载并加载到容器的时间超过 60 秒，请将 SDK 套接字超时设置为 70 秒。有关为 Boto 3 配置重试策略的更多信息，请参阅配置重试模式. 下面的代码显示了一个示例，该示例将重试策略配置为重试调用`invoke_endpoint`持续时间长达 180 秒。

```
import boto3
from botocore.config import Config

# This example retry strategy sets the retry attempts to 2.
# With this setting, the request can attempt to download and/or load the model
# for upto 180 seconds: 1 orginal request (60 seconds) + 2 retries (120 seconds)
config = Config(
    read_timeout=70,
    retries={
        'max_attempts': 2  # This value can be adjusted to 5 to go up to the 360s max
 timeout
    }
)
runtime_sagemaker_client = boto3.client('sagemaker-runtime', config=config)
```

添加或删除模型

您可以将其他模型部署到多模型终端节点，并立即通过该终端节点调用这些模型。添加新模型时，您无需更新或关闭终端节点，因此可以避免为每个新模型创建和运行单独的终端节点的成本。

当实例达到内存容量且需要在容器中下载更多模型时，SageMaker 将从容器中卸载未使用的模型。当卷达到容量且需要下载新模型时，SageMaker 还会从实例存储卷中删除未使用的模型构件。首次调用新添加的模型需要较长时间，因为终端节点需要一定时间才能将模型从 S3 下载到托管终端节点的实例中的容器内存

如果终端节点已在运行，请将新的模型构件集复制到存储模型的 Amazon S3 位置。

```
# Add an AdditionalModel to the endpoint and exercise it
aws s3 cp AdditionalModel.tar.gz s3://my-bucket/path/to/artifacts/
```

> **Important**
>
> 请按照添加新模型的方式继续操作以便更新模型。使用新的唯一名称。不要覆盖 Amazon S3 中的模型构件，这是因为仍可能在容器中或在终端节点的实例存储卷上加载旧版模型。然后，调用新模型可能会调用旧版模型。

在 S3 中存储其他目标模型后，客户端应用程序即可从这些模型请求预测。

```
response = runtime_sagemaker_client.invoke_endpoint(
                        EndpointName='<ENDPOINT_NAME>',
                        ContentType='text/csv',
                        TargetModel='AdditionalModel.tar.gz',
                        Body=body)
```

要从多模型终端节点删除模型，请停止从客户端调用该模型，并将其从存储模型构件的 S3 位置中删除。

使用多模型服务器构建您自己的容器

Amazon SageMaker 中部署的自定义弹性容器注册 (ECR) 映像应符合中介绍的基本合同。将您自己的推理代码用于托管服务 (p. 2435)它管理 SageMaker 如何与运行您自己的推理代码的 Docker 容器进行交互。为了使容器能够同时加载和服务多个模型，还必须遵循其他 API 和行为。该附加合同包括用于加载、列出、获取和卸载模型的新 API，以及用于调用模型的不同 API。在发生错误需要遵循 API 的情况下，还存在一些不同行为。为了表明您的容器符合额外要求，您可以在 Docker 文件中添加以下命令：

```
LABEL com.amazonaws.sagemaker.capabilities.multi-models=true
```

SageMaker 还会在容器中注入环境变量。

```
SAGEMAKER_MULTI_MODEL=true
```

如果要为串行推理管道创建多模型终端节点，Docker 文件必须同时具有多模型和串行推理管道所需的标签。有关串行信息管道的更多信息，请参阅运行实时预测使用推理管道 (p. 2135)。

为了帮助您实现自定义容器的这些要求，我们提供了两个库：

- 多模型服务器是一个提供机器学习模型的开源框架，可在容器中安装此框架以便提供满足新的多模型终端节点容器 API 要求的前端。它提供了多模型终端节点所需的 HTTP 前端和模型管理功能，可以在单个容器中托管多个模型，在容器中动态加载和卸载模型，以及对指定加载模型执行推理。它还提供了一个支持可插入自定义后端处理程序的可插入后端，您可以在该后端实现自己的算法。
- SageMaker 推理工具包是一个引导多模型服务器的库，其配置和设置使其与 SageMaker 多模型终端节点兼容。它还允许您根据场景需求调整重要的性能参数，例如，每个模型的工作人员数。

使用 SageMaker 推理工具包

目前，支持多模型终端节点的唯一预构建容器是 MXNet 推理容器和 PyTorch 推理容器。如果要使用任何其他框架或算法，您需要构建一个容器。要执行该操作，最简单的方法是使用SageMaker 推理工具包扩展现有的预构建容器。SageMaker 推理工具包是一个多模型服务器 (MMS) 实施，它创建可在 SageMaker 中部署的终端节点。有关演示如何在 SageMaker 中设置和部署支持多模型终端节点的自定义容器的示例笔记本，请参阅多模型终端节点 BYOC 示例笔记本.

> **Note**
>
> SageMaker 推理工具包仅支持 Python 模型处理程序。如果要以任何其他语言实施处理程序，您必须构建自己的容器以实施额外的多模型终端节点 API。有关信息，请参阅为多个模型提供自定义容器的合同 (p. 2122)。

使用 SageMaker 推理工具包扩展容器

1. 创建一个模型处理程序。MMS 需要使用一个模型处理程序，这是一个 Python 文件，它实施一些函数以进行预处理，从模型中获得结果以及在模型处理程序中处理输出。有关模型处理程序的示例，请参阅示例笔记本中的 model_handler.py。
2. 导入推理工具包并使用其 `model_server.start_model_server` 函数以启动 MMS。以下示例来自于示例笔记本中的 `dockerd-entrypoint.py` 文件。请注意，`model_server.start_model_server` 调用传递上一步中描述的模型处理程序：

```
import subprocess
import sys
import shlex
import os
from retrying import retry
from subprocess import CalledProcessError
from sagemaker_inference import model_server

def _retry_if_error(exception):
    return isinstance(exception, CalledProcessError or OSError)

@retry(stop_max_delay=1000 * 50,
       retry_on_exception=_retry_if_error)
def _start_mms():
    # by default the number of workers per model is 1, but we can configure it through
 the
    # environment variable below if desired.
    # os.environ['SAGEMAKER_MODEL_SERVER_WORKERS'] = '2'
    model_server.start_model_server(handler_service='/home/model-server/
model_handler.py:handle')

def main():
    if sys.argv[1] == 'serve':
        _start_mms()
    else:
        subprocess.check_call(shlex.split(' '.join(sys.argv[1:])))

    # prevent docker exit
    subprocess.call(['tail', '-f', '/dev/null'])

main()
```

3. 在 `Dockerfile` 中，从第一步中复制模型处理程序，并在 `Dockerfile` 中将上一步中的 Python 文件指定为入口点。以下几行来自于示例笔记本中使用的 Dockerfile：

```
# Copy the default custom service file to handle incoming data and inference requests
COPY model_handler.py /home/model-server/model_handler.py

# Define an entrypoint script for the docker image
```

```
ENTRYPOINT ["python", "/usr/local/bin/dockerd-entrypoint.py"]
```

4. 构建并注册您的容器。示例笔记本中的以下 Shell 脚本构建容器，并将其上传到您的Amazon账户：

```
%%sh

# The name of our algorithm
algorithm_name=demo-sagemaker-multimodel

cd container

account=$(aws sts get-caller-identity --query Account --output text)

# Get the region defined in the current configuration (default to us-west-2 if none
 defined)
region=$(aws configure get region)
region=${region:-us-west-2}

fullname="${account}.dkr.ecr.${region}.amazonaws.com/${algorithm_name}:latest"

# If the repository doesn't exist in ECR, create it.
aws ecr describe-repositories --repository-names "${algorithm_name}" > /dev/null 2>&1

if [ $? -ne 0 ]
then
    aws ecr create-repository --repository-name "${algorithm_name}" > /dev/null
fi

# Get the login command from ECR and execute it directly
$(aws ecr get-login --region ${region} --no-include-email)

# Build the docker image locally with the image name and then push it to ECR
# with the full name.

docker build -q -t ${algorithm_name} .
docker tag ${algorithm_name} ${fullname}

docker push ${fullname}
```

现在，您可以使用此容器在 SageMaker 中部署多模型终端节点。

主题

- 为多个模型提供自定义容器的合同 (p. 2122)

为多个模型提供自定义容器的合同

要处理多个模型，您的容器必须支持 API 集，这些 API 允许 Amazon SageMaker 与容器通信，以便根据需要加载、列出、获取和卸载模型。`model_name` 在新 API 集中用作键输入参数。客户容器需要使用 `model_name` 作为映射键来跟踪已加载的模型。此外，`model_name` 是一个不透明的标识符，不一定是传递给 `InvokeEndpoint` API 的 `TargetModel` 参数的值。`InvokeEndpoint` 请求中的原始 `TargetModel` 值作为可用于日志记录目的的 `X-Amzn-SageMaker-Target-Model` 标头传递给 API 中的容器。

主题

- 加载模型 API (p. 2123)
- 列出模型 API (p. 2123)
- 获取模型 API (p. 2124)
- 卸载模型 API (p. 2124)
- 调用模型 API (p. 2124)

加载模型 API

指示容器将主体的 `url` 字段中的特定模型加载到客户容器的内存中，并使用分配的 `model_name` 来跟踪该模型。加载模型后，容器应准备好使用此 `model_name` 提供推理请求。

```
POST /models HTTP/1.1
Content-Type: application/json
Accept: application/json

{
     "model_name" : "{model_name}",
     "url" : "/opt/ml/models/{model_name}/model",
}
```

> **Note**
>
> 如果已加载 `model_name`，此 API 应返回 409。由于内存不足或任何其他资源而无法加载模型时，此 API 都应向 SageMaker 返回 507 HTTP 状态代码，然后启动卸载未使用的模型以回收。

列出模型 API

返回已加载到客户容器内存中的模型列表。

```
GET /models HTTP/1.1
Accept: application/json

Response =
{
    "models": [
        {
             "modelName" : "{model_name}",
             "modelUrl" : "/opt/ml/models/{model_name}/model",
        },
        {
            "modelName" : "{model_name}",
            "modelUrl" : "/opt/ml/models/{model_name}/model",
        },
        ....
    ]
}
```

此 API 还支持分页。

```
GET /models HTTP/1.1
Accept: application/json

Response =
{
    "models": [
        {
             "modelName" : "{model_name}",
             "modelUrl" : "/opt/ml/models/{model_name}/model",
        },
        {
            "modelName" : "{model_name}",
            "modelUrl" : "/opt/ml/models/{model_name}/model",
        },
        ....
    ]
}
```

SageMaker 可以在一开始调用列出模型 API，而不必提供的值。next_page_token. 如果 nextPageToken 字段作为响应的一部分返回，它将在后续的列出模型调用中作为 next_page_token 的值提供。如果未返回 nextPageToken，则意味着没有其他模型可供返回。

获取模型 API

这是 model_name 实体上的简单读取 API。

```
GET /models/{model_name} HTTP/1.1
Accept: application/json

{
     "modelName" : "{model_name}",
     "modelUrl" : "/opt/ml/models/{model_name}/model",
}
```

Note

如果未加载 model_name，此 API 应返回 404。

卸载模型 API

指示 SageMaker 平台指示客户容器从内存中卸载模型。这将在开始新模型加载过程时开始移出由平台确定的候选模型。当此 API 返回响应时，容器应回收预配置到 model_name 的资源。

```
DELETE /models/{model_name}
```

Note

如果未加载 model_name，此 API 应返回 404。

调用模型 API

从提供的特定 model_name 生成预测请求。SageMaker 运行时间InvokeEndpoint请求支持X-Amzn-SageMaker-Target-Model作为新标头，它采用为调用指定的模型的相对路径。SageMaker 系统通过组合作为一部分提供的前缀来构造模型的绝对路径。CreateModel使用模型的相对路径进行 API 调用。

```
POST /models/{model_name}/invoke HTTP/1.1
Content-Type: ContentType
Accept: Accept
X-Amzn-SageMaker-Custom-Attributes: CustomAttributes
X-Amzn-SageMaker-Target-Model: [relativePath]/{artifactName}.tar.gz
```

Note

如果未加载 model_name，此 API 应返回 404。

多模型终端节点安全性

多模型终端节点中的模型和数据位于实例存储卷和容器内存的同一位置中。Amazon SageMaker 终端节点的所有实例都在您拥有的单个租户容器上运行。只有您的模型才能在您的多模型终端节点上运行。您负责管理请求到模型的映射，并为用户提供正确的目标模型的访问权限。SageMaker 使用IAM 角色为您提供 IAM 基于身份的策略，您可用于指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。

默认情况下，IAM 委托人InvokeEndpoint对多模型终端节点的权限可以在中定义的 S3 前缀的地址调用任何模型CreateModel操作，前提是在操作中定义的 IAM 执行角色具有下载模型的权限。如果你需要限制InvokeEndpoint访问 S3 中的一组有限模型，可以执行以下操作之一：

- RestrictInvokeEndpont使用sagemaker:TargetModelIAM 条件键。例如，仅当 TargetModel 字段的值与指定的正则表达式之一匹配时，以下策略才允许 InvokeEndpont 请求：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sagemaker:InvokeEndpoint"
            ],
            "Effect": "Allow",
            "Resource":
            "arn:aws:sagemaker:region:account-id:endpoint/endpoint_name",
            "Condition": {
                // TargetModel provided must be from this set of values
                "StringLike": {
                    "sagemaker:TargetModel": ["company_a/*", "common/*"]
                }
            }
        }
    ]
}
```

  有关 SageMaker 条件键的更多信息，请参阅Amazon SageMaker 的条件键中的Amazon Identity and Access Management用户指南.
- 创建具有更严格的 S3 前缀的多模型终端节点。

有关 SageMaker 如何使用角色管理对终端节点的访问和代表您执行操作的更多信息，请参阅。SageMaker 角色 (p. 2722). 您的客户可能还具有由他们自己的合规性要求决定的特定数据隔离要求，可使用 IAM 身份来满足这些要求。

多模型终端节点部署的 CloudWatch 指标

Amazon SageMaker 为终端节点提供了一些指标，可用于监控缓存命中率、加载模型数以及模型在多模型终端节点上加载、下载和上传的等待时间。想要了解有关信息，请参阅Multi-Model Endpoint Model Loading Metrics (多模型终端节点模型加载指标)和Multi-Model Endpoint Model Instance Metrics (多模型终端节点模型实例指标)在使用 Amazon CloudWatch 监控 Amazon SageMaker (p. 2810). 不支持基于模型的指标。

托管在一个终端后面使用不同容器的多个模型

使用 SageMaker 多容器终端节点，客户可以在单个 SageMaker 终端节点上部署多个使用不同模型或框架的容器。容器可以作为推理管道按顺序运行，也可以使用直接调用来单独访问每个容器，以提高终端节点利用率和优化成本。

有关按顺序调用多容器终端节点中容器的信息，请参阅主机模型以及预处理逻辑作为一个端点后面的串行推理管道 (p. 2131).

有关调用多容器终端节点中的特定容器的信息，请参阅使用具有直接调用的多容器终端节点 (p. 2127)

主题

- 创建多容器终端节点（Boto 3） (p. 2126)
- 更新多容器终端节点 (p. 2126)
- 删除多容器终端节点 (p. 2126)
- 使用具有直接调用的多容器终端节点 (p. 2127)

创建多容器终端节点（Boto 3）

通过调用创建多容器终端节点CreateModel、CreateEndpointConfig, 和CreateEndpoint像创建任何其他终端节点一样的 API。您可以按顺序作为推理管道运行这些容器，也可以使用直接调用来运行每个容器。调用多容器终端节点时有以下要求：`create_model`：

- 使用`Containers`参数而不是`PrimaryContainer`，并在`Containers`参数。
- 这些区域有：`ContainerHostname`具有直接调用的多容器终端节点中的每个容器都需要参数。
- 设置`Mode`的参数`InferenceExecutionConfig`字段到`Direct`直接调用每个容器，或`Serial`以便使用容器作为推理管道。默认模式为 `Serial`。

> **Note**
>
> 目前，多容器终端节点最多支持 15 个容器。

以下示例为直接调用创建多容器模型。

1. 创建容器元素`InferenceExecutionConfig`直接调用。

```
container1 = { 'Image':        '123456789012.dkr.ecr.us-east-1.amazonaws.com/
myimage1:mytag',
                'ContainerHostname': 'firstContainer'}

container2 = { 'Image':        '123456789012.dkr.ecr.us-east-1.amazonaws.com/
myimage2:mytag',
                'ContainerHostname': 'secondContainer'
             }
inferenceExecutionConfig = {'Mode': 'Direct'
                           }
```

2. 使用容器元素创建模型并设置`InferenceExecutionConfig`字段中返回的子位置类型。

```
import boto3
sm_client = boto3.Session().client('sagemaker')

response = sm_client.create_model(
              ModelName        = 'my-direct-mode-model-name',
              InferenceExecutionConfig = inferenceExecutionConfig,
              ExecutionRoleArn = role,
              Containers       = [container1, container2])
```

要创建 endoint，然后你可以打电话创建 _endpoint_config和创建 _终端点就像创建任何其他终端节点一样。

更新多容器终端节点

要更新多容器终端节点，请完成以下步骤。

1. Callcreate_model创建一个具有新价值的新模型`Mode`中的参数`InferenceExecutionConfig`字段中返回的子位置类型。
2. Call创建 _endpoint_config通过使用您在上一步中创建的新模型创建具有不同名称的新端点配置。
3. Callupdate_终端点以使用您在上一步中创建的新终端节点配置更新终端节点。

删除多容器终端节点

要删除终端节点，请调用delete_终端点，并提供要删除的终端节点的名称作为`EndpointName`参数。

使用具有直接调用的多容器终端节点

SageMaker 多容器端点使客户能够部署多个容器，以便在 SageMaker 终端节点上部署不同的模型。您可以在单个终端节点上托管多达 15 个不同的推理容器。通过使用直接调用，您可以向托管在多容器终端节点上的特定推理容器发送请求。

主题

- 直接调用调用多容器终端节点 (p. 2127)
- 通过直接调用实现多容器终端节点的安全 (p. 2127)
- 具有直接调用的多容器终端节点的指标 (p. 2128)
- 自动扩展多容器终端节点 (p. 2130)
- 排查多容器终端节点 (p. 2130)

直接调用调用多容器终端节点

要通过直接调用调用多容器终端节点，请调用invoke_终端点就像调用任何其他终端节点一样，并通过使用`TargetContainerHostname`参数。

以下示例直接调用`secondContainer`来获得预测的多容器终端节点。

```
import boto3
runtime_sm_client = boto3.Session().client('sagemaker-runtime')

response = runtime_sm_client.invoke_endpoint(
   EndpointName ='my-endpoint',
   ContentType = 'text/csv',
   TargetContainerHostname='secondContainer',
   Body = body)
```

对于向多容器终端节点发出的每个直接调用请求，只有具有`TargetContainerHostname`处理调用请求。如果执行以下任一操作，您将收到验证错误：

- 指定`TargetContainerHostname`终端节点中不存在
- 请勿为指定值`TargetContainerHostname`在向配置为直接调用的终端节点的请求中
- 为指定值`TargetContainerHostname`在向未配置为直接调用的终端节点的请求中。

通过直接调用实现多容器终端节点的安全

对于具有直接调用的多容器终端节点，通过共享内存和存储卷，在单个实例中托管多个容器。您有责任使用安全容器，维护请求与目标容器的正确映射，并为用户提供对目标容器的正确访问权限。SageMaker 使用 IAM 角色提供基于 IAM 身份的策略，您可以使用这些策略来指定是允许还是拒绝该角色访问资源，以及在什么情况下。有关 IAM 角色的信息，请参阅IAM 角色中的Amazon Identity and Access Management用户指南. 有关基于身份的策略的信息，请参阅基于身份的策略和基于资源的策略.

默认情况下，IAM 委托人`InvokeEndpoint`具有直接调用的多容器终端节点上的权限可以使用您在调用时指定的终端节点名称调用终端节点内的任何容器`invoke_endpoint`. 如果你需要限制`invoke_endpoint`访问多容器终端节点内的一组有限的容器，请使用`sagemaker:TargetContainerHostname`IAM 条件键。以下策略显示了如何限制对终端节点内特定容器的调用。

以下策略允许：`invoke_endpoint`仅当为时请求`TargetContainerHostname`字段与指定的正则表达式之一匹配。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Action": [
                "sagemaker:InvokeEndpoint"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:sagemaker:region:account-id:endpoint/endpoint_name",
            "Condition": {
                "StringLike": {
                    "sagemaker:TargetContainerHostname": ["customIps*", "common*"]
                }
            }
        }
    ]
}
```

以下政策拒绝invoke_endpoint请求时的值TargetContainerHostname字段匹配中的指定正则表达式之一Deny网页。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sagemaker:InvokeEndpoint"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:sagemaker:region:account-id:endpoint/endpoint_name",
            "Condition": {
                "StringLike": {
                    "sagemaker:TargetContainerHostname": ["*"]
                }
            }
        },
        {
            "Action": [
                "sagemaker:InvokeEndpoint"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:sagemaker:region:account-id:endpoint/endpoint_name",
            "Condition": {
                "StringLike": {
                    "sagemaker:TargetContainerHostname": ["special*"]
                }
            }
        }
    ]
}
```

有关 SageMaker 条件键的信息，请参阅SageMaker 的条件键中的Amazon Identity and Access Management用户指南.

具有直接调用的多容器终端节点的指标

除了中列出的终端节点指标使用 Amazon CloudWatch 监控 Amazon SageMaker (p. 2810)，SageMaker 还提供了每个容器的指标。

直接调用的多容器终端节点的每个容器指标位于 CloudWatch 中，并分为两个命名空间：AWS/SageMaker和aws/sagemaker/Endpoints. 这些区域有：AWS/SageMaker命名空间包括与调用相关的指标，以及aws/sagemaker/Endpoints命名空间包括内存和 CPU 利用率指标。

下表列出了直接调用的多容器终端节点的每容器指标。所有指标都使用 [EndpointName, VariantName, ContainerName] 维度，它过滤特定终端节点、特定变体以及与特定容器对应的指标。这些指标与推理管道中的指标名称相同，但是在每个容器级别上 [EndpointName, VariantName, ContainerName]。

指标名称	描述	维度	NameSpace
Invocations	的数量InvokeEndpoint发送到终端节点内容器的请求。要获取发送到该容器的请求总数，请使用Sum统计数据。单位：无有效统计数据：Sum、Sample Count	EndpointName, VariantName, ContainerName	AWS/SageMaker
Invocation4XX Errors	的数量InvokeEndpoint要求模型返回4xx特定容器上的 HTTP 响应代码。对于每个4xx响应时，SageMaker 发送1.单位：无有效统计数据：Average、Sum	EndpointName, VariantName, ContainerName	AWS/SageMaker
Invocation5XX Errors	的数量InvokeEndpoint要求模型返回5xx特定容器上的 HTTP 响应代码。对于每个5xx响应时，SageMaker 发送1.单位：无有效统计数据：Average、Sum	EndpointName, VariantName, ContainerName	AWS/SageMaker
ContainerLatency	从 SageMaker 观察到的那样，目标容器做出响应的时间。ContainerLatency包括以下操作所花的时间：发送请求，从模型容器中提取响应，以及完成容器中的推理。单位：微秒有效统计数据：Average、Sum、Min、Max、Sample Count	EndpointName, VariantName, ContainerName	AWS/SageMaker
OverheadLatency	为响应 SageMaker 的客户端提出的开销请求所花的时间增加。OverheadLatency从 SageMaker 收到请求的时间开始计算，直到它返回响应到客户端，减去ModelLatency.除其他因素外，开销延迟还可能由于请求和响应负载大小、请求频率以及请求的身份验证或授权而变化。单位：微秒有效统计数	EndpointName, VariantName, ContainerName	AWS/SageMaker

	据：Average、Sum、Min、Max，` 样本计数`		
CPUUtilization	实例上运行的每个容器所使用的 CPU 单位的百分比。值范围从 0% 到 100%，并且乘以 CPU 数目。例如，如果有四个 CPU，CPUUtilization 可以介于 0% 到 400% 之间。对于具有直接调用的终端节点，CPU 利用率指标的数量等于该终端节点中的容器数量。单位：百分比	EndpointName, VariantName, ContainerName	aws/sagemaker/Endpoints
MemoryUtilizaton	实例上运行的每个容器所使用的内存的百分比。此值范围从 0% 到 100%。与 CPULize 类似，在具有直接调用的终端节点中，内存利用率指标的数量等于该终端节点中的容器数量。单位：百分比	EndpointName, VariantName, ContainerName	aws/sagemaker/Endpoints

上表中的所有指标都特定于具有直接调用的多容器终端节点。除了这些特殊的每容器指标之外，还有一些具有维度的变体级别的指标[EndpointName, VariantName]对于表中的所有指标的预期ContainerLatency.

自动扩展多容器终端节点

如果要使用InvocationsPerInstance指标，我们建议每个容器中的模型在每个推理请求中显示相似的 CPU 利用率和延迟。建议这样做是因为如果流向多容器终端节点的流量从 CPU 使用率较高模式转变为低 CPU 使用率模式，但总呼叫量保持不变，则终端节点不会扩展，并且可能没有足够的实例来处理对高 CPU 使用率的所有请求模型。有关自动扩展终端节点的信息，请参自动扩展 Amazon SageMaker 模型 (p. 2144).

排查多容器终端节点

以下部分可帮助您解决多容器终端节点的错误。

ping Health 检查错误

对于多个容器，端点内存和 CPU 在端点创建过程中承受更大的压力。具体来说，MemoryUtilization和CPUUtilization指标高于单容器终端节点的指标，因为利用压力与容器数量成正比。因此，我们建议您选择具有足够内存和 CPU 的实例类型，以确保实例上有足够的内存来加载所有模型（同样的指导适用于部署推理管道）。否则，您的终端节点创建可能会失败并出现错误，例如xxx did not pass the ping health check.

缺少接受绑定到端口 = true Docker 标签

多容器终端节点中的容器侦听中指定的端口。SAGEMAKER_BIND_TO_PORT环境变量而不是端口 8080。当容器在多容器终端节点中运行时，SageMaker 会自动向容器提供此环境变量。如果此环境变量不存在，则容器应默认为使用端口 8080。要指示您的容器使用此要求进行编译，请使用以下命令将标签添加到您的 Dockerfile：

```
LABEL com.amazonaws.sagemaker.capabilities.accept-bind-to-port=true
```

否则，您将看到一条错误消息，例如`Your Ecr Image XXX does not contain required com.amazonaws.sagemaker.capabilities.accept-bind-to-port=true Docker label(s).`

如果您的容器需要侦听第二个端口，请在 `SAGEMAKER_SAFE_PORT_RANGE` 环境变量指定的范围中选择端口。以格式将值指定为包含范围*`XXXX`*-*`YYYY`*，其中 XXXX 和 YYYY 是多位数整数。当您在多容器终端节点中运行容器时，SageMaker 会自动提供该值。

主机模型以及预处理逻辑作为一个端点后面的串行推理管道

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的推理管道是 Amazon SageMaker 由两到十五个容器的线性序列组成的模型，用于处理对数据进行推理的请求。可以使用推理管道定义和部署预训练 SageMaker 内置算法与您打包在 Docker 容器中的自定义算法的任意组合。您可以使用推理管道合并预处理、预测和后处理数据科学任务。推理管道是完全托管的。

您可以添加 SageMaker Spark ML Services 和 scikit-learn 容器重复使用为训练模型开发的数据转换程序。整个组装的推理管道可以被视为 SageMaker 可用于直接进行实时预测或者处理批量转换，无需外部预处理。

在推理管道模型中， SageMaker 将调用作为 HTTP 请求序列进行处理。管道中的第一个容器处理初始请求，然后中间响应作为请求发送到第二个容器，以此类推，对于管道中的每个容器。 SageMaker 将最终响应返回给客户端。

当你部署管道模型时， SageMaker 在终端节点或转换作业中安装和运行每个 Amazon Elastic Compute Cloud (Amazon EC2) 实例上的所有容器。未来的处理和推理可以在低延迟下运行，因为容器与相同的 EC2 实例处于相同位置中。您可以使用 `CreateModel` 操作或者从控制台为管道模型定义容器。而不是设置一个`PrimaryContainer`，您将 `Containers`参数。要设置组成管道的容器，您还可以指定容器的执行顺序。

管道模型是不可变的，但您可以通过以下操作更新推理管道：使用`UpdateEndpoint`operation. 这种模块性支持在试验期间实现更高的灵活性。

有关如何使用创建推理管道的信息 SageMaker 模型注册表，请参阅使用模型注册表注册并部署模型 (p. 2252).

使用此功能不会产生额外费用。您仅需为终端节点上运行的实例支付费用。

主题

- 推理管道的示例笔记本 (p. 2131)
- 使用 Spark ML 和 Scikit-learn 的特征处理 (p. 2132)
- 创建管道模型 (p. 2132)
- 运行实时预测使用推理管道 (p. 2135)
- 运行使用推理管道进行 Batch 转换 (p. 2137)
- 推理管道日志和指标 (p. 2138)
- 推理管道问题排查 (p. 2142)

推理管道的示例笔记本

有关如何创建和部署推理管道的示例，请参阅在 EMR 上使用 SparkML 进行培训并在 SageMaker 上托管样本笔记本。有关创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147).

要查看所有 SageMaker 示例后，在创建并打开笔记本实例后，选择SageMaker 示例选项卡。有三个推理管道笔记本。刚刚介绍的前两个推理管道笔记本位于 `advanced_functionality` 文件夹中，第三个笔记本

位于 `sagemaker-python-sdk` 文件夹中。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

使用 Spark ML 和 Scikit-learn 的特征处理

在使用亚马逊训练模型之前 SageMaker 您可以使用 Spark 和 scikit-learn 预处理器转换数据和工程特征。

使用 Spark ML 的特征处理

你可以使用运行 Spark ML 作业AmazonGlue、来自您的无服务器 ETL（提取、转换、加载）服务 SageMaker 笔记本。您还可以连接到现有 EMR 集群以使用 Amazon EMR 运行 Spark ML 作业。要执行这个，您需要一个Amazon Identity and Access Management(IAM) 角色，该角色授予可从 SageMaker 笔记本发出调用的权限。到Amazon Glue.

> Note
>
> 要查看哪些 Python 和 Spark 版本Amazon Glue支持，请参阅Amazon Glue 发行说明.

在设计特征之后，您可以将带有 MLeap 的 Spark ML 作业打包并序列化为 MLeap 容器，您可以将这样的容器添加到推理管道中。您不需要使用外部管理的 Spark 集群。使用此方法，您可以从几个示例行无缝扩展到数 TB 的数据。相同的转换器可同时在训练和推理期间使用，因此您不需要重复预处理和特征设计逻辑，或者开发一次性解决方案来保存这些模型。借助推导管道，您不需要维护外部基础设施，可以直接利用数据输入进行预测。

当你在上运行 Spark ML 作业时Amazon Glue，aSpark ML 管道被序列化为mLeap格式的日期和时间。然后，你可以将该作业与SparkML 模型服务容器在 SageMaker 推理管道。mLeap是一种用于机器学习管道的序列化格式和执行引擎。它支持 Spark、Scikit-Learn 和 TensorFlow 用于训练管道并将其导出到称为 MLeap Bundle 的序列化管道中。您可以将服务包反序列化到 Spark 用于批量模式评分，或者反序列化到 MLeap 运行时以强化实时 API 服务。

使用 Scikit-Learn 进行功能处理

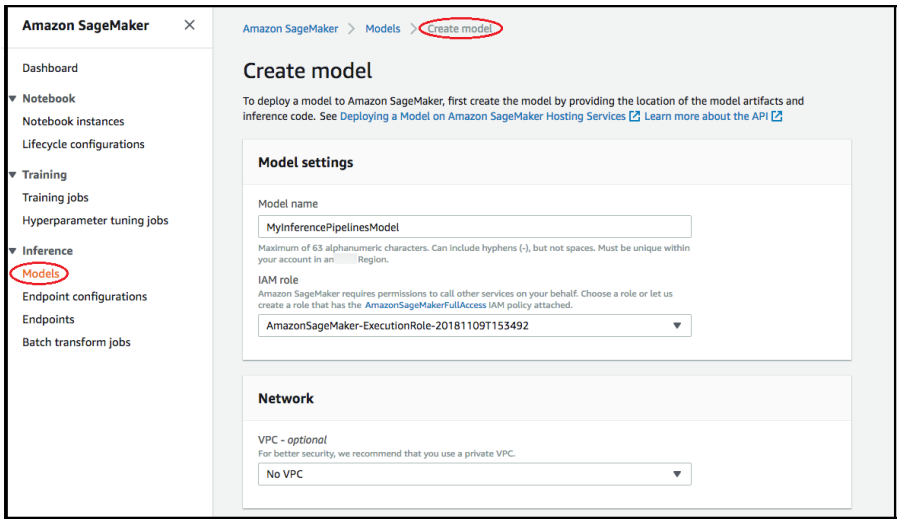
您可以直接在 Amazon SageMaker 中运行 scikit-learn 作业并将其打包到容器中。在一个 Python 代码示例中，生成通过费雪鸢尾花卉数据集进行训练的 scikit-learn 特征化模型并根据形态指标来预测鸢尾花。有关该示例，请参阅使用 Sagemaker Scikit-learn 的 IRIS 训练和预测。

创建管道模型

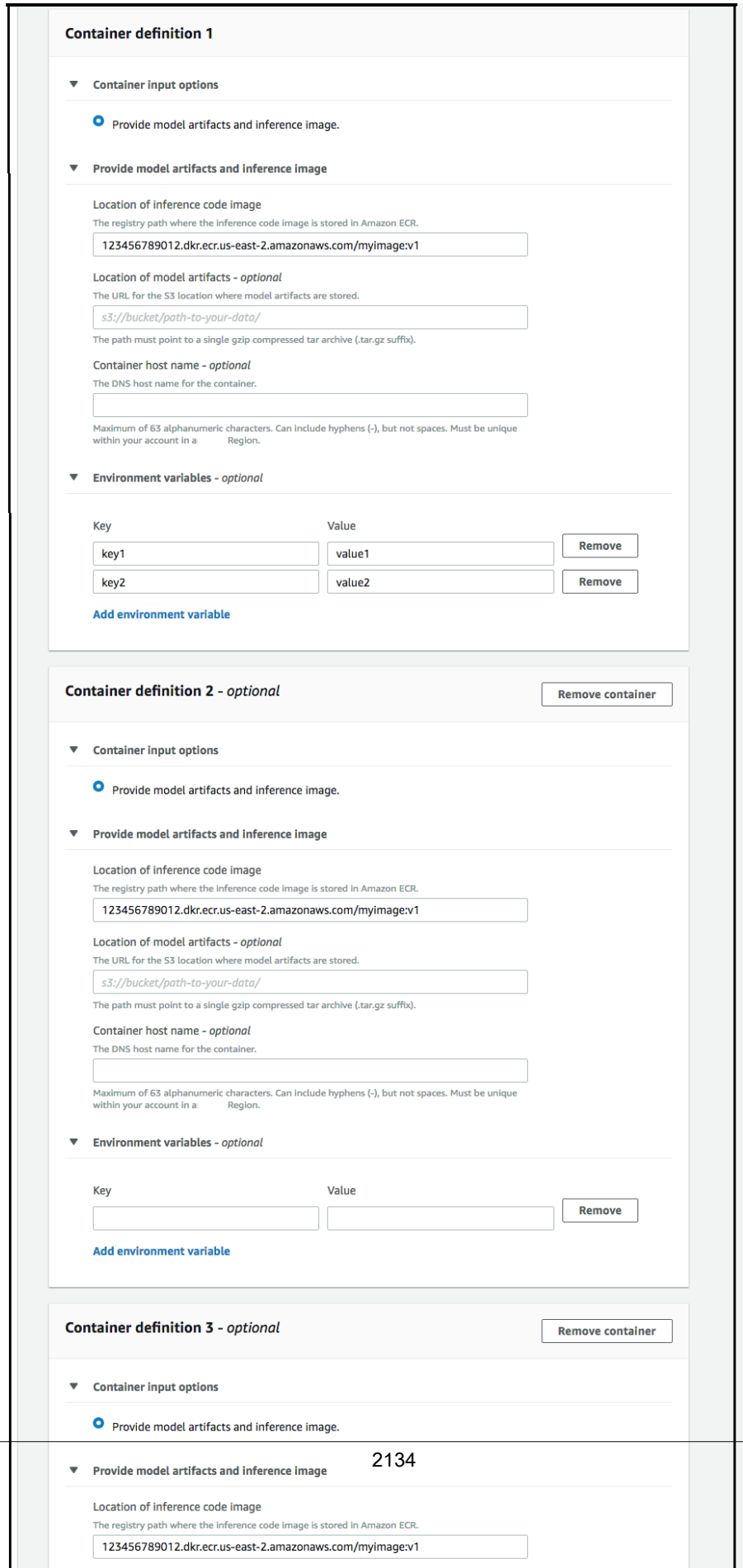
创建能够部署到的管道模型终端节点或用于批量转换任务，使用亚马逊 SageMaker 控制台或者`CreateModel`operation.

创建推理管道（控制台）

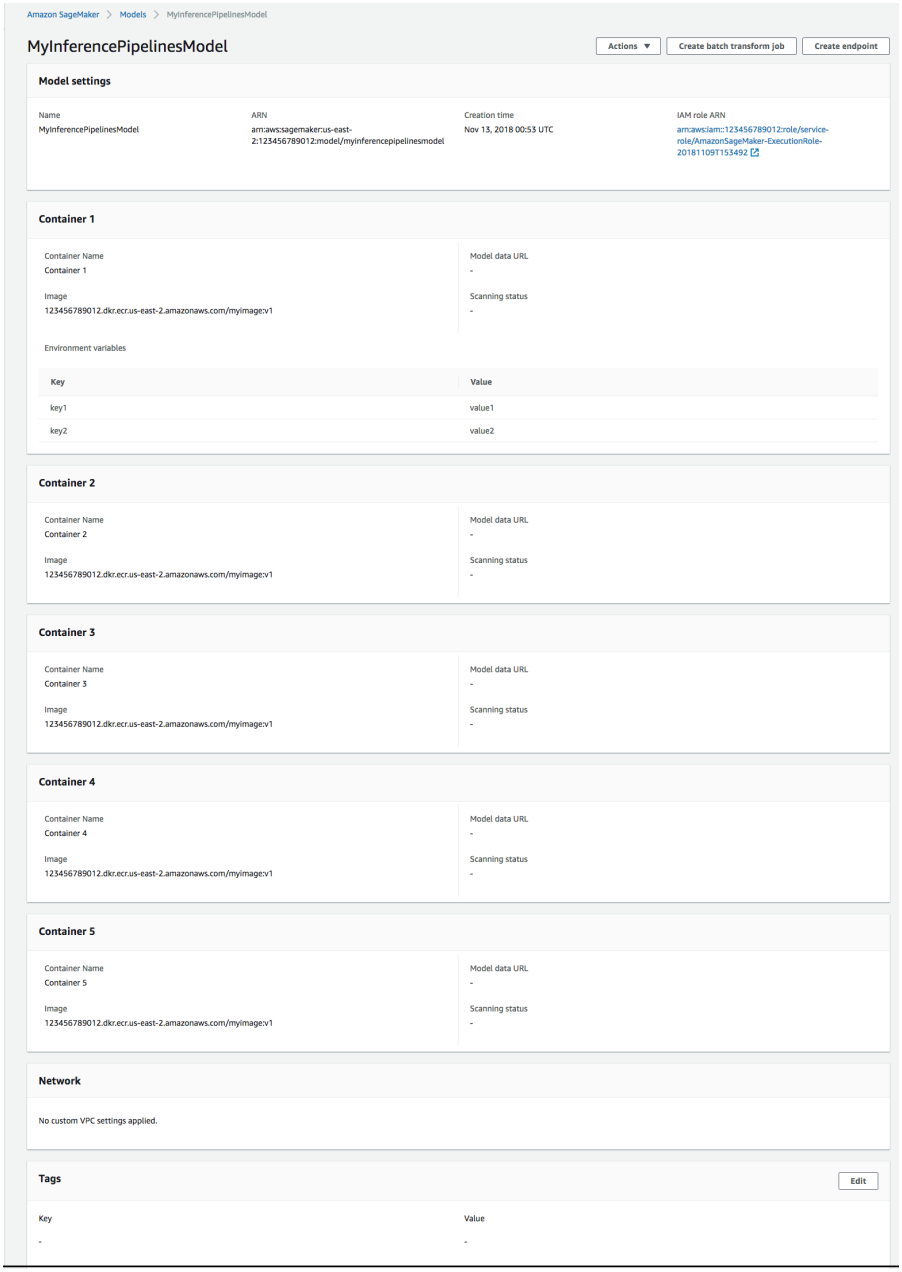
1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 选择模型，然后选择创建模型 来自 的推理组.
3. 在存储库的创建模型页面中，提供模型名称，选择 IAM 角色，如果您想使用私有 VPC，请指定 VPC价值.



4. 要添加推理管道中容器的信息，选择添加容器，然后选择下一步.
5. 按照要执行的顺序填写每个容器的字段，最多为 15 个字段。完成容器输入选项, ,推理代码图像的位置，以及（可选）模型工件的位置、容器主机名, 和环境变量字段之间没有不同。.


Container definition 1
Container input options
Provide model artifacts and inference image.
Provide model artifacts and inference image
Location of inference code image
The registry path where the inference code image is stored in Amazon ECR.
123456789012.dkr.ecr.us-east-2.amazonaws.com/myimage:v1
Location of model artifacts - optional
The URL for the S3 location where model artifacts are stored.
s3://bucket/path-to-your-data/
The path must point to a single gzip compressed tar archive (.tar.gz suffix).
Container host name - optional
The DNS host name for the container.
Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in a Region.
Environment variables - optional
Key
Value
key1
value1
Remove
key2
value2
Remove
Add environment variable
Container definition 2 - optional
Remove container
Container input options
Provide model artifacts and inference image.
Provide model artifacts and inference image
Location of inference code image
The registry path where the inference code image is stored in Amazon ECR.
123456789012.dkr.ecr.us-east-2.amazonaws.com/myimage:v1
Location of model artifacts - optional
The URL for the S3 location where model artifacts are stored.
s3://bucket/path-to-your-data/
The path must point to a single gzip compressed tar archive (.tar.gz suffix).
Container host name - optional
The DNS host name for the container.
Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in a Region.
Environment variables - optional
Key
Value
Remove
Add environment variable
Container definition 3 - optional
Remove container
Container input options
Provide model artifacts and inference image.
Provide model artifacts and inference image
Location of inference code image
The registry path where the inference code image is stored in Amazon ECR.
123456789012.dkr.ecr.us-east-2.amazonaws.com/myimage:v1

这些区域有：MyInferencePipelineModel页面总结了提供输入的容器的设置t 代表模型。如果您在对应的容器定义中提供了环境变量， SageMaker 将它们显示在环境变量字段中返回的子位置类型。



运行实时预测使用推理管道

您可以在推理管道中使用经训练的模型直接进行实时预测，无需执行外部预处理。当您配置管道时，您可以选择使用内置功能变压器已在 Amazon SageMaker 中提供。或者，您可以使用几行 scikit-learn 或 Spark 代码来实现自己的转换逻辑。

mLeap它是一种用于机器学习管道的序列化格式和执行引擎，它支持 Spark、scikit-learn 和 TensorFlow 用于训练管道并将其导出到称为 MLeap Bundle 的序列化管道中。您可以将服务包反序列化到 Spark 用于批量模式评分，或者反序列化到 MLeap 运行时以强化实时 API 服务。

管道中的容器侦听由 SAGEMAKER_BIND_TO_PORT 环境变量指定的端口（而不是 8080）。在推理管道中运行时，SageMaker 自动向容器提供此环境变量。如果此环境变量不存在，则容器应默认为使用端口 8080。要指示您的容器使用此要求进行编译，请使用以下命令将标签添加到您的 Dockerfile：

```
LABEL com.amazonaws.sagemaker.capabilities.accept-bind-to-port=true
```

如果你的容器需要在第二个端口上收听， 在指定范围内选择一个端口。SAGEMAKER_SAFE_PORT_RANGE环境变量。以格式将值指定为包含范围**"XXXX-YYYY"**，其中，XXXX和YYYY是多位整数。 SageMaker 在多容器管道中运行容器时，自动提供此值。

Note

在包括在内的管道中使用自定义 Docker 镜像SageMaker 内置算法，您需要Amazon Elastic Container Registry (Amazon ECR) 政策. 您的 Amazon ECR 存储库必须授予 SageMaker 权限才能拉取映像。有关更多信息，请参阅 推理管道的 Amazon ECR 权限问题排查 (p. 2143)。

创建和部署推理管道终端节点

以下代码使用 SparkML 和 XGBoost 模型按顺序创建和部署实时推理管道模型。 SageMaker SDK。

```
from sagemaker.model import Model
from sagemaker.pipeline_model import PipelineModel
from sagemaker.sparkml.model import SparkMLModel

sparkml_data = 's3://{}/{}/{}'.format(s3_model_bucket, s3_model_key_prefix, 'model.tar.gz')
sparkml_model = SparkMLModel(model_data=sparkml_data)
xgb_model = Model(model_data=xgb_model.model_data, image=training_image)

model_name = 'serial-inference-' + timestamp_prefix
endpoint_name = 'serial-inference-ep-' + timestamp_prefix
sm_model = PipelineModel(name=model_name, role=role, models=[sparkml_model, xgb_model])
sm_model.deploy(initial_instance_count=1, instance_type='ml.c4.xlarge',
 endpoint_name=endpoint_name)
```

请求从推理管道终端节点进行实时推理

以下示例演示如何通过调用推理终端节点并传递 JSON 格式的请求负载：

```
import sagemaker
from sagemaker.predictor import json_serializer, json_deserializer, Predictor

payload = {
        "input": [
            {
                "name": "Pclass",
                "type": "float",
                "val": "1.0"
            },
            {
                "name": "Embarked",
                "type": "string",
                "val": "Q"
            },
            {
                "name": "Age",
                "type": "double",
                "val": "48.0"
            },
            {
                "name": "Fare",
                "type": "double",
```

```
                "val": "100.67"
            },
            {
                "name": "SibSp",
                "type": "double",
                "val": "1.0"
            },
            {
                "name": "Sex",
                "type": "string",
                "val": "male"
            }
        ],
        "output": {
            "name": "features",
            "type": "double",
            "struct": "vector"
        }
    }

predictor = Predictor(endpoint=endpoint_name, sagemaker_session=sagemaker.Session(),
 serializer=json_serializer,
                                content_type='text/csv', accept='application/json'

print(predictor.predict(payload))
```

您从 `predictor.predict(payload)` 中得到的响应是模型的推理结果。

实时推理管道示例

您可以使用 SKLearn 预测器运行此示例笔记本，该预测器显示如何部署终端节点，运行推理请求，然后对响应进行反序列化。在中找到此笔记本和更多例子亚马逊 SageMaker 示例 GitHub 知识库.

运行使用推理管道进行 Batch 转换

要获取对整个数据集的推理，请在训练好的模型上运行批处理转换。要在完整数据集上运行推理，您可以在批量转换作业中使用创建并部署到终端节点上用于实时处理的同一个推理管道模型。要在管道中运行批量转换作业，请从 Amazon S3 下载输入数据，并在一个或多个 HTTP 请求中将这些数据发送到推理管道模型。有关说明如何准备数据用于批量转换的示例，请参阅 “第 2 节-使用 Sciit Learn 预处理原始住房数据”。亚马逊 SageMaker 使用线性学习者示例笔记本的多模型终端. 有关亚马逊的信息 SageMaker 请参阅批量转换使用批量转换 (p. 2229).

> **Note**
>
> 在包括在内的管道中使用自定义 Docker 镜像亚马逊 SageMaker 内置算法，您需要Amazon Elastic Container Registry (ECR) 政策. 您的 Amazon ECR 存储库必须授予 SageMaker 提取镜像的权限。有关更多信息，请参阅 推理管道的 Amazon ECR 权限问题排查 (p. 2143)。

以下示例演示如何使用运行转换作业：这亚马逊 SageMaker Python 开发工具包. 在此示例中，`model_name` 是组合了 SparkML 和 XGBoost 模型（这些模型在前面的示例中创建）的推理管道。由指定的 Amazon S3 位置`input_data_path`包含要下载并发送到 Spark ML 模型的 CSV 格式的输入数据。转换作业完成后，指定的 Amazon S3 位置：`output_data_path`包含 XGBoost 模型返回的输出数据（采用 CSV 格式）。

```
import sagemaker
input_data_path = 's3://{}/{}/{}'.format(default_bucket, 'key', 'file_name')
output_data_path = 's3://{}/{}'.format(default_bucket, 'key')
transform_job = sagemaker.transformer.Transformer(
    model_name = model_name,
    instance_count = 1,
    instance_type = 'ml.m4.xlarge',
    strategy = 'SingleRecord',
```

```
    assemble_with = 'Line',
    output_path = output_data_path,
    base_transform_job_name='inference-pipelines-batch',
    sagemaker_session=sagemaker.Session(),
    accept = CONTENT_TYPE_CSV)
transform_job.transform(data = input_data_path,
                        content_type = CONTENT_TYPE_CSV,
                        split_type = 'Line')
```

推理管道日志和指标

要维护 Amazon 的可靠性、可用性和性能，实施监控非常重要。 SageMaker 资源的费用。要监控推理管道性能并进行问题排查，请使用 Amazon CloudWatch 日志和错误消息。有关 SageMaker 提供的监控工具的信息，请参阅。监控 Amazon SageMaker (p. 2810).

使用指标监控多容器模型

要监控推理管道中的多容器模型，请使用 Amazon CloudWatch。CloudWatch 收集并将数据处理为便于读取的近乎实时的指标。SageMaker写入训练作业和终端节点 CloudWatch 中的指标和日志`AWS/SageMaker` 命名空间.

以下各表列出以下内容的指标和维度。

- 终端节点调用
- 训练作业、批量转换作业和终端节点实例

维度 是用于唯一标识指标的名称/值对。您可以为一个指标分配最多 10 个维度。有关使用 CloudWatch 监控的更多信息，请参阅使用 Amazon CloudWatch 监控 Amazon SageMaker (p. 2810).

终端节点调用指标

这些区域有： `AWS/SageMaker`命名空间包括以下：请求来自调用的指标`InvokeEndpoint`.

指标每 1 分钟报告一次。

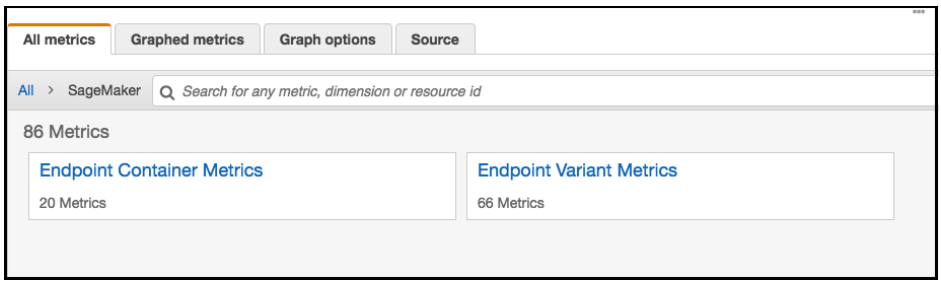
指标	说明
`Invocation4XXErrors`	模型为其返回 `4xx` HTTP 响应代码的 `InvokeEndpoint` 请求的数量。对于每个`4xx`响应如下， SageMaker 发送`1`. 单位：None（无） 有效统计数据：`Average`、`Sum`
`Invocation5XXErrors`	模型为其返回 `5xx` HTTP 响应代码的 `InvokeEndpoint` 请求的数量。对于每个`5xx`响应如下， SageMaker 发送`1`. 单位：None（无） 有效统计数据：`Average`、`Sum`
`Invocations`	发送到模型终端节点的 `number of InvokeEndpoint` 请求。 要获取发送到模型终端节点的请求总数，请使用 `Sum` 统计数据。 单位：None（无） 有效统计数据：`Sum`、`Sample Count`

指标	说明
InvocationsPerInstance	发送到模型的终端节点调用数，通过规范化InstanceCount在每个ProductionVariant. SageMaker 发送 1/numberOfInstances作为每个请求的值，其中numberOfInstances是用于的活动实例数ProductionVariant请求时的终端节点。 单位：None（无） 有效统计数据：Sum
ModelLatency	模型进行响应所需的时间。这包括以下操作所花的时间：发送请求，从模型容器中提取响应，以及完成容器中的推理。ModelLatency 是一个推理管道中所有容器所花的总时间。 单位：微秒 有效统计数据：Average、Sum、Min、Max，样品计数
OverheadLatency	由于以下操作添加到对客户端的响应用时中的时间： SageMaker 用于开销。OverheadLatency是从那个时候开始测量的 SageMaker 收到请求，直至它返回响应到客户端，减去ModelLatency. 除其他因素外，开销延迟还可能由于请求和响应负载大小、请求频率以及请求的身份验证或授权而变化。 单位：微秒 有效统计数据：Average、Sum、Min、Max、Sample Count
ContainerLatency	推理管道容器响应花费的时间。从 SageMaker 中看到.ContainerLatency包括以下操作所花的时间：发送请求，从模型容器中提取响应，以及完成容器中的推理。 单位：微秒 有效统计数据：Average、Sum、Min、Max、Sample Count

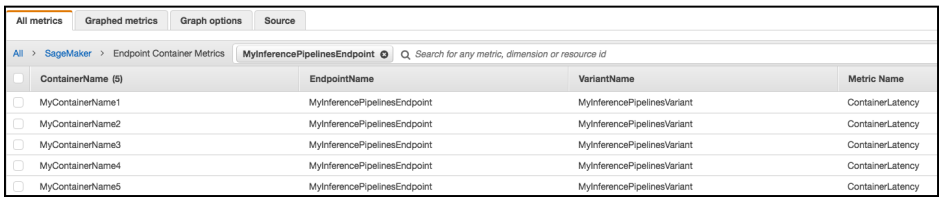
终端节点调用指标的维度

维度	说明
EndpointName, VariantName, ContainerName	筛选终端节点调用指标的ProductionVariant在指定终端节点处以及指定的变体。

对于推理管道端点， CloudWatch 将账户中的每容器延迟指标列为端节点容器指标和终端节点变体指中的SageMaker命名空间，如下所示。该 ContainerLatency 指标仅适用于推理管道。



对于每个终端节点和每个容器，延迟指标显示的名称：容器、终端节点、变体和指标。



训练作业、批量转换作业和终端节点实例指标

命名空间 `/aws/sagemaker/TrainingJobs`、`/aws/sagemaker/TransformJobs` 和 `/aws/sagemaker/Endpoints` 包括以下用于训练作业和终端节点实例的指标。

指标每 1 分钟报告一次。

指标	说明
`CPUUtilization`	实例上运行的容器所使用的 CPU 单位的百分比。值范围从 0% 到 100%，并且乘以 CPU 数目。例如，如果有四个 CPU，`CPUUtilization` 可以介于 0% 到 400% 之间。 对于训练作业，`CPUUtilization` 是实例上运行的算法容器的 CPU 利用率。 对于批量转换作业，`CPUUtilization` 是实例上运行的转换容器的 CPU 利用率。 对于多容器模型，`CPUUtilization` 是实例上运行的所有容器的 CPU 利用率总和。 适用于终端节点变体， `CPUUtilization`是实例上运行的所有容器的 CPU 利用率总和。 单位：百分比
`MemoryUtilization`	实例上运行的容器所使用的内存的百分比。此值范围从 0% 到 100%。 对于训练作业，`MemoryUtilization` 是实例上运行的算法容器所使用的内存。 对于批量转换作业，`MemoryUtilization` 是实例上运行的转换容器所使用的内存。 对于多容器模型，`MemoryUtilization` 是实例上运行的所有容器的所使用的内存总和。 对于终端节点变体，`MemoryUtilization` 是实例上运行的所有容器所使用的内存总和。 单位：百分比
`GPUUtilization`	在实例上运行的容器所使用的 GPU 单元的百分比。`GPUUtilization` 范围从 0% 到 100%，并乘以 GPU 的数目。例如，如果有四个 GPU，`GPUUtilization` 可以介于 0% 到 400% 之间。 对于训练作业，`GPUUtilization` 是实例上运行的算法容器所使用的 GPU。 对于批量转换作业，`GPUUtilization` 是实例上运行的转换容器所使用的 GPU。

指标	说明
	对于多容器模型，GPUUtilization 是实例上运行的所有容器的所使用的 GPU 总和。 对于终端节点变体，GPUUtilization 是实例上运行的所有容器所使用的 GPU 总和。 单位：百分比
GPUMemoryUtilization	实例上运行的容器所使用的 GPU 内存的百分比。GPUMemoryUtilization 范围从 0% 到 100%，并乘以 CPU 的数目。例如，如果有四个 GPU，GPUMemoryUtilization 可以介于 0% 到 400% 之间。 对于训练作业，GPUMemoryUtilization 是实例上运行的算法容器所使用的 GPU 内存。 对于批量转换作业，GPUMemoryUtilization 是实例上运行的转换容器所使用的 GPU 内存。 对于多容器模型，GPUMemoryUtilization 是实例上运行的所有容器的所用的 GPU 总和。 对于终端节点变体，GPUMemoryUtilization 是实例上运行的所有容器所使用的 GPU 内存总和。 单位：百分比
DiskUtilization	实例上运行的容器所使用的磁盘空间的百分比。 DiskUtilization 范围从 0% 到 100%。批量转换作业不支持此指标。 对于训练作业，DiskUtilization 是实例上运行的算法容器所使用的磁盘空间。 对于终端节点变体，DiskUtilization是实例上运行的所有提供容器所使用的磁盘空间总和。 单位：百分比

训练作业、批量转换作业和终端节点实例指标的维度

维度	说明
Host	对于训练作业，Host 格式为 [training-job-name]/algo-[instance-number-in-cluster]。使用此维度可筛选指定训练作业和实例的实例指标。此维度格式仅存在于 /aws/sagemaker/TrainingJobs 命名空间中。 对于批量转换作业，Host 格式为 [transform-job-name]/[instance-id]。使用此维度可筛选指定批量转换作业和实例的实例指标。此维度格式仅存在于 /aws/sagemaker/TransformJobs 命名空间中。 对于终端节点，Host 格式为 [endpoint-name]/[production-variant-name]/[instance-id]。使用此维度可筛选指定终端节点、变体和实例的实例指标。此维度格式仅存在于 /aws/sagemaker/Endpoints 命名空间中。

为帮助您调试训练作业、终端节点和笔记本实例生命周期配置， SageMaker 此外，还会发送算法容器、模型容器或笔记本实例生命周期配置发送到的任何内容也`stdout`要么`stderr`到 Amazon CloudWatch 日志。您可以使用此信息用于调试并分析进度。

使用日志监控推理管道

下表列出了 SageMaker 日志组和日志流。发送到亚马逊 CloudWatch

日志流是共享同一个源的一系列日志事件。每个独立的登录源 CloudWatch 构成一个独立的日志流。日志组是一组具有相同保留期、监控和访问控制设置的日志流。

日志

日志组名称	日志流名称
`/aws/sagemaker/TrainingJobs`	`[training-job-name]/algo-[instance-number-in-cluster]-[epoch_timestamp]`
`/aws/sagemaker/Endpoints/[EndpointName]`	`[production-variant-name]/[instance-id]`
	`[production-variant-name]/[instance-id]`
	`[production-variant-name]/[instance-id]/[container-name provided in the SageMaker model] (For Inference Pipelines)` 对于推理管道日志，如果你不提供容器名称，使用 CloudWatch按照在模型中提供容器的顺序，采用 “Container-1”、“Container-2”，以此类推。
`/aws/sagemaker/NotebookInstances`	`[notebook-instance-name]/[LifecycleConfigHook]`
`/aws/sagemaker/TransformJobs`	`[transform-job-name]/[instance-id]-[epoch_timestamp]`
	`[transform-job-name]/[instance-id]-[epoch_timestamp]/data-log`
	`[transform-job-name]/[instance-id]-[epoch_timestamp]/[container-name provided in the SageMaker model] (For Inference Pipelines)` 对于推理管道日志，如果你不提供容器名称，使用 CloudWatch按照在模型中提供容器的顺序，采用 “Container-1”、“Container-2”，以此类推。

> **Note**
>
> SageMaker 创建`/aws/sagemaker/NotebookInstances`在您使用生命周期配置创建笔记本实例时的日志组。有关更多信息，请参阅 使用生命周期配置脚本自定义笔记本实例 (p. 152) 。

有关 的更多信息 SageMaker 请参阅日志记录使用 Amazon CloudWatch 记录 Amazon SageMaker 事件 (p. 2820).

推理管道问题排查

要排查推理管道问题，使用 CloudWatch 日志和错误消息。如果您在包括 Amazon 的管道中使用自定义 Docker 镜像。 SageMaker 内置算法时，您可能还会遇到权限问题。要授予所需权限，请创建 Amazon Elastic Container Registry (Amazon ECR) 策略。

主题

- 推理管道的 Amazon ECR 权限问题排查 (p. 2143)

- 使用 CloudWatch 要排查的日志 SageMaker 推理管道 (p. 2143)
- 使用错误消息对推理管道进行问题排查 (p. 2144)

推理管道的 Amazon ECR 权限问题排查

当您在包括内容的管道中使用自定义 Docker 镜像时SageMaker 内置算法，您需要一个Amazon ECR 政策. 该策略允许您的 Amazon ECR 存储库向授予权限。 SageMaker 要拉取镜像。该策略必须添加以下权限：

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "allowSageMakerToPull",
            "Effect": "Allow",
            "Principal": {
                "Service": "sagemaker.amazonaws.com"
            },
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability"
            ]
        }
    ]
}
```

使用 CloudWatch 要排查的日志 SageMaker 推理管道

SageMaker 发布将推理管道部署到 Amazon 的终端节点的容器日志。 CloudWatch 在每个容器的以下路径中。

```
/aws/sagemaker/Endpoints/{EndpointName}/{Variant}/{InstanceId}/{ContainerHostname}
```

例如，此终端节点的日志会发布到以下日志组和流：

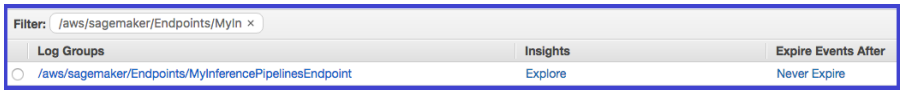
```
EndpointName: MyInferencePipelinesEndpoint
Variant: MyInferencePipelinesVariant
InstanceId: i-0179208609ff7e488
ContainerHostname: MyContainerName1 and MyContainerName2
```

```
logGroup: /aws/sagemaker/Endpoints/MyInferencePipelinesEndpoint
logStream: MyInferencePipelinesVariant/i-0179208609ff7e488/MyContainerName1
logStream: MyInferencePipelinesVariant/i-0179208609ff7e488/MyContainerName2
```

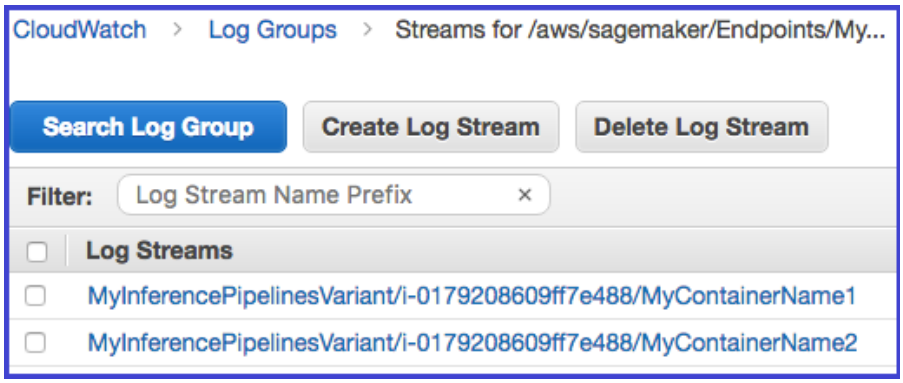
一个日志流是共享同一个源的一系列日志事件。每个独立的登录源 CloudWatch 构成一个独立的日志流。日志组是一组具有相同保留期、监控和访问控制设置的日志流。

查看日志组和流

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 在导航窗格中，选择 Logs (日志)。
3. 在 Log Groups (日志组) 中，筛选 **`MyInferencePipelinesEndpoint`**：



4. 要查看日志流，请在 CloudWatch 日志组页面，选择**`MyInferencePipelinesEndpoint`**，然后搜索日志组.



有关日志列表 SageMaker 发布，请参阅推理管道日志和指标 (p. 2138).

使用错误消息对推理管道进行问题排查

推理管道错误消息指示哪些容器失败。

如果出错 SageMaker 正在调用终端节点，该服务将返回`ModelError`（错误代码 424），指示哪个容器失败。如果请求负载（来自以前容器的响应）超过了 5 MB 的限制， SageMaker 提供了详细的错误消息，例如：

Received response from MyContainerName1 with status code 200. However, the request payload from MyContainerName1 to MyContainerName2 is 6000000 bytes, which has exceeded the maximum limit of 5 MB.

如果容器未通过 ping 运行状况检查 SageMaker 正在创建终端节点，它会返回`ClientError`并指示在上次运行状况检查中未能通过 ping 检查的所有容器。

自动扩展 Amazon SageMaker 模型

亚马逊 SageMaker 支持托管模型的自动扩展。自动扩展 动态调整为模型预配置的实例数，以响应工作负荷的变化。当工作负载增加时，自动扩展会让更多实例联机。当工作负载减小时，自动扩展会删除不必要的实例，这样您就不会为未使用的预配置实例付费。

主题

- 先决条件 (p. 2145)
- 使用控制台配置模型自动扩展 (p. 2147)
- 注册模型 (p. 2148)
- 删除扩展策略 (p. 2149)
- 应用扩展策略 (p. 2150)
- 编辑扩展策略 (p. 2152)
- 删除扩展策略 (p. 2153)
- 查询端点自动扩缩历史记 (p. 2154)
- 更新使用自动扩展的终端节点 (p. 2155)
- 对您的自动扩展配置进行负载测试 (p. 2156)
- 使用 Amazon CloudFormation 更新 Autoscaling 策略 (p. 2157)

先决条件

必须先创建亚马逊，然后才能使用自动扩展 SageMaker 模型部署。部署的模型称为生产变体。这包括有关模型以及用于托管此模型的资源的信息。

有关部署模型终端节点的更多信息，请参阅将模型部署到 SageMaker 托管服务 (p. 69)。

要为模型启用自动扩展，您可以使用控制台、Amazon CLI，或者 Application Auto Scaling API。建议您尝试使用控制台配置模型自动扩展 (p. 2147)以熟悉这些要求并测试您的第一个自动扩展配置。使用Amazon CLI或者 Application Auto Scaling，该流程是注册模型，定义扩展策略，然后应用它。以下概述提供有关与自动扩展结合使用的先决条件和组件的更多信息。

自动扩展策略概述

要使用自动扩展，您需要定义并应用使用您分配的 Amazon CloudWatch 指标和目标值的扩展策略。自动扩展使用该策略增加或减少实例数以响应实际工作负载。

您可以使用 Amazon Web Services Management Console根据预定义的指标应用扩展策略。在枚举中定义了预定义指标，因此您可以按名称在代码中指定它或在 Amazon Web Services Management Console 中使用它。或者，您也可以使用 Amazon Command Line Interface (Amazon CLI) 或 Application Auto Scaling API 根据预定义或自定义指标应用扩展策略。

支持的扩展策略有两种类型：目标跟踪扩展和步骤扩展。建议对自动扩展配置使用目标跟踪扩展策略。您需要指定预定义或自定义的指标并为指标指定目标值，以配置目标跟踪扩展策略。有关使用 Application Auto Scaling 目标跟踪扩展策略的更多信息，请参阅目标跟踪扩展策略.

当您需要高级配置时，您可以使用步进扩展，如指定要在什么条件下部署多少个实例。否则，首选使用目标跟踪扩展，因为它将完全自动执行。有关使用 Application Auto Scaling 步进扩展策略的更多信息，请参阅步进扩展策略.

扩展策略包含以下组件：

- 目标指标 — 亚马逊 CloudWatch 指标 SageMaker 自动扩展用于确定何时扩展时间以及扩展多少。
- 最小和最大容量 — 用于扩展的最小和最大实例数。
- 冷却时段 —— 在一个过去后的时长 (秒)缩放或缩放但活动在另一个扩展活动开始前完成。
- 必需权限 — 执行自动扩展操作所需的权限。
- 服务相关角色 —Amazon Identity and Access Management(IAM) 角色与特定相关的角色Amazon服务。服务相关角色包含服务调用其他服务所需的一切权限。Amazon服务代表您。 SageMaker 自动扩展会自动生成此角色，`AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint`，对你来说。

用于自动扩展的目标指标

亚马逊 CloudWatch 警报触发扩展策略，该策略计算如何根据您设置的指标和目标值调整扩展。扩展策略根据需要添加或删除终端节点实例，将指标保持在指定的目标值或接近指定的目标值。此外，扩展策略还可在工作负载更改时根据指标的波动进行调整。扩展策略最大限度地减少了模型的可用实例数的快速波动。

例如，使用具有目标值为 70 的预定义 `InvocationsPerInstance` 指标的扩展策略可以将 `InvocationsPerInstance` 保持在或接近 70。

最小和最大容量

您可以为模型指定最大终端节点实例数。最大值必须等于或大于为最小端点实例数指定的值。 SageMaker 自动扩展不会对此值强制限制。

您还必须为模型指定最小实例数。此值必须至少为 1，并且等于或小于为最大模型实例数指定的值。

要确定典型流量所需的最小和最大实例数，请使用模型的预期通信速率测试自动扩展配置。

Important

在无流量时横向缩减：如果变体的流量变为零， SageMaker 自动扩展到指定的最小实例数。在本例中， SageMaker 发出值为零的指标。最小实例数必须为 1 或更高。

冷却时间

通过添加冷却时间来优化扩展策略的响应能力。冷却时间 控制模型缩减（通过减少实例）或扩大（通过增加实例）的时间。具体方法为阻止后续缩减或扩展请求，直至冷却时间到期。这会减慢缩减请求的实例删除，并减慢扩展请求的实例创建。冷却时间有助于确保扩展策略在上一个扩展活动生效之前不会启动或终止其他实例。在自动扩展使用扩展策略动态扩展后，它会等待冷却时间完成，然后再继续扩展活动。

您可以在自动扩展策略中配置冷却时间。您可以指定以下冷却时间：

- 缩减活动会减少实例的数量。缩减冷却时间指定在完成一个缩减活动后开始另一个缩减活动之前等待的时间 (秒)。
- 扩展活动会增加实例的数量。扩展冷却时间指定在完成一个扩展活动后开始另一个扩展活动之前等待的时间 (秒)。

如果您不指定缩减或扩展冷却时间，则自动缩放会使用默认值，即每个 300 秒钟。

如果在您测试自动扩展配置时添加或删除实例太快，请考虑增大此值。如果去往模型的流量有很多峰值，或者如果有，则可以看到此行为。为一个变体定义了多个自动扩展策略。

如果无法快速添加实例以处理增加的流量，请考虑减小此值。

权限

这些区域有：`SagemakerFullAccessPolicy`IAM 策略具有执行自动扩展所需的所有 IAM 权限。有关 的更多信息 SageMaker IAM 权限，请参阅SageMaker 角色 (p. 2722).

如果您使用的是自定义权限策略，则必须包含以下权限：

```
{
  "Effect": "Allow",
  "Action": [
    "sagemaker:DescribeEndpoint",
    "sagemaker:DescribeEndpointConfig",
    "sagemaker:UpdateEndpointWeightsAndCapacities"
  ],
  "Resource": "*"
}
                {
    "Action": [
        "application-autoscaling:*"
    ],
    "Effect": "Allow",
    "Resource": "*"
}

{
  "Action": "iam:CreateServiceLinkedRole",
  "Effect": "Allow",
  "Resource":
  "arn:aws:iam::*:role/aws-service-role/sagemaker.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint",
  "Condition": {
    "StringLike": { "iam:AWSServiceName": "sagemaker.application-
autoscaling.amazonaws.com" }
  }
}
```

```
{
  "Effect": "Allow",
  "Action": [
    "cloudwatch:PutMetricAlarm",
    "cloudwatch:DescribeAlarms",
    "cloudwatch:DeleteAlarms"
  ],
  "Resource": "*"
}
```

服务相关角色

自动扩展使用 `AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint` 服务相关角色；会自动为您创建此角色。服务相关角色是一种独特类型的 IAM 角色，它与Amazon服务直接相关。服务相关角色是由服务预定义的，具有服务代表您调用其他 Amazon 服务所需的所有权限。有关更多信息，请参阅服务相关角色。

使用控制台配置模型自动扩展

使用控制台为模型配置自动扩展

1. 打开亚马逊 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中，选择 Endpoints（端点）。
3. 选择要配置的终端节点。
4. 对于 Endpoint runtime settings (终端节点运行时设置)，选择您要配置的模型。
5. 对于 Endpoint runtime settings (终端节点运行时设置)，选择 Configure autoscaling (配置自动扩展)。

   此时将显示 Configure variant automatic scaling (配置变体自动扩展) 页面。
6. 对于 Minimum capacity (最小容量)，键入您希望扩展策略维护的最小实例数。至少需要 1 个实例。
7. 对于 Maximum capacity (最大容量)，键入您希望扩展策略维护的最大实例数。
8. 对于 target value (目标值)，键入模型每分钟每个实例的平均调用次数。要确定该值，请按照负载测试 (p. 2156)中的准则进行操作。

   Application Auto Scaling 添加或删除实例以使指标接近指定的值。
9. 对于 Scale-in cool down (seconds) (缩减冷却 (秒)) 和 Scale-out cool down (seconds) (扩展冷却 (秒))，键入每个冷却时间的秒数。假设列表中的顺序是基于最重要的到最不重要的，或第一个应用的到最后一个应用的。
10. 如果您想确保您的变体扩展以解决增加的流量，但不关心在流量减少时删除实例以降低成本，请选择 Disable scale in (禁用缩减) 来禁用扩展活动，从而防止扩展策略删除变体实例。

    扩展活动始终处于启用状态，以便扩展策略可以根据需要创建终端节点实例。
11. 选择 Save（保存）。

此过程使用 Application Auto Scaling 将模型注册为可扩展目标。当您注册模型时，Application Auto Scaling 执行验证检查以确保以下内容：

- 该模型存在
- 权限足够
- 您没有注册具有可突增性能实例 (如 T2) 的变体。

  Note

  SageMaker 不支持可突增发实例 (如 T2) 的自动扩展，因为它们已经允许在增加的工作负载下增加容量。有关可突增性能实例的信息，请参阅 Amazon EC2 实例类型。

注册模型

您可以使用为模型添加自动扩展Amazon CLI或者 Application Auto Scaling API。首先必须注册模型，然后必须定义自动扩展策略。

使用 Amazon CLI 注册模型

使用 Amazon CLI，您可以根据预定义或自定义的指标来配置自动扩展。

要注册终端节点，请使用具有以下参数的 `register-scalable-target` Amazon CLI 命令：

- `--service-namespace`— 将该值设置为`sagemaker`.
- `--resource-id`— 模型的资源标识符（特别是生产变体）。对于该参数，资源类型为 `endpoint`，唯一标识符为产品变体的名称。例如，`endpoint/MyEndpoint/variant/MyVariant`。
- `--scalable-dimension`— 将该值设置为`sagemaker:variant:DesiredInstanceCount`.
- `--min-capacity`— 此模型的最小实例数。将 `min-capacity` 设置为至少 1。它必须等于或小于为 `max-capacity` 指定的值。
- `--max-capacity`— Application Auto Scaling 应管理的最大实例数。将 `max-capacity` 设置为至少 1。它必须等于或大于为 `min-capacity` 指定的值。

Example

以下示例演示如何注册名为 `MyVariant` 的模型，该模型动态扩展以具有一个到八个实例：

```
aws application-autoscaling register-scalable-target \
--service-namespace sagemaker \
--resource-id endpoint/MyEndPoint/variant/MyVariant \
--scalable-dimension sagemaker:variant:DesiredInstanceCount \
--min-capacity 1 \
--max-capacity 8
```

使用 Application Auto Scaling API 注册模型

要定义模型的扩展限制，请向 Application Auto Scaling 注册模型。Application Auto Scaling 会动态扩展生产变体实例的数量。

要使用 Application Auto Scaling 将模型注册到，请使用`RegisterScalableTarget`具有以下参数的 Application Auto Scaling API 操作：

- `ServiceNamespace`— 将该值设置为`sagemaker`.
- `ResourceID`— 生产变体的资源标识符。对于该参数，资源类型为 `endpoint`，唯一标识符为变体的名称，例如，`endpoint/MyEndPoint/variant/MyVariant`。
- `ScalableDimension`— 将该值设置为`sagemaker:variant:DesiredInstanceCount`.
- `MinCapacity`— 由 Application Auto Scaling 管理的最小实例数。必须将此值设置为至少 1，并且必须等于或小于为 `MaxCapacity` 指定的值。
- `MaxCapacity`— 由 Application Auto Scaling 管理的最大实例数。必须将此值设置为至少 1，并且必须等于或大于为 `MinCapacity` 指定的值。

Example

以下示例演示如何注册 SageMaker 动态扩展以使用一个到八个实例的生产变体：

```
POST / HTTP/1.1
    Host: autoscaling.us-east-2.amazonaws.com
    Accept-Encoding: identity
```

```
    X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
    X-Amz-Date: 20160506T182145Z
    User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
    Content-Type: application/x-amz-json-1.1
    Authorization: AUTHPARAMS

    {
        "ServiceNamespace": "sagemaker",
        "ResourceId": "endpoint/MyEndPoint/variant/MyVariant",
        "ScalableDimension": "sagemaker:variant:DesiredInstanceCount",
        "MinCapacity": 1,
        "MaxCapacity": 8
    }
```

删除扩展策略

要为扩展策略指定指标和目标值，请配置目标跟踪扩展策略。您可以使用预定义指标或自定义指标。

扩展策略配置由 JSON 块表示。您可以在文本文件中将扩展策略配置保存为 JSON 块。在调用 Amazon CLI 或 Application Auto Scaling API 时，您可以使用该文本文件。有关策略配置语法的更多信息，请参阅 Application Auto Scaling API 参考 中的 `TargetTrackingScalingPolicyConfiguration`。

可以使用以下选项定义目标跟踪扩展策略配置。

使用预定义的指标

要快速为变体定义目标跟踪扩展策略，请使用 `SageMakerVariantInvocationsPerInstance` 预定义指标。`SageMakerVariantInvocationsPerInstance` 是每分钟调用变体各个实例的平均次数。我们强烈建议您使用此指标。

要在扩展策略中使用预定义的指标，请为策略创建一个目标跟踪配置。在目标扩展配置中，包含 `PredefinedMetricSpecification` 以表示预定义的指标，并包含 `TargetValue` 以表示该指标的目标值。

Example

以下示例是变体的典型目标跟踪扩展策略配置。在此配置中，我们使用 `SageMakerVariantInvocationsPerInstance` 预定义指标来调整变体实例数，以便每个实例的 `InvocationsPerInstance` 指标都为 70。

```
{
    "TargetValue": 70.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "SageMakerVariantInvocationsPerInstance"
    }
}
```

使用自定义指标

如果您需要定义满足您的自定义要求的目标跟踪扩展策略，请定义自定义指标。您可以根据随扩展成比例变化的任何生产变体指标定义一个自定义指标。

并非所有 SageMaker 指标为目标跟踪而工作。指标必须是有效的使用率指标，它必须描述实例的繁忙程度。指标的值必须随变体实例数按反比例增大或减小。也就是说，当实例数增加时，指标的值应减小。

> **Important**
>
> 在生产中部署自动扩展之前，您必须使用自定义指标来测试自动扩展。

Example

以下示例是扩展策略的目标跟踪配置。在此配置中，对于一个名为 my-variant 变体，自定义指标根据所有实例的平均 CPU 利用率 50% 来调整变体。

```
{
    "TargetValue": 50,
    "CustomizedMetricSpecification":
    {
        "MetricName": "CPUUtilization",
        "Namespace": "/aws/sagemaker/Endpoints",
        "Dimensions": [
            {"Name": "EndpointName", "Value": "my-endpoint" },
            {"Name": "VariantName","Value": "my-variant"}
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

添加冷却时间

要添加用于扩展模型的冷却时间，请为 ScaleOutCooldown 指定一个值（以秒为单位）。类似地，要添加用于缩减模型的冷却时间，请为 ScaleInCooldown 添加一个值 (以秒为单位)。有关 ScaleInCooldown 和 ScaleOutCooldown 的更多信息，请参阅 Application Auto Scaling API 参考 中的 TargetTrackingScalingPolicyConfiguration。

Example

以下示例是扩展策略的目标跟踪配置示例。在此配置中，SageMakerVariantInvocationsPerInstance预定义指标用于根据平均值调整扩展70在该变体的所有实例中。该配置将缩减冷却时间指定为 10 分钟，并将扩展冷却时间指定为 5 分钟。

```
{
    "TargetValue": 70.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "SageMakerVariantInvocationsPerInstance"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
}
```

应用扩展策略

注册模型并定义扩展策略后，将扩展策略应用于已注册的模型。要应用扩展策略，您可以使用Amazon CLI或者 Application Auto Scaling API。

应用扩展策略 (Amazon CLI)

要将扩展策略应用于模型，请使用具有以下参数的 put-scaling-policy Amazon CLI 命令：

- --policy-name— 扩展策略的名称。
- --policy-type—将该值设置为TargetTrackingScaling.
- --resource-id— 变体的资源标识符。对于该参数，资源类型为 endpoint，唯一标识符为变体的名称。例如 endpoint/MyEndpoint/variant/MyVariant。
- --service-namespace— 将该值设置为sagemaker.
- --scalable-dimension— 将该值设置为 sagemaker:variant:DesiredInstanceCount.

- `--target-tracking-scaling-policy-configuration`—用于模型的目标跟踪扩展策略配置。

Example

以下示例与 Application Auto Scaling 结合使用以应用一个名为的目标跟踪扩展策略。`myscalablepolicy`到名为的模型（变体）`myscalablevariant`. 策略配置保存在一个名为 `config.json` 的文件中。

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id endpoint/MyEndpoint/variant/MyVariant \
    --service-namespace sagemaker \
    --scalable-dimension sagemaker:variant:DesiredInstanceCount \
    --target-tracking-scaling-policy-configuration file://config.json
```

应用扩展策略（Application Auto Scaling API）

要使用 Application Auto Scaling API 将扩展策略应用于变体，请使用`PutScalingPolicy`具有以下参数的 Application Auto Scaling API 操作：

- `PolicyName`— 扩展策略的名称。
- `ServiceNamespace`— 将该值设置为`sagemaker`.
- `ResourceID`— 变体的资源标识符。对于该参数，资源类型为 `endpoint`，唯一标识符为变体的名称。例如，`endpoint/MyEndpoint/variant/MyVariant`。
- `ScalableDimension`— 将该值设置为到`sagemaker:variant:DesiredInstanceCount`.
- `PolicyType`— 将该值设置为`TargetTrackingScaling`.
- `TargetTrackingScalingPolicyConfiguration`—用于变体的目标跟踪扩展策略配置。

Example

以下示例使用 Application Auto Scaling 应用一个名为的目标跟踪扩展策略。`myscalablepolicy`转换为名为的变体`myscalablevariant`. 它基于 `SageMakerVariantInvocationsPerInstance` 预定义指标使用策略配置。

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
X-Amz-Target: AnyScaleFrontendService.
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "sagemaker",
    "ResourceId": "endpoint/MyEndpoint/variant/MyVariant",
    "ScalableDimension": "sagemaker:variant:DesiredInstanceCount",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 70.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "SageMakerVariantInvocationsPerInstance"
        }
    }
```

```
}
```

编辑扩展策略

您可以使用编辑自动扩展策略Amazon Web Services Management Console，Amazon CLI，或者Application Auto Scaling API。

缩减

在无流量时横向缩减：如果变体的流量变为零， SageMaker 自动扩展到指定的最小实例数。在本例中，SageMaker 发出值为零的指标。最小实例数必须为 1 或更高。

禁用缩减活动

您可以禁用缩减活动以禁止目标跟踪扩展策略配置缩减变体。禁用缩减活动将禁止扩展策略删除实例，同时仍允许它根据需要创建副本。

要对模型启用或禁用缩减活动，请为 `DisableScaleIn` 指定一个布尔值。有关 `DisableScaleIn` 的更多信息，请参阅 Application Auto Scaling API 参考 中的 `TargetTrackingScalingPolicyConfiguration`。

Example

以下是扩展策略的目标跟踪配置示例，其中扩展策略将扩展而不缩减。在此配置中，`SageMakerVariantInvocationsPerInstance` 预定义指标将基于模型所在的所有实例上平均 70 次调用（推理请求）进行扩展。该配置还禁用扩展策略的缩减活动。

```
{
    "TargetValue": 70.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "SageMakerVariantInvocationsPerInstance"
    },
    "DisableScaleIn": true
}
```

扩展

要手动进行扩展，请调整最小容量。您可以使用控制台更新此值。或者，使用Amazon CLI使用`--min-capacity`参数，或使用 Application Auto Scaling API`MinCapacity`参数。

禁用扩展活动

要防止扩展，请调整最大容量。您可以使用控制台更新此值。或者，使用Amazon CLI使用`--max-capacity`参数，或使用 Application Auto Scaling API`MaxCapacity`参数。

编辑扩展策略（控制台）

要使用 Amazon Web Services Management Console编辑扩展策略，请使用与使用控制台配置模型自动扩展 (p. 2147)相同的过程。

编辑扩展策略 (Amazon CLI或 Application Auto Scaling API)

您可以使用 Amazon CLI 或 Application Auto Scaling API 按照与应用扩缩策略相同的方式编辑扩缩策略：

- 在使用 Amazon CLI 时，请在 `--policy-name` 参数中指定要编辑的策略名称。为要更改的参数指定新的值。
- 在使用 Application Auto Scaling API 时，请在`PolicyName`参数。为要更改的参数指定新的值。

有关更多信息，请参阅 应用扩展策略 (p. 2150)。

删除扩展策略

您可以使用删除扩展策略Amazon Web Services Management Console，Amazon CLI，或者 Application Auto Scaling API。如果您希望更新模型的终端节点，则必须删除扩展策略。

删除扩展策略（控制台）

删除自动扩展策略（控制台）

1. 打开亚马逊 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中，选择 Endpoints（端点）。
3. 选择要为其删除自动扩展的终端节点。
4. 对于 Endpoint runtime settings (终端节点运行时设置)，选择您要配置的变体。
5. 选择 Configure auto scaling (配置自动扩展)。
6. 选择 Deregister auto scaling (取消注册自动扩展)。

删除扩展策略 (Amazon CLI或 Application Auto Scaling API)

您可以使用Amazon CLI或使用 Application Auto Scaling API 从变体中删除扩展策略。

删除扩展策略 (Amazon CLI)

要从变体中删除扩展策略，请使用具有以下参数的 `delete-scaling-policy` Amazon CLI 命令：

- `--policy-name`— 扩展策略的名称。
- `--resource-id`— 变体的资源标识符。对于该参数，资源类型为 `endpoint`，唯一标识符为变体的名称。例如，`endpoint/MyEndpoint/variant/MyVariant`。
- `--service-namespace`— 将该值设置为`sagemaker`.
- `--scalable-dimension`— 将该值设置为`sagemaker:variant:DesiredInstanceCount`.

Example

以下示例从一个名为 `myscalablepolicy` 的变体中删除一个名为 `myscalablevariant` 的目标跟踪扩展策略。

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id endpoint/MyEndpoint/variant/MyVariant \
    --service-namespace sagemaker \
    --scalable-dimension sagemaker:variant:DesiredInstanceCount
```

删除扩展策略（Application Auto Scaling API）

要从变体中删除扩展策略，请使用`DeleteScalingPolicy`具有以下参数的 Application Auto Scaling API 操作：

- `PolicyName`— 扩展策略的名称。
- `ServiceNamespace`— 将该值设置为`sagemaker`.
- `ResourceID`— 变体的资源标识符。对于该参数，资源类型为 `endpoint`，唯一标识符为变体的名称。例如，`endpoint/MyEndpoint/variant/MyVariant`。
- `ScalableDimension`— 将该值设置为`sagemaker:variant:DesiredInstanceCount`.

Example

以下示例使用 Application Auto Scaling API 删除一个名为的目标跟踪扩展策略。myscalablepolicy来自名为的变体myscalablevariant.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "sagemaker",
    "ResourceId": "endpoint/MyEndpoint/variant/MyVariant",
    "ScalableDimension": "sagemaker:variant:DesiredInstanceCount"
}
```

查询端点自动扩缩历史记

您可以使用以下方式查看从终端节点扩展活动的状态。DescribeScalingActivities.DescribeScalingActivities提供了有关过去六周指定命名空间中的扩展活动的描述性信息。

如何查询端点自动扩缩操作

使用查询自动扩缩终端节点DescribeScalingActivities. 为此，请指定ServiceNameSpace参数。ServiceNameSpace的名称是Amazon提供资源的服务。

有效的服务名称值包括：

ecs | elasticmapreduce | ec2 | appstream | dynamodb | rds | sagemaker | custom-resource | comprehend | lambda | cassandra

在这种情况下你需要设置ServiceNameSpace到sagemaker.

使用以下命令Amazon CLI命令以查看有关您的所有详细信息sagemaker具有扩展策略的终端节点：

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace sagemaker
```

您可以使用搜索特定终端节点。ResourceId：

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace sagemaker \
    --resource-id endpoint/<endpoint_name>/variant/<variant_name>
```

运行此命令时，它会返回以下输出：

```
{
    "ActivityId": "activity-id",
    "ServiceNamespace": "sagemaker",
    "ResourceId": "endpoint/<endpoint_name>/variant/<variant_name>",
    "ScalableDimension": "sagemaker:variant:DesiredInstanceCount",
    "Description": "string",
    "Cause": "string",
    "StartTime": timestamp,
    "EndTime": timestamp,
```

```
    "StatusCode": "string",
    "StatusMessage": "string"
}
```

如何识别已阻止 AutoScaling 由于实例配额

当您扩展或添加更多实例时，您可能会达到账户级别的实例配额。您可以使用DescribeScalingActivities以检查您是否已达到实例配额。当你超出配额时，自动扩展将被阻止。

要检查您是否已达到实例配额，请使用Amazon CLI命令，如前面的示例中所示，其中您指定ResourceId：

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace sagemaker \
    --resource-id endpoint/<endpoint_name>/variant/<variant_name>
```

在返回语法中，检查StatusCode和StatusMessage键及其关联的值。StatusCode回报Failed.在StatusMessage有一条消息表明已达到账户级别的服务配额。以下为该消息具体形式的示例：

```
{
    "ActivityId": "activity-id",
    "ServiceNamespace": "sagemaker",
    "ResourceId": "endpoint/<endpoint_name>/variant/<variant_name>",
    "ScalableDimension": "sagemaker:variant:DesiredInstanceCount",
    "Description": "string",
    "Cause": "minimum capacity was set to 110",
    "StartTime": timestamp,
    "EndTime": timestamp,
    "StatusCode": "Failed",
    "StatusMessage": "Failed to set desired instance count to 110. Reason: The
    account-level service limit 'ml.xx.xxxxxx for endpoint usage' is 1000
    Instances, with current utilization of 997 Instances and a request delta
    of 20 Instances. Please contact Amazon support to request an increase for this
    limit. (Service: AmazonSageMaker; Status Code: 400;
    Error Code: ResourceLimitExceeded; Request ID: request-id)."
}
```

更新使用自动扩展的终端节点

主题

- 更新使用自动扩展的终端节点 (p. 2155)
- 删除为自动扩展配置的终端节点 (p. 2156)

更新使用自动扩展的终端节点

当您更新终端节点时，Application Auto Scaling 会检查该终端节点上的任何模型是否为自动扩展的目标。如果更新将更改作为自动扩展目标的任何模型的实例类型，则更新将失败。

在 Amazon Web Services Management Console中，您会看到一条警告，指示您必须从自动扩展中取消注册模型，然后才能对其进行更新。如果您试图通过调用UpdateEndpoint API 来更新终端节点，则调用将失败。在更新终端节点前，通过调用删除为其配置的任何扩展策略。DeleteScalingPolicy然后调用 Application Auto Scaling API 操作，DeregisterScalableTarget取消将变体注册为可扩展目标。在您更新终端节点后，可以将变体注册为可扩展目标，并将自动扩展策略附加到更新后的变体中。

有一个例外。如果您更改为自动扩展而配置的变体的模型，亚马逊 SageMaker 自动扩展允许更新。这是因为更改模型通常不会影响足够的性能来更改自动扩展行为。如果您确实要更新为自动扩展而配置的变体的模型，请确保对模型的更改不会显著影响性能和自动扩展行为。

在更新时 SageMaker 应用了自动扩展的终端节点，请完成以下步骤：

更新应用了自动扩展的终端节点

1. 通过调用 DeregisterScalableTarget，取消注册该终端节点，使其不再作为可扩展目标。
2. 由于在进行更新操作期间（或者如果您在上一步中关闭了自动扩展）会阻止自动扩展，因此，您可能需要采取额外的预防措施，以便在更新期间增加终端节点的实例数量。为此，请通过调用 `UpdateEndpointWeightsAndCapacities`，为该终端节点上托管的生产变体增加实例数量。
3. 重复调用 `DescribeEndpoint`，直到响应的 `EndpointStatus` 字段的值为 `InService`。
4. 调用 `DescribeEndpointConfig` 来获取当前终端节点配置值。
5. 通过调用 `CreateEndpointConfig` 创建新的终端节点配置。对于要保留现有实例计数或权重的生产变体，请使用调用所得到的响应中的相同变体名称。`DescribeEndpointConfig`在上一个步骤中。对于所有其他值，请使用在上一步骤中调用 `DescribeEndpointConfig` 时作为响应获取的值。
6. 通过调用 `UpdateEndpoint` 更新终端节点。指定在上一步中创建的终端节点配置作为 `EndpointConfig` 字段。如果要保留变体属性（如实例计数或权重），请将 `RetainAllVariantProperties` 参数的值设置为 `True`。这会指定具有相同名称的生产变体将使用从调用 `DescribeEndpoint` 得到的响应中的最新 `DesiredInstanceCount` 进行更新，而不考虑新 `EndpointConfig` 中 `InitialInstanceCount` 字段的值。
7. （可选）通过调用 RegisterScalableTarget 重新启用自动扩展功能。

> **Note**
>
> 只有在使用以下更改更新终端节点时，才需要步骤 1 和 7：
>
> - 更改配置了自动扩展的生产变体的实例类型。
> - 删除配置了自动扩展的生产变体。

删除为自动扩展配置的终端节点

如果您删除终端节点，Application Auto Scaling 会检查该终端节点上的任何模型是否为自动扩展的目标。如果有的话你有权限要取消注册模型，Application Auto Scaling 会取消注册这些模型为可扩展目标而不通知您。如果使用不提供 DeleteScalingPolicy 和 DeregisterScalableTarget 操作权限的自定义权限策略，您必须删除自动扩展策略并取消注册可扩展目标，然后再删除终端节点。

> **Note**
>
> 您是 IAM 用户，如果另一个 IAM 用户为终端节点上的变体配置了自动扩展，则您可能没有足够的权限删除该终端节点。

对您的自动扩展配置进行负载测试

执行负载测试以选择按所需方式工作的自动扩展配置。

有关用于优化亚马逊自动扩展的负载测试的示例 SageMaker 端点，请参阅加载测试和优化亚马逊 SageMaker 使用自动扩展的端点.

以下负载测试准则假定您使用的是一个使用预定义目标指标 `SageMakerVariantInvocationsPerInstance` 的自动扩展策略。

主题

- 确定性能特征 (p. 2156)
- 计算目标负载 (p. 2157)

确定性能特征

执行负载测试以便随并发度增加查找模型的生产变体可以处理的峰值 `InvocationsPerInstance`，以及请求的延迟。

此值取决于所选的实例类型、模型的客户端通常发送的有效负载以及模型具有的任何外部依赖项的性能。

为了找到高峰 requests-per-second (RPS) 模型的生产变体可以处理以及请求的延迟

1. 使用单个实例为模型设置终端节点。有关如何设置终端节点的信息，请参阅将模型部署到 SageMaker 托管服务 (p. 69)。
2. 使用负载测试工具可生成越来越多的并行请求，并在负载测试工具的输出中监控 RPS 和模型延迟。

   Note

   你也可以监控 requests-per-minute 而非 RPS。在这种情况下，不要在等式中乘以 60 来计算下面显示的 `SageMakerVariantInvocationsPerInstance`。

   当模型延迟增加或成功事务的比例降低时，这是您的模型可以处理的峰值 RPS。

计算目标负载

在您找到变体的性能特征后，您可以确定应允许发送到实例的最大 RPS。用于扩展的阈值必须小于此最大值。将以下公式与负载测试结合使用可确定自动缩放扩配置中 `SageMakerVariantInvocationsPerInstance` 目标指标的正确值。

```
SageMakerVariantInvocationsPerInstance = (MAX_RPS * SAFETY_FACTOR) * 60
```

其中，`MAX_RPS` 是您之前确定的最大 RPS，`SAFETY_FACTOR` 是您为确保客户端不超过最大 RPS 而选择的安全系数。乘以 60 以便从 RPS 转换为 invocations-per-minute 为此，将每分钟匹配 CloudWatch 指标 SageMaker 用于实现自动扩展（如果您测量，则无需执行此操作。 requests-per-minute 而不是每秒请求）。

Note

SageMaker 建议您使用`SAFETY_FACTOR`的 0.5。测试您的自动扩展配置，以确保它在您的模型中以您期望的方式运行，以便在终端节点上增加和减少客户流量。

使用 Amazon CloudFormation 更新 Autoscaling 策略

以下示例说明了如何使用 Amazon CloudFormation 在终端节点上启用 Autoscaling。

```
Endpoint:
  Type: "AWS::SageMaker::Endpoint"
  Properties:
    EndpointName: yourEndpointName
    EndpointConfigName: yourEndpointConfigName

ScalingTarget:
  Type: "AWS::ApplicationAutoScaling::ScalableTarget"
  Properties:
    MaxCapacity: 10
    MinCapacity: 2
    ResourceId: endpoint/MyEndPoint/variant/MyVariant
    RoleARN: arn
    ScalableDimension: sagemaker:variant:DesiredInstanceCount
    ServiceNamespace: sagemaker

ScalingPolicy:
  Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
  Properties:
    PolicyName: myscalablepolicy
    PolicyType: TargetTrackingScaling
```

```
      ScalingTargetId:
        Ref: ScalingTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 75.0
        ScaleInCooldown: 600
        ScaleOutCooldown: 30
        PredefinedMetricSpecification:
          PredefinedMetricType: SageMakerVariantInvocationsPerInstance
```

有关更多详细信息，请参阅 Amazon CloudFormation 的 AutoScalingPlans API 参考。

主机实例存储卷

在创建终端节点时，Amazon SageMaker 将 Amazon Elastic Block Store (Amazon EBS) 存储卷附加到托管该终端节点的 Amazon EC2 实例。存储卷的大小是可扩展的，存储选项分为两类：SSD 支持的存储和 HDD 支持的存储。

有关 Amazon EBS 存储和功能的更多信息，请参阅以下页面。

- Amazon EBS 功能
- Amazon EBS 用户指南

有关主机实例存储卷的完整列表，请参阅主机实例存储卷表

> Note
>
> 仅在创建时，Amazon SageMaker 将 Amazon Elastic Block Store (Amazon EBS) 存储卷附加到 Amazon EC2 实例。异步推理 (p. 2215)要么实时推理 (p. 2102)终端节点类型。

在生产中安全更新模型

在生产 ML 工作流中，数据科学家和工程师经常尝试以各种方式改进其模型，例如，执行 使用 SageMaker 执行自动模型优化 (p. 1939)，通过额外或较新的数据进行训练以及改进功能选择。对于新模型的验证过程，在新模型和旧模型之间使用生产流量执行 A/B 测试可能是有效的最后一步。在 A/B 测试中，您测试模型的不同变体，并比较每个变体的性能。如果模型的较新版本提供的性能优于以前存在的版本，请在生产中使用模型的新版本替换旧版本。

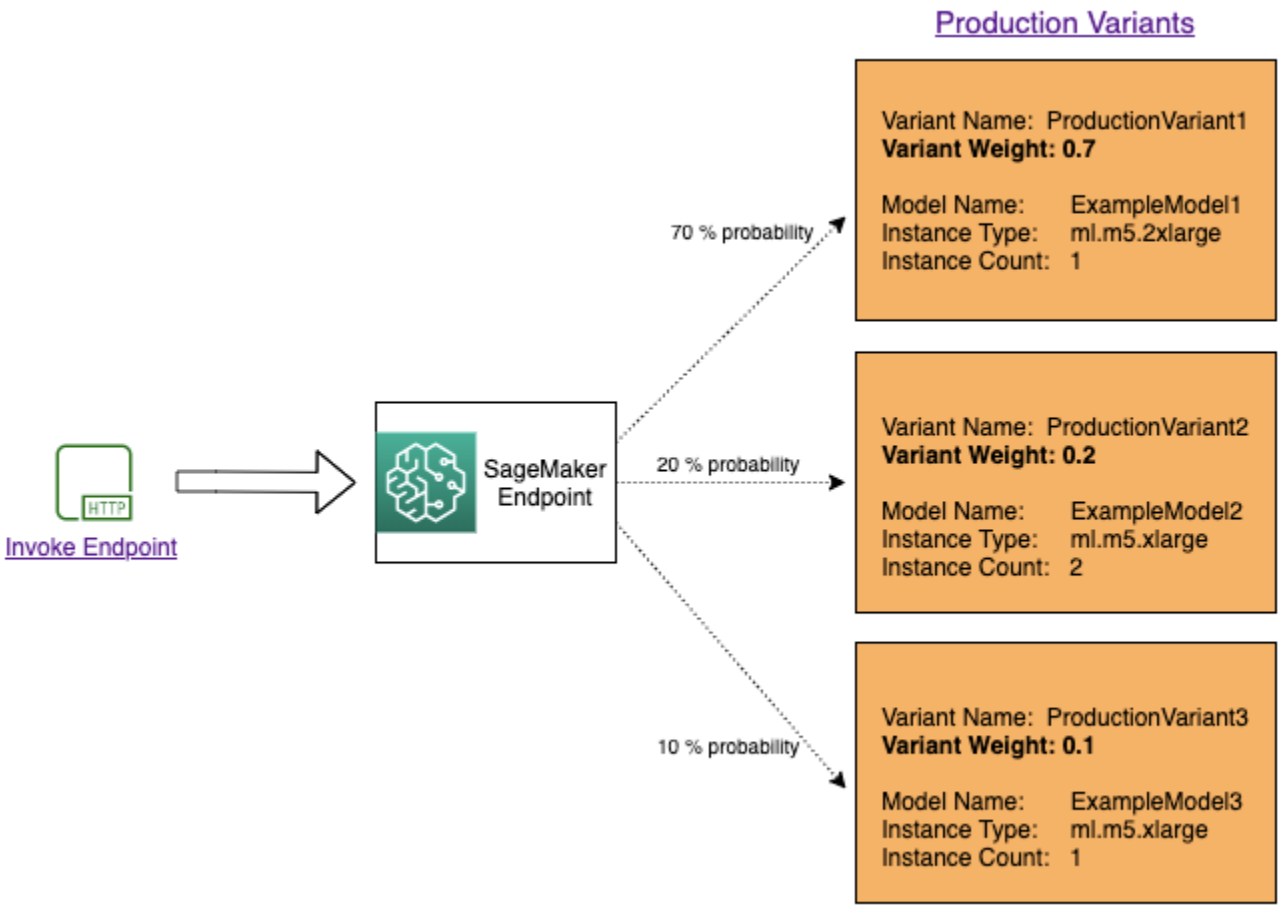
允许您 Amazon SageMaker 用生产变体在同一终端节点后面测试多个模型或模型版本。每个生产变体标识一种机器学习 (ML) 模型以及为托管该模型而部署的资源。通过使用生产变体，您可以测试使用不同数据集训练的 ML 模型、使用不同算法和 ML 框架训练的 ML 模型、部署到不同实例类型的 ML 模型或所有这些 ML 模型的任意组合。您可以为每个生产变体提供流量分配以在多个变体之间分配终端节点调用请求，也可以为每个请求直接调用特定的变体。在本主题中，我们介绍了两种测试 ML 模型的方法。

主题

- 指定流量分配以测试模型 (p. 2158)
- 调用特定的变体以测试模型 (p. 2159)
- 模型 A/B 测试示例 (p. 2160)

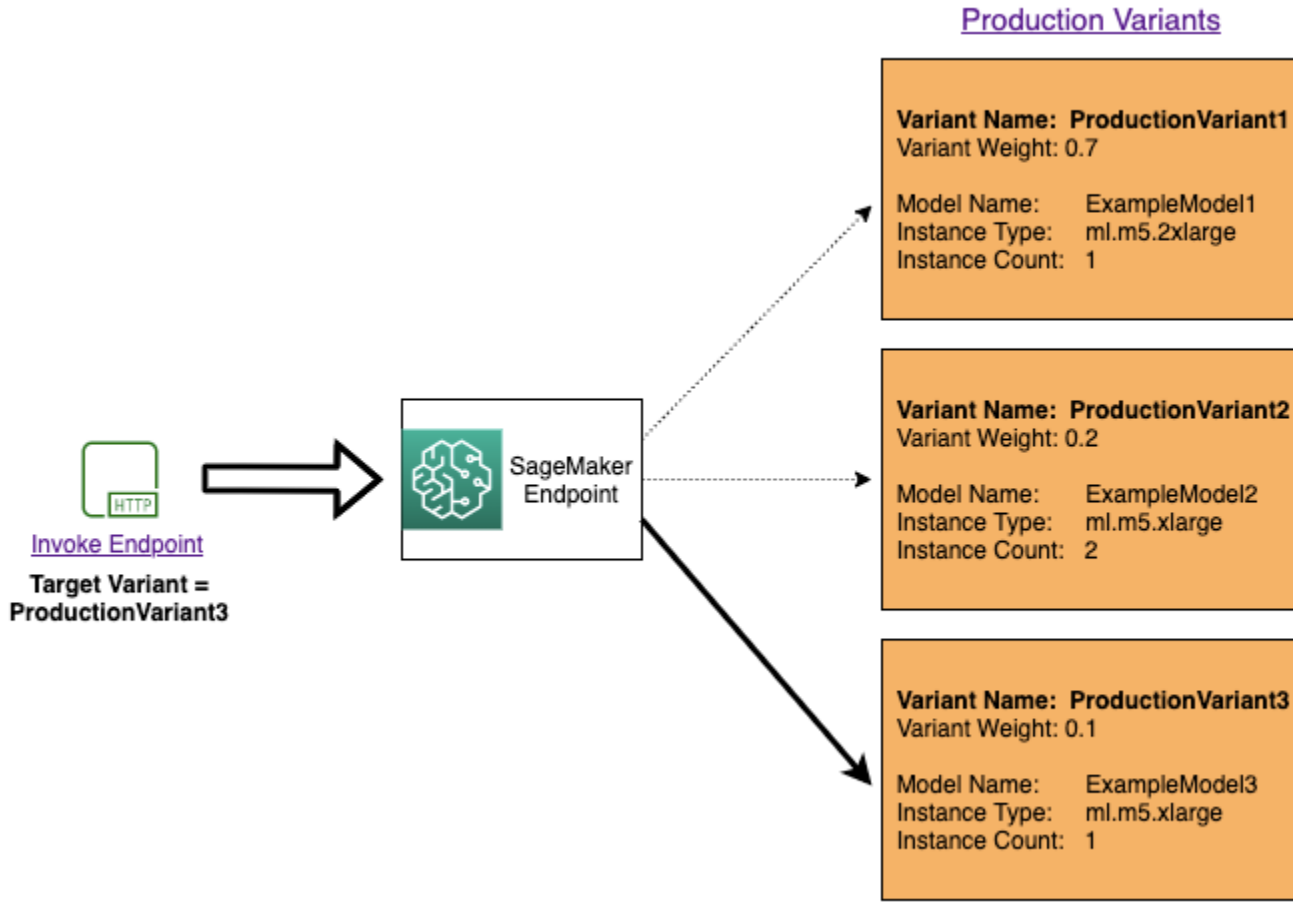
指定流量分配以测试模型

要在多个模型之间分配流量以测试这些模型，请在终端节点配置中为每个生产变体指定权重以指定路由到每个模型的流量百分比。有关信息，请参阅 CreateEndpointConfig。下图更详细地说明了它的工作方式。



调用特定的变体以测试模型

要通过为每个请求调用特定模型来测试多个模型，请通过提供`TargetVariant`调用时的参数InvokeEndpoint. SageMaker 确保请求由您指定的生产变体处理。如果您已提供流量分配并指定 `TargetVariant` 参数值，目标路由将覆盖随机流量分配。下图更详细地说明了它的工作方式。



模型 A/B 测试示例

以下示例说明了如何执行 A/B 模型测试。有关实施该示例的示例笔记本，请参阅在生产中对 ML 模型进行 A/B 测试。

第 1 步：创建并部署模型

首先，我们定义模型在 Amazon S3 中的位置。在后续步骤中部署模型时，将使用这些位置：

```
model_url = f"s3://{path_to_model_1}"
model_url2 = f"s3://{path_to_model_2}"
```

接下来，我们使用图像和模型数据创建模型对象。这些模型对象用于在终端节点上部署生产变体。这些模型是通过不同数据集、不同算法或 ML 框架以及不同超参数训练 ML 模型而开发的：

```
from sagemaker.amazon.amazon_estimator import get_image_uri

model_name = f"DEMO-xgb-churn-pred-{datetime.now():%Y-%m-%d-%H-%M-%S}"
model_name2 = f"DEMO-xgb-churn-pred2-{datetime.now():%Y-%m-%d-%H-%M-%S}"
image_uri = get_image_uri(boto3.Session().region_name, 'xgboost', '0.90-1')
image_uri2 = get_image_uri(boto3.Session().region_name, 'xgboost', '0.90-2')

sm_session.create_model(name=model_name, role=role, container_defs={
```

```
    'Image': image_uri,
    'ModelDataUrl': model_url
})

sm_session.create_model(name=model_name2, role=role, container_defs={
    'Image': image_uri2,
    'ModelDataUrl': model_url2
})
```

现在，我们创建两个生产变体，每个变体具有自己不同的模型和资源要求（实例类型和数量）。这样，您还可以在不同的实例类型上测试模型。

我们将两个变体的 initial_weight 设置为 1。这意味着，50% 的请求发送到 `Variant1`，其余 50% 的请求发送到 `Variant2`。两个变体的权重总和为 2，每个变体的权重分配为 1。这意味着，每个变体接收总流量的 1/2 或 50%。

```
from sagemaker.session import production_variant

variant1 = production_variant(model_name=model_name,
                              instance_type="ml.m5.xlarge",
                              initial_instance_count=1,
                              variant_name='Variant1',
                              initial_weight=1)

variant2 = production_variant(model_name=model_name2,
                              instance_type="ml.m5.xlarge",
                              initial_instance_count=1,
                              variant_name='Variant2',
                              initial_weight=1)
```

最后，我们准备在 SageMaker 终端节点上部署这些生产变体。

```
endpoint_name = f"DEMO-xgb-churn-pred-{datetime.now():%Y-%m-%d-%H-%M-%S}"
print(f"EndpointName={endpoint_name}")

sm_session.endpoint_from_production_variants(
    name=endpoint_name,
    production_variants=[variant1, variant2]
)
```

第 2 步：调用已部署的模型

现在，我们将请求发送到该终端节点以实时获得推理结果。我们同时使用流量分配和直接定位。

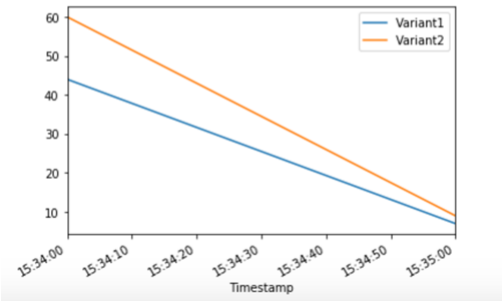
首先，我们使用在上一步中配置的流量分配。每个推理响应包含处理请求的生产变体的名称，因此，我们可以看到发送到两个生产变体的流量大致相等。

```
# get a subset of test data for a quick test
!tail -120 test_data/test-dataset-input-cols.csv > test_data/
test_sample_tail_input_cols.csv
print(f"Sending test traffic to the endpoint {endpoint_name}. \nPlease wait...")

with open('test_data/test_sample_tail_input_cols.csv', 'r') as f:
    for row in f:
        print(".", end="", flush=True)
        payload = row.rstrip('\n')
        sm_runtime.invoke_endpoint(EndpointName=endpoint_name,
                                   ContentType="text/csv",
                                   Body=payload)
        time.sleep(0.5)
```

```
print("Done!")
```

SageMaker 发出诸如下列指标`Latency`和`Invocations`针对 Amazon CloudWatch 中的每个变体。有关 SageMaker 发出的指标的完整列表，请参阅。使用 Amazon CloudWatch 监控 Amazon SageMaker (p. 2810). 让我们查询 CloudWatch 以获取每个变体的调用次数，以显示默认情况下如何在变体之间拆分调用：

Timestamp	Variant1	Variant2
2020-06-05 15:34:00+00:00	44.0	60.0
2020-06-05 15:35:00+00:00	7.0	9.0

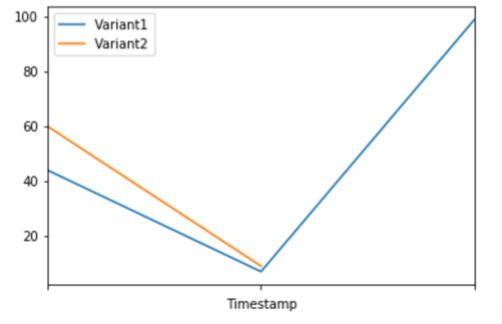


现在，让我们在 `invoke_endpoint` 调用中将 `Variant1` 指定为 `TargetVariant` 以调用特定模型版本。

```
print(f"Sending test traffic to the endpoint {endpoint_name}. \nPlease wait...")
with open('test_data/test_sample_tail_input_cols.csv', 'r') as f:
    for row in f:
        print(".", end="", flush=True)
        payload = row.rstrip('\n')
        sm_runtime.invoke_endpoint(EndpointName=endpoint_name,
                                   ContentType="text/csv",
                                   Body=payload,
                                   *TargetVariant="Variant1"*) # Notice the new parameter
        time.sleep(0.5)
```

确认所有新调用都已由`Variant1`，我们可以查询 CloudWatch 以获取每个变体的调用次数。我们看到，对于最新的调用（最新的时间戳），`Variant1` 按我们指定的方式处理了所有请求。没有针对 `Variant2` 的调用。

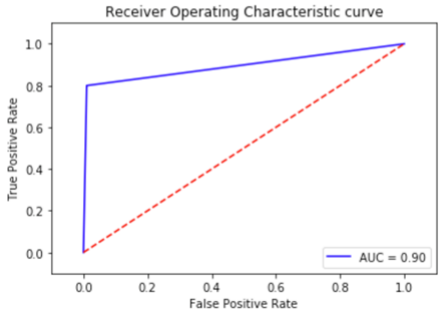
Timestamp	Variant1	Variant2
2020-06-05 15:34:00+00:00	44.0	60.0
2020-06-05 15:35:00+00:00	7.0	9.0
2020-06-05 15:36:00+00:00	99.0	NaN



第 3 步：评估模型性能

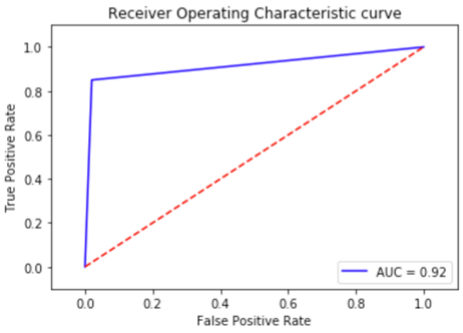
为了查看哪个模型版本的性能更好，让我们评估每个变体的准确性、精度、召回率、F1 分数以及受试者操作特征/曲线下面积。首先，让我们看一下 `Variant1` 的这些指标：

```
Accuracy: 0.9583333333333334
Precision: 0.9411764705882353
Recall: 0.8
F1 Score: 0.8648648648648648
AUC is 0.895
```



现在，让我们看一下 `Variant2` 的指标：

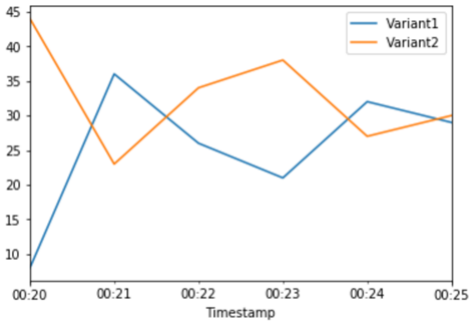
```
Accuracy: 0.9583333333333334
Precision: 0.8947368421052632
Recall: 0.85
F1 Score: 0.8717948717948718
AUC is 0.915
```



对于我们定义的大多数指标，`Variant2` 性能更好，因此，这是我们要在生产中使用的变体。

第 4 步：为最佳模型增加流量

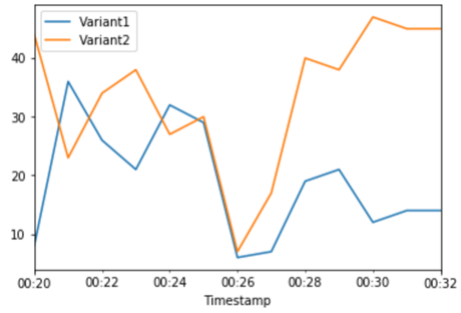
既然我们已确定 `Variant2` 性能比 `Variant1` 好，我们将更多流量转移到该变体。我们可以继续使用 `TargetVariant` 调用特定的模型变体，但更简单的方法是调用 UpdateEndpointWeightsAndCapacities 以更新分配给每个变体的权重。这会更改生产变体的流量分配，而无需更新终端节点。回顾一下设置部分，我们设置变体权重以将流量拆分为 50/50。下面有关每个变体的总调用次数的 CloudWatch 指标向我们显示每个变体的调用方式：



现在我们将 75% 的流量转移到`Variant2`通过使用为每个变体分配新的权重`UpdateEndpointWeightsAndCapacities`. SageMaker 现在将 75% 的推理请求发送到`Variant2`剩余 25% 的请求`Variant1`.

```
sm.update_endpoint_weights_and_capacities(
    EndpointName=endpoint_name,
    DesiredWeightsAndCapacities=[
        {
            "DesiredWeight": 25,
            "VariantName": variant1["VariantName"]
        },
        {
            "DesiredWeight": 75,
            "VariantName": variant2["VariantName"]
        }
    ]
)
```

有关每个变体的总调用次数的 CloudWatch 指标向我们显示的调用次数较高：`Variant2`值是`Variant1`：

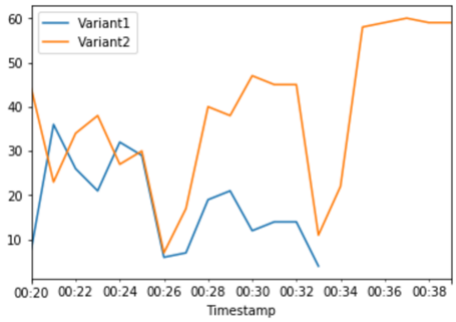


我们可以继续监控指标，在对某个变体的性能感到满意时，我们可以将 100% 的流量路由到该变体。我们使用 `UpdateEndpointWeightsAndCapacities` 更新变体的流量分配。有关的权重`Variant1`将设置为 0，而`Variant2`将设置为 1。SageMaker 现在将所有推理请求的 100% 发送到`Variant2`.

```
sm.update_endpoint_weights_and_capacities(
    EndpointName=endpoint_name,
    DesiredWeightsAndCapacities=[
        {
            "DesiredWeight": 0,
            "VariantName": variant1["VariantName"]
        },
        {
            "DesiredWeight": 1,
```

```
            "VariantName": variant2["VariantName"]
        }
    ]
)
```

有关每个变体的总调用次数的 CloudWatch 指标表明，所有推理请求由进行处理。`Variant2`并且没有处理的推理请求`Variant1`.



现在，您可以安全地更新终端节点，并将 `Variant1` 从终端节点中删除。您也可以在终端节点中添加新的变体，并执行步骤 2-4 以继续在生产中测试新模型。

最佳实践

本主题提供了在 Amazon SageMaker 中部署机器学习模型的最佳实践指南。

主题

- 在上部署模型的最佳实践 SageMaker 托管服务 (p. 2165)
- Amazon 故障排除 SageMaker 模型部署 (p. 2166)

在上部署模型的最佳实践 SageMaker 托管服务

使用托管模型时 SageMaker 托管服务，请考虑以下事项：

- 通常，客户端应用程序会向 SageMaker HTTPS 终端节点，以从部署的模型中获取推理。您也可以在测试期间从 Jupyter 笔记本向此终端节点发送请求。
- 您可以部署训练的模型 SageMaker 到您自己的部署目标。为此，您需要知道模型训练生成的模型构件的特定于算法的格式。有关输出格式的更多信息，请参阅用于训练的常见数据格式 (p. 1579)中与要使用的算法对应的章节。
- 您可以将模型的多个变体部署到同一个模型 SageMaker HTTPS 终端节点。这对于测试生产环境中模型的变体非常有用。例如，假设您已在生产环境中部署了模型。您想通过将少量流量（比如 5%）定向到新模型来测试模型的变体。为此，请创建描述模型的两个变体的终端节点配置。您在向 `ProductionVariant` 发送的请求中指定 `CreateEndPointConfig`。有关更多信息，请参阅 `ProductionVariant`。
- 您可以配置`ProductionVariant`使用应用程序 Auto Scaling。有关配置自动扩展的信息，请参阅自动扩展 Amazon SageMaker 模型 (p. 2144)。
- 您可以修改终端节点，而不用让已部署到生产环境中的模型停止服务。例如，您可以添加新的模型变体，更新现有模型变体的 ML 计算实例配置，或者更改模型变体之间的流量分配。要修改终端节点，请提供新的终端节点配置。 SageMaker 实施更改，而不会造成任何停机。有关更多信息，请参阅 `UpdateEndpoint` 和 `UpdateEndpointWeightsAndCapacities`。
- 在部署模型后更改或删除模型构件或者更改推理代码会产生不可预测的结果。如果您需要更改或删除模型构件或者更改推理代码，则通过提供新的终端节点配置来修改终端节点。一旦提供新的终端节点配置，您便可更改或删除对应于旧终端节点配置的模型构件。

- 如果您希望在整个数据集上获取推理，请考虑使用批量转换作为托管服务的替代。有关信息，请参阅。使用批量转换 (p. 2229)

跨可用区部署多个实例

托管模型时创建强大的终端节点。SageMaker 端点可以帮助保护您的应用免受可用区中断和实例故障。如果出现停机或者某个实例失败， SageMaker 自动尝试在可用区之间分布您的实例。因此，我们强烈建议您为每个生产终端节点部署多个实例。

如果您将Amazon Virtual Private Cloud (VPC)中，为 VPC 配置至少两个`Subnets`，每个都位于不同的可用区。如果出现停机或者某个实例失败，Amazon SageMaker 自动尝试在可用区之间分布您的实例。

一般而言，要实现更可靠的性能，请在不同的可用区中使用更多的小实例类型来托管您的终端节点。

Amazon 故障排除 SageMaker 模型部署

如果您在 Amazon SageMaker 中部署机器学习模型时遇到问题，请参阅以下指南。

主题

- 活动 CPU 计数中的检测错误 (p. 2166)

活动 CPU 计数中的检测错误

如果你部署 SageMaker 使用 Linux 虚拟机 (JVM) 模型，您可能会遇到检测错误，导致无法使用可用的 CPU 资源。此问题会影响支持 Java 8 和 Java 9 的某些 JVM 以及支持 Java 10 和 Java 11 的多数 JVM。这些 JVM 实施一种机制，可检测和处理在 Docker 容器（更普遍的是，在 Linux 中运行模型时的可用 CPU 计数和最大可用内存）。`taskset`命令或控制组 (cgroup)。SageMaker 部署利用了 JVM 用于管理这些资源的一些设置。目前，这会导致容器错误地检测可用 CPU 数。

SageMaker 不限制对实例上 CPU 的访问。但是，当有更多 CPU 可供容器使用时，JVM 可能会将 CPU 计数检测为 `1`。因此，JVM 会将其所有内部设置调整为像只有 `1` 个 CPU 内核可用时运行一样。这些设置会影响垃圾回收、锁定、编译器线程及其他 JVM 内部组件，从而对容器的并发性、吞吐量和延迟产生负面影响。

有关错误检测的示例，在配置的容器中 SageMaker 运行以下命令以启动 JVM（它是使用基于 Java8_191 的 JVM 进行部署的，运行以下命令以启动 JVM：

```
java -XX:+UnlockDiagnosticVMOptions -XX:+PrintActiveCpus -version
```

这会生成以下输出：

```
active_processor_count: sched_getaffinity processor count: 4
active_processor_count: determined by OSContainer: 1
active_processor_count: sched_getaffinity processor count: 4
active_processor_count: determined by OSContainer: 1
active_processor_count: sched_getaffinity processor count: 4
active_processor_count: determined by OSContainer: 1
active_processor_count: sched_getaffinity processor count: 4
active_processor_count: determined by OSContainer: 1
openjdk version "1.8.0_191"
OpenJDK Runtime Environment (build 1.8.0_191-8u191-b12-2ubuntu0.16.04.1-b12)
OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

受此问题影响的许多 JVM 选择禁用此行为，并重新建立对实例上所有 CPU 的完全访问权限。通过在启动 Java 应用程序时包含 `-XX:-UseContainerSupport` 参数，禁用这一不需要的行为并建立对所有实例 CPU 的完全访问权限。例如，按如下所示运行 `java` 命令以启动 JVM：

```
java -XX:-UseContainerSupport -XX:+UnlockDiagnosticVMOptions -XX:+PrintActiveCpus -version
```

这会生成以下输出：

```
active_processor_count: sched_getaffinity processor count: 4
active_processor_count: sched_getaffinity processor count: 4
active_processor_count: sched_getaffinity processor count: 4
active_processor_count: sched_getaffinity processor count: 4
openjdk version "1.8.0_191"
OpenJDK Runtime Environment (build 1.8.0_191-8u191-b12-2ubuntu0.16.04.1-b12)
OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

检查容器中使用的 JVM 是否支持 `-XX:-UseContainerSupport` 参数。如果支持，则在您启动 JVM 时，始终会传递该参数。这会提供对实例中所有 CPU 的访问权限。

在中间接使用 JVM 时，您可能还会遇到此问题。 SageMaker 容器。例如，使用 JVM 支持 SparkML Scala 时。`-XX:-UseContainerSupport` 参数也会影响 Java `Runtime.getRuntime().availableProcessors()` API 返回的输出。

监控模型的数据和模型质量、偏见和可解释性

亚马逊 SageMaker 模型监控器持续监控亚马逊的质量 SageMaker 生产中的机器学习模型。利用模型监控器，您可以设置在模型质量出现偏差时向您发送通知的警报。对这些偏差的早期主动检测使您能够采取纠正措施，例如重新训练模型、审计上游系统或解决质量问题，而不必手动监控模型或构建其他工具。您可以使用不需要编写代码的模型监控器预构建监控功能。您还可以通过编码来提供自定义分析，从而灵活地监控模型。

模型监视器提供以下监控类型：

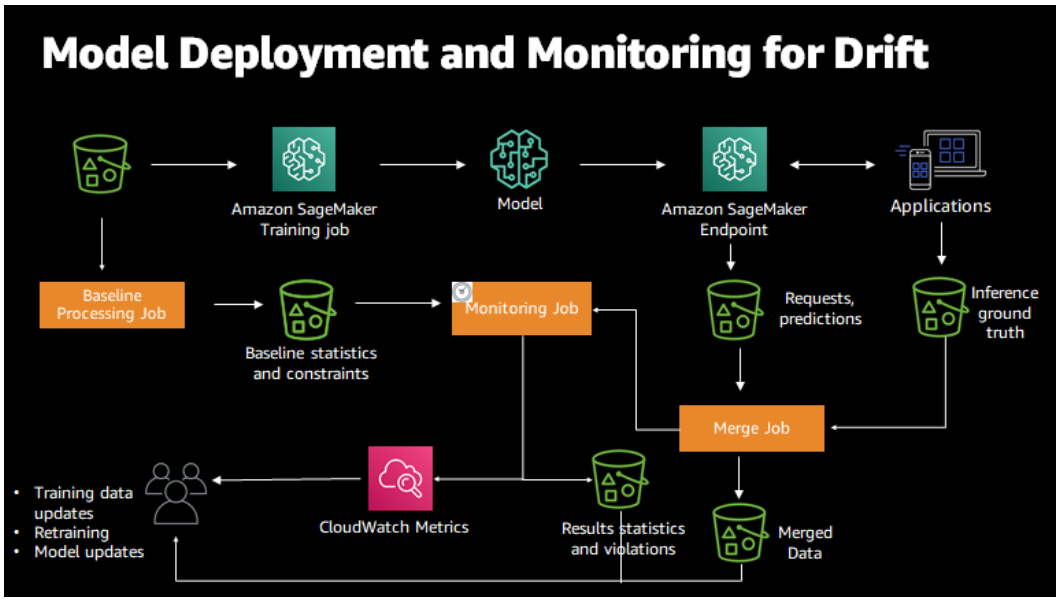
- 监控数据质量 (p. 2169)-监控数据质量的漂移。
- 监控模型质量 (p. 2173)-监控模型质量指标的偏移，例如准确性。
- 监控生产中模型的偏差漂移 (p. 2179)-监控模型预测中的偏差。
- 监控生产中模型的功能归因漂移 (p. 2182)-监控功能归因的偏移。

主题

- 模型监控器的工作原 (p. 2167)
- 监控数据质量 (p. 2169)
- 监控模型质量 (p. 2173)
- 监控生产中模型的偏差漂移 (p. 2179)
- 监控生产中模型的功能归因漂移 (p. 2182)
- 捕获数据 (p. 2186)
- 计划监控作业 (p. 2191)
- 亚马逊 SageMaker 模型监控器预构建容器 (p. 2192)
- 解释结果 (p. 2194)
- 在亚马逊可视化结果 SageMaker 工作室 (p. 2195)
- 高级主题 (p. 2201)

模型监控器的工作原

亚马逊 SageMaker Model Monitor 自动监控生产中的机器学习 (ML) 模型，并在出现质量问题时向您发送通知。模型监视器使用规则来检测模型中的漂移，并在发生漂移时向您发出警报。下图说明了此流程的工作方式。



要启用模型监控，您需要执行以下步骤，这些步骤在各种数据收集、监控和分析过程中跟随数据路径：

- 使终端节点能够捕获从传入请求到训练的 ML 模型和结果模型预测的数据。
- 从用于训练模型的数据集创建基准。基线计算指标并建议对指标的约束条件。模型的实时预测将与约束条件进行比较，如果这些预测超出了约束值，则将其报告为违规情况。
- 创建一个监控计划，指定要收集哪些数据、数据收集频率、数据分析方式以及生成哪些报告。
- 检查将最新数据与基准进行比较的报告，并观察报告的任何违规行为以及来自 Amazon CloudWatch 的指标和通知。

注意

- 模型监视器目前仅支持表格数据。
- Model Monitor 目前仅支持托管单个模型的终端节点，不支持监控多模型终端节点。有关使用多模型终端节点的信息，请参阅在一个端点后面的一个容器中托管多个模 (p. 2112)。
- 模型监控器支持监控推理管道，但捕获和分析数据是针对整个管道而不是针对管道中的各个容器完成的。
- 为了防止对推理请求的影响，Data Capture 停止捕获高磁盘使用率的请求。建议将磁盘利用率保持在 75% 以下，以确保数据捕获继续捕获请求。
- 如果你启动 SageMaker Studio 在自定义 Amazon VPC 中，您需要创建 VPC 终端节点以使模型监视器能够与 Amazon S3 和 CloudWatch 进行通信。有关 VPC 终端节点的信息，请参阅VPC 终端节点中的Amazon Virtual Private Cloud 用户指南. 有关启动的信息 SageMaker 自定义 VPC 中的 Studio，请参阅Connect (连接) SageMaker VPC 中的 Studio 笔记本电脑到外部资源 (p. 2781).

模型监控器示例笔记本

对于带您完成完整操作的示例笔记本 end-to-end 模型监视器的工作流程，请参阅亚马逊简介 SageMaker 模型监控器.

有关可视化监控计划中选定执行的 statistics.json 文件的示例笔记本，请参阅模型监控器可视化。

有关向您展示如何创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147). 在您创建笔记本实例并打开该实例并打开该实例后，请选

择SageMaker 示例选项卡以查看所有 SageMaker 示例。要打开笔记本，请选择笔记本使用选项卡并选择创建副本.

监控数据质量

数据质量监控自动监控生产中的机器学习 (ML) 模型，并在出现数据质量问题时向您发送通知。生产中的 ML 模型必须对实际数据进行预测，而这些数据并不像大多数训练数据集那样经过了精心策划。如果模型在生产过程中接收到的数据的统计性质偏离了训练所依据的基准数据的性质，则模型将开始失去其预测的准确性。亚马逊 SageMaker Model Monitor 使用规则来检测数据漂移，并在发生数据漂移时提醒. 要监控数据质量，请按照下列步骤操作：

- 启用数据捕获。这将捕获来自实时推理终端节点的推理输入和输出，并将数据存储在 Amazon S3 中。有关更多信息，请参阅 捕获数据 (p. 2186)。
- 创建基准。在此步骤中，您将运行基线作业来分析您提供的输入数据集。基线使用该基线计算每个要素的基线架构约束和统计信息德渠，一个基于 Apache Spark 构建的开源库，用于衡量大型数据集中的数据质量。有关更多信息，请参阅 创建基准 (p. 2169)。
- 定义和安排数据质量监控作业。有关更多信息，请参阅 计划监控作业 (p. 2191)。
- 查看数据质量指标。有关更多信息，请参阅 统计数据的架构（statistics.json 文件）(p. 2170)。
- 将数据质量监控与 Amazon CloudWatch 集成。有关更多信息，请参阅 CloudWatch 指标 (p. 2172)。
- 解释监控作业的结果。有关更多信息，请参阅 解释结果 (p. 2194)。
- 使用 SageMaker Studio 可实现数据质量监控和可视化结果。有关更多信息，请参阅 在亚马逊可视化结果 SageMaker 工作室 (p. 2195)。

> **Note**
>
> 亚马逊 SageMaker 模型监视器目前仅支持表格数据。

主题

- 创建基准 (p. 2169)
- 统计数据的架构（statistics.json 文件）(p. 2170)
- CloudWatch 指标 (p. 2172)
- 违规情况的架构（constraint_violations.json 文件）(p. 2172)

创建基准

统计数据和约束的基准计算需作为检测数据漂移和其他数据质量问题的标准。模型监控器提供了一个内置容器，该容器能够自动建议针对 CSV 和平面 JSON 输入的约束。该sagemaker 模型监视器分析器容器还为您提供了一系列的模型监控功能，包括针对基准的约束验证和发出亚马逊。 CloudWatch 指标。此容器基于 Spark，并通过 Deequ 构建。基线数据集中的所有列名必须符合 Spark。对于列名，请仅使用小写字符，以及_作为唯一的特殊字符。

用来训练模型的训练数据集通常是一个很好的基准数据集。训练数据集数据架构和推理数据集架构应完全匹配（特征的数量和顺序）。请注意，将预测/输出列假定为训练数据集中的第一列。从训练数据集中，你可以询问 SageMaker 建议一组基准约束，并生成描述性统计数据来探究数据。对于此示例，上传已用于训练本示例中包含的预训练模型的训练数据集。如果您已将训练数据集存储在 Amazon S3 中，则可直接指向它。

要根据训练数据集创建基线

如果您已准备好训练数据并将其存储在 Amazon S3 中，请使用通过启动基准处理作业。`DefaultModelMonitor.suggest_baseline(..)`使用亚马逊 SageMaker Python 开发工具包. 这将使用亚马逊 SageMaker 模型监控器预构建容器 (p. 2192)，它会生成基准统计数据，建议数据集的基准约束并将其写入您指定的 `output_s3_uri` 位置。

```
from sagemaker.model_monitor import DefaultModelMonitor
from sagemaker.model_monitor.dataset_format import DatasetFormat
```

```
my_default_monitor = DefaultModelMonitor(
    role=role,
    instance_count=1,
    instance_type='ml.m5.xlarge',
    volume_size_in_gb=20,
    max_runtime_in_seconds=3600,
)

my_default_monitor.suggest_baseline(
    baseline_dataset=baseline_data_uri+'/training-dataset-with-header.csv',
    dataset_format=DatasetFormat.csv(header=True),
    output_s3_uri=baseline_results_uri,
    wait=True
)
```

Note

如果提供训练数据集中的特征/列名称作为第一行，并设置`header=True`选项，如前一个代码示例所示， SageMaker 使用约束和统计信息文件中的功能名称。

数据集的基准统计数据包含在 statistics.json 文件中，建议的基准约束包含在 constraints.json 文件中，这两个文件位于您使用 `output_s3_uri` 指定的位置。

表格式数据集统计数据和约束的输出文件

文件名称	描述
`statistics.json`	此文件应具有所分析数据集中每个特征的列式统计数据。有关此文件架构的更多信息，请参阅。统计数据的架构（statistics.json 文件）(p. 2206).
`constraints.json`	此文件应对观察到的特征有约束。有关此文件架构的更多信息，请参阅。约束的架构（constraints.json 文件）(p. 2207).

这些区域有：亚马逊 SageMaker Python 开发工具包为生成基准统计数据和约束提供了易于使用的函数。但是，如果要改为直接调用处理作业来达到此目的，则需要设置`Environment`map，如以下示例所示：

```
"Environment": {
    "dataset_format": "{\"csv\": { \"header\": true}",
    "dataset_source": "/opt/ml/processing/sm_input",
    "output_path": "/opt/ml/processing/sm_output",
    "publish_cloudwatch_metrics": "Disabled",
}
```

统计数据的架构（statistics.json 文件）

亚马逊 SageMaker 模型监控器预构建的容器按列/特征统计数据进行计算。将为基准数据集以及正在分析的当前数据集计算统计数据。

```
{
    "version": 0,
    # dataset level stats
    "dataset": {
        "item_count": number
    },
    # feature level stats
    "features": [
        {
            "name": "feature-name",
```

```
        "inferred_type": "Fractional" | "Integral",
        "numerical_statistics": {
            "common": {
                "num_present": number,
                "num_missing": number
            },
            "mean": number,
            "sum": number,
            "std_dev": number,
            "min": number,
            "max": number,
            "distribution": {
                "kll": {
                    "buckets": [
                        {
                            "lower_bound": number,
                            "upper_bound": number,
                            "count": number
                        }
                    ],
                    "sketch": {
                        "parameters": {
                            "c": number,
                            "k": number
                        },
                        "data": [
                            [
                                num,
                                num,
                                num,
                                num
                            ],
                            [
                                num,
                                num
                            ][
                                num,
                                num
                            ]
                        ]
                    }#sketch
                }#KLL
            }#distribution
        }#num_stats
    },
    {
        "name": "feature-name",
        "inferred_type": "String",
        "string_statistics": {
            "common": {
                "num_present": number,
                "num_missing": number
            },
            "distinct_count": number,
            "distribution": {
                "categorical": {
                     "buckets": [
                            {
                                "value": "string",
                                "count": number
                            }
                      ]
                 }
            }
        },
        #provision for custom stats
```

```
        }
    ]
}
```

请注意以下几点：

- 预构建的容器计算KLL 素描，这是一个紧凑的分位数草图。
- 默认情况下，我们将分配具体化到 10 个存储桶中。目前，这是不可配置的。

CloudWatch 指标

您可以使用内置的亚马逊 SageMaker 的模型监控器容器 CloudWatch 指标。当`emit_metrics`选项是`Enabled`在基线约束文件中， SageMaker 对于在中的数据集中观测到的每个要素/列发出这些指标`/aws/sagemaker/Endpoints/data-metric`命名空间与`EndpointName`和`ScheduleName`维度。

对于数字字段，内置容器会发出以下 CloudWatch 指标：

- 指标：最大值 → 查询`MetricName: feature_data_{feature_name}, Stat: Max`
- 指标：最小值 → 查询`MetricName: feature_data_{feature_name}, Stat: Min`
- 指标：Sum → 查询`MetricName: feature_data_{feature_name}, Stat: Sum`
- 指标： SampleCount → 查询`MetricName: feature_data_{feature_name}, Stat: SampleCount`
- 指标：平均值 → 查询`MetricName: feature_data_{feature_name}, Stat: Average`

对于数字字段和字符串字段，内置容器会发出以下内容 CloudWatch 指标：

- 指标：完整性 → 查询`MetricName: feature_non_null_{feature_name}, Stat: Sum`
- 指标：基准偏差 → 查询`MetricName: feature_baseline_drift_{feature_name}, Stat: Sum`

违规情况的架构（constraint_violations.json 文件）

违规情况文件作为 `MonitoringExecution` 的输出生成，其中列出了针对所分析的当前数据集评估约束（在 constraints.json 文件中指定）的结果。Amazon SageMaker Model Monitor 预构建的容器提供以下违规情况检查。

```
{
    "violations": [{
      "feature_name" : "string",
      "constraint_check_type" :
              "data_type_check",
            | "completeness_check",
            | "baseline_drift_check",
            | "missing_column_check",
            | "extra_column_check",
            | "categorical_values_check"
      "description" : "string"
    }]
}
```

监控的违规的类型

违规情况检查类型	描述
`data_type_check`	如果当前执行中的数据类型与基准数据集中的数据类型不同，则会标记此违规情况。

违规情况检查类型	描述
	在基准步骤中，生成的约束会为每个列建议推断的数据类型。可以调整 `monitoring_config.datatype_check_threshold` 参数，以便调整标记为违规时的阈值。
`completeness_check`	如果当前执行中观察到的完整性（非空项目的百分比）超过了为每个特征指定的完整性阈值中指定的阈值，则会标记此违规情况。 在基准步骤中，生成的约束会建议一个完整性值。
`baseline_drift_check`	如果当前数据集和基准数据集之间计算的分布距离大于 `monitoring_config.comparison_threshold` 中指定的阈值，则会标记此违规情况。
`missing_column_check`	如果当前数据集中的列数小于基准数据集中的列数，则会标记此违规情况。
`extra_column_check`	如果当前数据集中的列数大于基准数据集中的列数，则会标记此违规情况。
`categorical_values_check`	如果当前数据集中的未知值多于基准数据集中的未知值，则会标记此违规情况。此值由 `monitoring_config.domain_content_threshold` 中的阈值决定。

监控模型质量

模型质量监控作业通过将模型的预测与模型试图预测的实际地面真相标签进行比较，来监控模型的性能。为此，模型质量监控将从实时推断中捕获的数据与存储在 Amazon S3 存储桶中的实际标签合并，然后将预测结果与实际标签进行比较。

为了衡量模型质量，模型监视器使用取决于机器学习问题类型的指标。例如，如果您的模型是针对回归问题，则评估的指标之一是均方误差 (mse)。有关用于不同机器学习问题类型的所有指标的信息，请参阅模型质量指标 (p. 2176).

模型质量监控遵循与数据质量监控相同的步骤，但增加了将 Amazon S3 的实际标签与从实时推理终端节点捕获的预测结果合并的额外步骤。要监控模型质量，请执行以下步骤：

- 启用数据捕获。这将捕获来自实时推理终端节点的推理输入和输出，并将数据存储在 Amazon S3 中。有关更多信息，请参阅 捕获数据 (p. 2186)。
- 创建基准。在此步骤中，您将运行一个基线作业，将模型的预测与基线数据集中的地面真相标签进行比较。基线作业会自动创建基线统计规则和约束，这些规则和约束条件定义了评估模型性能的阈值。有关更多信息，请参阅 创建模型质量基准 (p. 2174)。
- 定义和计划模型质量监控作业。有关更多信息，请参阅 计划模型质量监控职位 (p. 2175)。
- 采集模型监视器的地面真相标签与来自实时推理端点捕获的预测数据合并。有关更多信息，请参阅 摄取 Ground Truth 标签并将其与预测合并 (p. 2175)。
- 将模型质量监控与 Amazon CloudWatch 集成。有关更多信息，请参阅 模型质量 CloudWatch 指标 (p. 2179)。
- 解释监控作业的结果。有关更多信息，请参阅 解释结果 (p. 2194)。
- 使用 SageMaker Studio 可启用模型质量监控并可视化结果。有关更多信息，请参阅 在亚马逊可视化结果 SageMaker 工作室 (p. 2195)。

主题

- 创建模型质量基准 (p. 2174)
- 计划模型质量监控职位 (p. 2175)
- 摄取 Ground Truth 标签并将其与预测合并 (p. 2175)
- 模型质量指标 (p. 2176)
- 模型质量 CloudWatch 指标 (p. 2179)

创建模型质量基准

创建一个基线作业，将模型预测与比较基本实际情况您存储在 Amazon S3 中的基准数据集中的标签。通常，您将训练数据集用作基线数据集。基线作业计算模型的指标，并建议用于监控模型质量偏移的约束条件。

要创建基线作业，您需要有一个数据集，其中包含模型的预测以及表示数据地面真相的标签。

要创建基线作业，请使用`ModelQualityMonitor`由提供的课程 SageMaker Python SDK，然后完成以下步骤。

要创建模型质量基线作业

1. 首先，创建一个实例`ModelQualityMonitor`类。以下代码片段演示了如何执行此操作。

```
from sagemaker import get_execution_role, session, Session
from sagemaker.model_monitor import ModelQualityMonitor

role = get_execution_role()
session = Session()

model_quality_monitor = ModelQualityMonitor(
    role=role,
    instance_count=1,
    instance_type='ml.m5.xlarge',
    volume_size_in_gb=20,
    max_runtime_in_seconds=1800,
    sagemaker_session=session
)
```

2. 现在调用`suggest_baseline`方法`ModelQualityMonitor`对象来运行基准作业。以下代码段假定您有一个包含存储在 Amazon S3 中的预测和标签的基线数据集。

```
baseline_job_name = "MyBaseLineJob"
job = model_quality_monitor.suggest_baseline(
    job_name=baseline_job_name,
    baseline_dataset=baseline_dataset_uri, # The S3 location of the validation dataset.
    dataset_format=DatasetFormat.csv(header=True),
    output_s3_uri = baseline_results_uri, # The S3 location to store the results.
    problem_type='BinaryClassification',
    inference_attribute= "prediction", # The column in the dataset that contains
 predictions.
    probability_attribute= "probability", # The column in the dataset that contains
 probabilities.
    ground_truth_attribute= "label" # The column in the dataset that contains ground
 truth labels.
)
job.wait(logs=False)
```

3. 在基准作业完成后，您可以看到作业生成的约束。首先，通过调用`latest_baselining_job`方法`ModelQualityMonitor`对象。

```
baseline_job = model_quality_monitor.latest_baselining_job
```

4. 基线作业建议了约束条件，这些限制是建模监控度量的指标的阈值。如果指标超出了建议的阈值，模型监视器会报告违规情况。要查看基线作业生成的约束条件，请调用suggested_constraints基准作业的方法。以下代码段将二进制分类模型的约束加载到 Pandas 数据框中。

```
import pandas as pd
pd.DataFrame(baseline_job.suggested_constraints().body_dict["binary_classification_constraints"]).T
```

   我们建议您查看生成的约束并根据需要对其进行修改，然后再使用它们进行监视。例如，如果限制过于激进，您可能会收到的违规警报数量可能超过您想要的数量。
5. 若您满意所创建的约束，请将其作为constraints参数创建监控计划时间。有关更多信息，请参阅 计划模型质量监控职位 (p. 2175)。

建议的基准约束包含在 constraints.json 文件中，这两个文件位于您使用指定的位置。output_s3_uri. 有关此文件架构的信息，请参阅约束的架构（constraints.json 文件） (p. 2207).

计划模型质量监控职位

有关计划监控作业的一般信息，请参阅the section called “计划监控作业” (p. 2191). 对于模型质量监控，您还必须考虑地面真相标签的供应可能会延迟这一事实。

要解决这个问题，请使用偏移。模型质量职位包括StartOffset和EndOffset，这些字段是ModelQualityJobInput参数create_model_quality_job_definition方法工作方式如下：

- StartOffset-如果已指定，则作业将从开始时间中减去此时间。
- EndOffset-如果已指定，则作业将从结束时间中减去此时间。

例如，偏移的格式为-P7H，其中 7H 为 7 小时。您可以使用-P #H 或-P #D，其中 H= 小时、d= 天和 M = 分钟，# 是数字。

例如，如果你的实际真相在 1 天后开始进入，但是一周没有完成，请设置StartOffset到-P8D和EndOffset到-P1D. 然后，如果你安排作业运行在2020-01-09T13:00，它分析之间的数据2020-01-01T13:00和2020-01-08T13:00.

> **Important**
>
> 时间表的节奏应该是在下一次执行开始之前完成一次执行，这样就可以完成地面真相合并作业和监控作业从执行到完成。执行的最长运行时间分为两个作业，因此对于每小时模型质量监控作业，MaxRuntimeInSeconds指定为StoppingCondition应该不超过 1800。

摄取 Ground Truth 标签并将其与预测合并

模型质量监控将模型的预测与地面真相标签进行比较，以衡量模型的质量。为此，您需要定期为终端节点捕获的数据添加标签并将其上传到 Amazon S3。

要将地面真相标签与捕获的预测数据匹配，数据集中的每条记录都必须有唯一的标识符。地面真相数据的每个记录的结构如下所示：

```
{
  "groundTruthData": {
    "data": "1",
    "encoding": "CSV" # only CSV supported at launch, we assume "data" only consists of
 label
  },
  "eventMetadata": {
```

```
    "eventId": "aaaa-bbbb-cccc"
  },
  "eventVersion": "0"
}
```

在groundTruthData结构，eventId的值可以是以下值之一：

- eventId— 此 ID 在用户调用终端节点时触发终端节点。
- inferenceId— 呼叫者在调用终端节点时提供此 ID。

如果inferenceId存在于捕获的数据记录中，模型监视器使用它将捕获的数据与地面真相记录合并。你有责任确保inferenceId在地面真相记录中匹配inferenceId在捕获的记录中。如果inferenceId在捕获的数据中不存在，模型监视器使用eventId从捕获的数据记录中把它们与地面真相记录相匹配。

您必须将基本真实数据上传到与捕获数据具有相同路径格式的 Amazon S3 存储桶，该存储桶的格式如下：

```
s3://bucket/prefixyyyy/mm/dd/hh
```

此路径中的日期是收集地面真相标签的日期，不必与生成推理的日期相匹配。

创建并上传地面真相标签后，在创建监控作业时将标签的位置作为参数包括在内。如果您将Amazon SDK for Python (Boto3)通过将地面真相标签的位置指定为S3Uri字段中的字段GroundTruthS3Input调用中的参数create_model_quality_job_definition方法。如果您使用的是 SageMaker Python SDK，请将实际情况标签的位置指定为ground_truth_input调用中的参数create_monitoring_schedule的ModelQualityMonitor对象。

模型质量指标

模型质量监控作业根据机器学习问题类型计算不同的指标。以下部分列出了针对每种 ML 问题类型分析的指标。

> **Note**
>
> 仅当至少有 200 个样本可用时，才提供指标的标准差。Model Monitor 通过对 80% 的数据进行 5 次随机抽样、计算指标并获取这些结果的标准差来计算标准差。

回归指标

下面显示了建模质量监控器计算回归问题的指标示例。

```
"regression_metrics" : {
    "mae" : {
      "value" : 0.3711832061068702,
      "standard_deviation" : 0.0037566388129940394
    },
    "mse" : {
      "value" : 0.3711832061068702,
      "standard_deviation" : 0.0037566388129940524
    },
    "rmse" : {
      "value" : 0.609248066149471,
      "standard_deviation" : 0.003079253267651125
    },
    "r2" : {
      "value" : -1.3766111872212665,
      "standard_deviation" : 0.022653980022771227
    }
  }
```

二进制分类指标

下面显示了模型质量监视器为二进制分类问题计算的指标示例。


```
"binary_classification_metrics" : {
    "confusion_matrix" : {
      "0" : {
        "0" : 1,
        "1" : 2
      },
      "1" : {
        "0" : 0,
        "1" : 1
      }
    },
    "recall" : {
      "value" : 1.0,
      "standard_deviation" : "NaN"
    },
    "precision" : {
      "value" : 0.3333333333333333,
      "standard_deviation" : "NaN"
    },
    "accuracy" : {
      "value" : 0.5,
      "standard_deviation" : "NaN"
    },
    "recall_best_constant_classifier" : {
      "value" : 1.0,
      "standard_deviation" : "NaN"
    },
    "precision_best_constant_classifier" : {
      "value" : 0.25,
      "standard_deviation" : "NaN"
    },
    "accuracy_best_constant_classifier" : {
      "value" : 0.25,
      "standard_deviation" : "NaN"
    },
    "true_positive_rate" : {
      "value" : 1.0,
      "standard_deviation" : "NaN"
    },
    "true_negative_rate" : {
      "value" : 0.3333333333333337,
      "standard_deviation" : "NaN"
    },
    "false_positive_rate" : {
      "value" : 0.6666666666666666,
      "standard_deviation" : "NaN"
    },
    "false_negative_rate" : {
      "value" : 0.0,
      "standard_deviation" : "NaN"
    },
    "receiver_operating_characteristic_curve" : {
      "false_positive_rates" : [ 0.0, 0.0, 0.0, 0.0, 0.0, 1.0 ],
      "true_positive_rates" : [ 0.0, 0.25, 0.5, 0.75, 1.0, 1.0 ]
    },
    "precision_recall_curve" : {
      "precisions" : [ 1.0, 1.0, 1.0, 1.0, 1.0 ],
      "recalls" : [ 0.0, 0.25, 0.5, 0.75, 1.0 ]
    },
    "auc" : {
      "value" : 1.0,
```

```
      "standard_deviation" : "NaN"
    },
    "f0_5" : {
      "value" : 0.3846153846153846,
      "standard_deviation" : "NaN"
    },
    "f1" : {
      "value" : 0.5,
      "standard_deviation" : "NaN"
    },
    "f2" : {
      "value" : 0.7142857142857143,
      "standard_deviation" : "NaN"
    },
    "f0_5_best_constant_classifier" : {
      "value" : 0.29411764705882354,
      "standard_deviation" : "NaN"
    },
    "f1_best_constant_classifier" : {
      "value" : 0.4,
      "standard_deviation" : "NaN"
    },
    "f2_best_constant_classifier" : {
      "value" : 0.625,
      "standard_deviation" : "NaN"
    }
  }
```

多类别指标

下面显示了建模质量监视器计算多类分类问题的指标示例。

```
"multiclass_classification_metrics" : {
    "confusion_matrix" : {
      "0" : {
        "0" : 1180,
        "1" : 510
      },
      "1" : {
        "0" : 268,
        "1" : 138
      }
    },
    "accuracy" : {
      "value" : 0.6288167938931297,
      "standard_deviation" : 0.00375663881299405
    },
    "weighted_recall" : {
      "value" : 0.6288167938931297,
      "standard_deviation" : 0.003756638812994008
    },
    "weighted_precision" : {
      "value" : 0.6983172269629505,
      "standard_deviation" : 0.006195912915307507
    },
    "weighted_f0_5" : {
      "value" : 0.6803947317178771,
      "standard_deviation" : 0.005328406973561699
    },
    "weighted_f1" : {
      "value" : 0.6571162346664904,
      "standard_deviation" : 0.004385008075019733
    },
    "weighted_f2" : {
```

```
      "value" : 0.6384024354394601,
      "standard_deviation" : 0.003867109755267757
    },
    "accuracy_best_constant_classifier" : {
      "value" : 0.19370229007633588,
      "standard_deviation" : 0.0032049848450732355
    },
    "weighted_recall_best_constant_classifier" : {
      "value" : 0.19370229007633588,
      "standard_deviation" : 0.0032049848450732355
    },
    "weighted_precision_best_constant_classifier" : {
      "value" : 0.03752057718081697,
      "standard_deviation" : 0.001241536088657851
    },
    "weighted_f0_5_best_constant_classifier" : {
      "value" : 0.04473443104152011,
      "standard_deviation" : 0.0014460485504284792
    },
    "weighted_f1_best_constant_classifier" : {
      "value" : 0.06286421244683643,
      "standard_deviation" : 0.0019113576884608862
    },
    "weighted_f2_best_constant_classifier" : {
      "value" : 0.10570313141262414,
      "standard_deviation" : 0.002734216826748117
    }
  }
```


模型质量 CloudWatch 指标

如果你设置的值`enable_cloudwatch_metrics`到`True`当您创建监控时间表时，模型质量监控作业将所有指标发送给 Amazon CloudWatch。

模型质量指标显示在`aws/sagemaker/Endpoints/model-metrics`命名空间。有关发出的指标的列表，请参阅。模型质量指标 (p. 2176).

您可以使用 CloudWatch 指标用于在特定指标未达到您指定的阈值时创建警报。有关如何创建的说明 CloudWatch 警报，请参阅创建 CloudWatch 根据静态阈值启动警报中的亚马逊 CloudWatch 用户指南.

监控生产中模型的偏差漂移

亚马逊 SageMaker 澄清偏见监控可帮助数据科学家和机器学习工程师定期监控偏差的预测。随着模型受到监控，客户可以查看可导出的报告和图表，详细说明中的偏差 SageMaker 在 Amazon 中配置 Studio 和配置警报 CloudWatch 以便在检测到超过特定阈值的偏差时接收通知。如果训练数据与模型在部署期间看到的数据（即实时数据）不同，则可以在已部署的机器学习模型中引入或加剧偏差。实时数据分布中的这些变化可能是暂时的（例如，由于一些短暂的真实世界事件）或永久性变化。无论哪种情况，检测这些变化都可能很重要。例如，如果用于培训模型的抵押贷款利率与当前的现实世界的抵押贷款利率不同，则用于预测房价的模型的输出可能会变得偏差。使用模型监视器中的偏差检测功能， SageMaker 检测到超过特定阈值的偏差，它会自动生成您可以在其中查看的指标 SageMaker 工作室和通过 Amazon CloudWatch 提醒。

一般来说，只在 train-and-deploy 阶段可能还不够。模型部署后，部署的模型看到的数据（即实时数据）的分布可能与训练数据集中的数据分布不同。随着时间的推移，这种变化可能会在模型中引入偏 实时数据分布的变化可能是暂时的（例如，由于假日季等一些短暂的行为），也可能是永久性的。无论哪种情况，都必须检测这些变化，并酌情采取措施减少偏见。

为了检测这些变化， SageMaker 澄清提供了持续监控已部署模型的偏差指标的功能，并在指标超过阈值时发出自动警报。例如，考虑 DPPL 偏差指标。指定允许的值范围 A = (a_{min}，a_{max}），例如，在部署期间，DPPL 应该属于的间隔 (-0.1, 0.1)。与此范围的任何偏差都应该提高检测到偏差警报。与 SageMaker 澄清，你可以定期执行这些检查。

例如，您可以将支票的频率设置为 2 天。这意味着 SageMaker 澄清根据在两天窗口内收集的数据计算 DPPL 指标。在此示例中，D_{win}是模型在最近 2 天窗口中处理的数据。如果 DPPL 值 b，则发出警报$_{win}$计算 D_{win}超出允许的范围 A。这种检查是否 b 的方法$_{win}$在 A 之外可能会有点吵。D_{win}可能由很少的样本组成，而且可能不代表实时数据分发。样本数量小意味着偏差 b 的值$_{win}$计算 D_{win}可能不是一个非常可靠的估计。事实上，b 的非常高（或低）值$_{win}$可能纯粹由于偶然而被观察。为了确保从观察到的数据得出的结论 D_{win}具有统计意义， SageMaker 澄清利用置信区间。具体来说，它使用普通引导时间间隔方法来构建区间 C = (c_{min}c, c_{max}) 这样 SageMaker 澄清相信，对完整实时数据计算出的真正偏差值包含在 C 中的概率很高。现在，如果置信区间 C 与允许的范围 A 重叠， SageMaker 澄清将其解释为 “实时数据分布的偏差指标值可能在允许的范围内”。如果 C 和 A 不相交， SageMaker 澄清有信心偏差指标不在 A 中，并提出警报。

模型监控器样本

亚马逊 SageMaker 澄清提供了以下示例笔记本，展示了如何捕获实时推理数据、创建基线以监控不断变化的偏见以及检查结果：

- 监控偏差漂移和功能归因漂移亚马逊 SageMaker 澄清— 使用 Amazon SageMaker 模型监视器，用于监控随时间推移的偏差和功能归因漂移。

这款笔记本已经验证可在亚马逊上运行 SageMaker 仅限 Studio。如果您需要有关如何在亚马逊中打开笔记本电脑的说明 SageMaker 工作室，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86). 如果系统会提示您选择内核，请选择Python 3（数据科学）. 以下主题包含最后两个步骤的亮点，其中包含来自示例笔记本的代码示例。

主题

- 创建偏差漂移基线 (p. 2180)
- 计划偏差漂移监控职位 (p. 2181)
- 检查报告中的数据偏差漂移 (p. 2182)

创建偏差漂移基线

在将应用程序配置为捕获实时推理数据之后，监控偏差偏移的第一个任务是创建基线。这涉及配置数据输入、哪些组敏感、如何捕获预测、模型及其训练后偏差指标。然后你需要开始基线工作。

模型偏差监视器可以定期检测机器学习模型的偏差漂移。与其他监测类型类似，创建模型偏差监视器的标准程序首先是确定基准，然后确定监测时间表。

```
model_bias_monitor = ModelBiasMonitor(
    role=role,
    sagemaker_session=sagemaker_session,
    max_runtime_in_seconds=1800,
)
```

`DataConfig`存储有关待分析的数据集（例如，数据集文件）、其格式（即 CSV 或 jsonLine）、标题（如果有）和标签的信息。

```
model_bias_baselining_job_result_uri = f"{baseline_results_uri}/model_bias"
model_bias_data_config = DataConfig(
    s3_data_input_path=validation_dataset,
    s3_output_path=model_bias_baselining_job_result_uri,
    label=label_header,
    headers=all_headers,
    dataset_type=dataset_type,
)
```

`BiasConfig`是数据集中敏感组的配置。通常，偏差是通过计算指标并跨组进行比较来衡量的。利益集团被称为特点. 对于训练后的偏见，你还应该考虑积极的标签。

```
model_bias_config = BiasConfig(
    label_values_or_threshold=[1],
    facet_name="Account Length",
    facet_values_or_threshold=[100],
)
```

`ModelPredictedLabelConfig`指定如何从模型输出中提取预测的标签。在此示例中，选择 0.8 截止时间是为了预计客户会经常转账。对于更复杂的输出，还有一些选项，例如 “label” 是索引、名称或 JSONPath，用于在端点响应负载中定位预测标签。

```
model_predicted_label_config = ModelPredictedLabelConfig(
    probability_threshold=0.8,
)
```

`ModelConfig`是与用于推理的模型相关的配置。为了计算训练后的偏差指标，计算需要获得对所提供的模型名称的推断。为了实现这一目标，处理作业使用模型创建临时终端节点（也称为阴影端点)。计算完成后，处理作业将删除影子端点。可解释性监视器也使用此配置。

```
model_config = ModelConfig(
    model_name=model_name,
    instance_count=endpoint_instance_count,
    instance_type=endpoint_instance_type,
    content_type=dataset_type,
    accept_type=dataset_type,
)
```

现在，您可以启动基线作业了。

```
model_bias_monitor.suggest_baseline(
    model_config=model_config,
    data_config=model_bias_data_config,
    bias_config=model_bias_config,
    model_predicted_label_config=model_predicted_label_config,
)
print(f"ModelBiasMonitor baselining job: {model_bias_monitor.latest_baselining_job_name}")
```

计划的监视器会自动获取基线作业名称，然后在监视开始之前等待。

计划偏差漂移监控职位

现在，您有了基线，可以调用`create_monitoring_schedule()`方法来安排每小时监视器以使用监控时间表分析数据。如果您提交了基线作业，监视器会自动从基线作业中获取分析配置。如果跳过基线步骤或捕获数据集的性质与训练数据集不同，则必须提供分析配置。

```
model_bias_analysis_config = None
if not model_bias_monitor.latest_baselining_job:
    model_bias_analysis_config = BiasAnalysisConfig(
        model_bias_config,
        headers=all_headers,
        label=label_header,
    )
model_bias_monitor.create_monitoring_schedule(
    analysis_config=model_bias_analysis_config,
    output_s3_uri=s3_report_path,
    endpoint_input=EndpointInput(
```

```
        endpoint_name=endpoint_name,
        destination="/opt/ml/processing/input/endpoint",
        start_time_offset="-PT1H",
        end_time_offset="-PT0H",
        probability_threshold_attribute=0.8,
    ),
    ground_truth_input=ground_truth_upload_path,
    schedule_cron_expression=schedule_expression,
)
print(f"Model bias monitoring schedule: {model_bias_monitor.monitoring_schedule_name}")
```

检查报告中的数据偏差漂移

如果您无法在中生成的报告中检查监控结果 SageMaker Studio，你可以按如下方式打印出来：

```
schedule_desc = model_bias_monitor.describe_schedule()
execution_summary = schedule_desc.get("LastMonitoringExecutionSummary")
if execution_summary and execution_summary["MonitoringExecutionStatus"] in ["Completed",
 "CompletedWithViolations"]:
    last_model_bias_monitor_execution = model_bias_monitor.list_executions()[-1]
    last_model_bias_monitor_execution_report_uri =
 last_model_bias_monitor_execution.output.destination
    print(f'Report URI: {last_model_bias_monitor_execution_report_uri}')
    last_model_bias_monitor_execution_report_files =
 sorted(S3Downloader.list(last_model_bias_monitor_execution_report_uri))
    print("Found Report Files:")
    print("\n ".join(last_model_bias_monitor_execution_report_files))
else:
    last_model_bias_monitor_execution = None
    print("====STOP==== \n No completed executions to inspect further. Please wait till an
 execution completes or investigate previously reported failures.")
```

如果与基准相比存在违规情况，则会在此列出它们：

```
if last_model_bias_monitor_execution:
    model_bias_violations = last_model_bias_monitor_execution.constraint_violations()
    if model_bias_violations:
        print(model_bias_violations.body_dict)
```

In SageMaker Studio，你可以看到分析结果的可视化 CloudWatch 通过选择指标终端节点选项卡，然后双击终端节点。

监控生产中模型的功能归因漂移

生产模型的实时数据分布的偏移可能会导致功能归因值的相应变化，就像监控偏差指标时可能导致偏差偏移一样。亚马逊 SageMaker 澄清功能归因监控可帮助数据科学家和机器学习工程师定期监控功能归因偏移的预测。随着模型受到监控，客户可以在中查看可导出的报告和图表，详细说明功能属性 SageMaker 在 Amazon 中配置 Studio 和配置警报 CloudWatch 在检测到归因值超出特定阈值时接收通知。

为了用具体情况来说明这一点，请考虑大学入学的假设情况。假设我们在训练数据和实时数据中观察到以下（聚合的）功能归因值：

大学入学假设情景

功能	训练数据中的归因	实时数据中的归因
SAT 分数	0.70	0.10
GPA	0.50	0.20

功能	训练数据中的归因	实时数据中的归因
类等级	0.05 美元	0.70

从训练数据到实时数据的变化似乎很重要。功能排名已经完全逆转。与偏差偏移类似，功能归因偏差可能是由实时数据分布的变化引起的，值得仔细研究实时数据的模型行为。再次，在这些情况下，第一步是提出一个关于已经发生了漂移的警报。

我们可以通过比较单个功能的排名从训练数据到实时数据的变化来检测出漂移。除了对排名顺序的变化敏感外，我们还希望对功能的原始归因分数保持敏感。例如，鉴于从培训到实时数据的职位数量相同，在排名中的两个功能，我们希望对培训数据中归因分数较高的功能更加敏感。考虑到这些属性，我们使用标准化折扣累积收益 (NDCG) 分数来比较培训和实时数据的功能归因排名。

具体而言，假定我们有以下条件：

- F= [f_1,..., f_m]是根据培训数据中的归因分数排序的功能列表，其中M是要素的总数。例如，在我们的例子中，F= [SAT 分数、GPA、班级排名]。
- a (f)是一个函数，用于返回给定功能的训练数据的功能归因分数f. 例如，一个（SAT 分数）= 0.70。
- F′= [f'_1,..., f'_m]是根据实时数据中的归因分数排序的要素列表。例如，F′= [班级排名、GPA、SAT 分数]。

然后，我们可以将 NDCG 计算为：

$$ndCG=DG/IDCG$$

替换为

- DCG = Σ_1^M一个(f'_i) /$\log_2(i+1)$
- iDCG = Σ_1^M一个(f_i) /$\log_2(i+1)$

DCG 的数量衡量培训数据中属性较高的要素在根据实时数据计算的要素归因中的排名是否也较高。iDCG 测量的数量理想的分数而且这只是一个标准化因素，可以确保最终数量位于 [0, 1] 范围内，1 是可能的最佳值。NDCG 值为 1 表示实时数据中的功能归因排名与训练数据中的功能归因排名相同。在这个特定的例子中，由于排名有了很大的变化，因此 NDCG 值为 0.69。

In SageMaker 澄清，如果 NDCG 值低于 0.90，我们会自动发出警报。

模型监控器例笔记本

SageMaker Clarity 提供了以下示例笔记本，该笔记本展示了如何捕获实时推理数据、创建基线以监控不断变化的偏见以及检查结果：

- 监控偏差漂移和功能归因漂移亚马逊 SageMaker 澄清— 使用 Amazon SageMaker 模型监视器，用于监控随时间推移的偏差和功能归因漂移。

这款笔记本已经验证可以运行 SageMaker 仅限 Studio。如果您需要有关如何在中打开笔记本电脑的说明 SageMaker 工作室，请参阅创建或打开亚马逊 SageMaker Studio Notebook (p. 86). 如果系统会提示您选择内核，请选择Python 3（数据科学）. 以下主题包含最后两个步骤的亮点，其中包含来自示例笔记本的代码示例。

主题

- 为生产中的模型创建 SHAP 基线 (p. 2184)
- 计划功能属性漂移监控作业 (p. 2185)
- 检查生产模型中的要素属性漂移报告 (p. 2185)

为生产中的模型创建 SHAP 基线

解释通常是对比的，也就是说，它们解释了偏离基线的原因。有关可解释性基准的信息，请参阅SHAP 基准的可解释性 (p. 2073).

除了对每个实例的推断提供解释外， SageMaker 澄清还支持对机器学习模型的全局解释，以帮助您从特性角度了解模型作为一个整体的行为。 SageMaker 澄清通过在多个实例上聚合 Shapley 值来生成机器学习模型的全局解释。 SageMaker 澄清支持以下不同的聚合方式，您可以使用这些方式来定义基线：

- `mean_abs`— 所有实例的绝对 SHAP 值的均值。
- `median`— 所有实例的 SHAP 值的中位数。
- `mean_sq`— 所有实例的 SHAP 平方值的均值。

在将应用程序配置为捕获实时推理数据之后，监控要素归因偏移的第一项任务是创建一个基线进行比较。这涉及配置数据输入、哪些组敏感、如何捕获预测、模型及其训练后偏差指标。然后你需要开始基线工作。模型可解释性监视器可以解释已部署模型的预测，该模型正在生成推论并定期检测功能归因偏移。

```
model_explainability_monitor = ModelExplainabilityMonitor(
    role=role,
    sagemaker_session=sagemaker_session,
    max_runtime_in_seconds=1800,
)
```

在此示例中，可解释性基线作业将测试数据集与偏差基线作业共享，因此它使用相同的`DataConfig`，唯一的区别是作业输出 URI。

```
model_explainability_baselining_job_result_uri = f"{baseline_results_uri}/
model_explainability"
model_explainability_data_config = DataConfig(
    s3_data_input_path=validation_dataset,
    s3_output_path=model_explainability_baselining_job_result_uri,
    label=label_header,
    headers=all_headers,
    dataset_type=dataset_type,
)
```

目前 SageMaker 澄清解释器提供了 SHAP 的可扩展和高效实施，因此可解释性配置是 shapConfig，包括以下内容：

- `baseline`— 要在内核 SHAP 算法中用作基线数据集的行（至少一个）或 S3 对象 URI 的列表。格式应与数据集格式相同。每行应只包含要素列/值，并忽略标注列/值。
- `num_samples`— 内核 SHAP 算法中要使用的样本数量。此数字决定了用于计算 SHAP 值的生成合成数据集的大小。
- agg_method — 全局 SHAP 值的聚合方法。有效值如下所示：
  - `mean_abs`— 所有实例的绝对 SHAP 值的均值。
  - `median`— 所有实例的 SHAP 值的中位数。
  - `mean_sq`— 所有实例的 SHAP 平方值的均值。
- `use_logit`— 是否将 logit 函数应用于模型预测的指标。默认为 `False`。如果`use_logit`是`True`，SHAP 值将具有对数赔率单位。
- `save_local_shap_values`(bool) — 指示是否将本地 SHAP 值保存在输出位置。默认为 `False`。

```
# Here use the mean value of test dataset as SHAP baseline
test_dataframe = pd.read_csv(test_dataset, header=None)
```

```
shap_baseline = [list(test_dataframe.mean())]

shap_config = SHAPConfig(
    baseline=shap_baseline,
    num_samples=100,
    agg_method="mean_abs",
    save_local_shap_values=False,
)
```

开始基线工作。相同的model_config是必需的，因为可解释性基线作业需要创建阴影端点才能获得对生成的合成数据集的预测。

```
model_explainability_monitor.suggest_baseline(
    data_config=model_explainability_data_config,
    model_config=model_config,
    explainability_config=shap_config,
)
print(f"ModelExplainabilityMonitor baselining job:
 {model_explainability_monitor.latest_baselining_job_name}")
```

计划功能属性漂移监控作业

模型可解释性监控有助于您理解和解释机器学习模型所做的预测。当模型监视器配置为监控模型可解释性时，SageMaker 会自动检测功能相对重要性的任何偏移，并创建解释功能属性的报告。

调用create_monitoring_schedule()方法来安排每小时监视器以使用监控时间表分析数据。如果已提交基线作业，监视器将自动从基线作业中拾取分析配置。但是，如果跳过基线步骤或捕获数据集的性质与训练数据集不同，则必须提供分析配置。ModelConfig是必需的ExplainabilityAnalysisConfig出于基线工作所需的同样原因。请注意，计算要素归因只需要功能，因此您应该排除地面真相标签。

检查生产模型中的要素属性漂移报告

在默认情况下启动您设置的计划后，您需要等待其第一次执行开始，然后停止计划以避免产生费用。

要检查报告，请使用以下代码：

```
schedule_desc = model_explainability_monitor.describe_schedule()
execution_summary = schedule_desc.get("LastMonitoringExecutionSummary")
if execution_summary and execution_summary["MonitoringExecutionStatus"] in ["Completed",
 "CompletedWithViolations"]:
    last_model_explainability_monitor_execution =
 model_explainability_monitor.list_executions()[-1]
    last_model_explainability_monitor_execution_report_uri =
 last_model_explainability_monitor_execution.output.destination
    print(f'Report URI: {last_model_explainability_monitor_execution_report_uri}')
    last_model_explainability_monitor_execution_report_files =
 sorted(S3Downloader.list(last_model_explainability_monitor_execution_report_uri))
    print("Found Report Files:")
    print("\n ".join(last_model_explainability_monitor_execution_report_files))
else:
    last_model_explainability_monitor_execution = None
    print("====STOP==== \n No completed executions to inspect further. Please wait till an
 execution completes or investigate previously reported failures.")
```

如果与基准相比存在违规情况，则会在此列出它们：

```
if last_model_explainability_monitor_execution:
    model_explainability_violations =
 last_model_explainability_monitor_execution.constraint_violations()
    if model_explainability_violations:
```

```
        print(model_explainability_violations.body_dict)
```

In SageMaker Studio，你可以看到分析结果的可视化 CloudWatch 通过选择指标终端节点选项卡，然后双击终端节点。

捕获数据

将输入记录到终端节点和推理输出 SageMaker 到 Amazon S3 的实时终端节点，您可以启用名为数据捕获. 它通常用于记录可用于训练、调试和监控的信息。亚马逊 SageMaker Model Monitor 会自动解析这些捕获的数据，并将此数据中的指标与您为模型创建的基线进行比较。有关模型监控器的更多信息，请监控模型的数据和模型质量、偏见和可解释性 (p. 2167).

> Note
>
> 为了防止对推理请求的影响，Data Capture 停止捕获高磁盘使用率的请求。建议将磁盘利用率保持在 75% 以下，以确保数据捕获继续捕获请求。

要捕获数据，必须使用 SageMaker 托管服务。这要求你创建 SageMaker 模型、定义终端节点配置并创建 HTTPS 终端节点。

启用数据捕获所需的步骤都类似，无论您是否使用Amazon SDK for Python (Boto)或者 SageMaker Python 开发工具包。如果您将AmazonSDK，定义DataCaptureConfig字典以及必填字段在CreateEndpointConfig启用数据捕获的方法。如果您将 SageMaker Python SDK，导入DataCaptureConfigClass 并从此类中初始化实例。然后，将此对象传递到`DataCaptureConfig`中的参数`sagemaker.model.Model.deploy()`方法。

要使用继续执行的代码片段，请替换*#######*在示例代码中，包含您自己的信息。

如何启用数据捕获

指定数据捕获配置。可以使用此配置捕获请求负载和/或响应负载。继续的代码片段演示了如何使用Amazon SDK for Python (Boto)和 SageMaker Python 开发工具包。

> Note
>
> 您不需要使用模型监视器来捕获请求或响应有效载荷。

Amazon SDK for Python (Boto)

使用配置要捕获的数据DataCaptureConfig使用创建终端节点时使用`CreateEndpointConfig`方法。Set`EnableCapture`到布尔值 True。此外，还提供以下必备参数：

- `EndpointConfigName`：终端节点配置的名称。当你制作`CreateEndpoint`请求.
- `ProductionVariants`：您想在此终端节点上托管的型号列表。为每个模型定义字典数据类型。
- `DataCaptureConfig`：字典数据类型，您可以在其中指定一个整数值，该值与要采样的数据的初始百分比相对应 (`InitialSamplingPercentage`)、您希望存储捕获数据的 Amazon S3 URI，以及捕获选项（`CaptureOptions`列表。指定`Input`要么`Output`为了`CaptureMode`中的`CaptureOptions`列表。

您可以选择指定方式 SageMaker 应该通过将键值对参数传递给捕获的数据进行编码`CaptureContentTypeHeader`词典。

```
# Create an endpoint config name.
endpoint_config_name = '<endpoint-config-name>'

# The name of the production variant.
variant_name = '<name-of-production-variant>'

# The name of the model that you want to host.
```

```
# This is the name that you specified when creating the model.
model_name = '<The_name_of_your_model>'

instance_type = '<instance-type>'
#instance_type='ml.m5.xlarge' # Example

# Number of instances to launch initially.
initial_instance_count = <integer>

# Sampling percentage. Choose an integer value between 0 and 100
initial_sampling_percentage = <integer>



# The S3 URI of where to store captured data in S3
s3_capture_upload_path = 's3://<bucket-name>/<data_capture_s3_key>'

# Specify either Input, Output, or both
capture_modes = [ "Input",  "Output" ]
#capture_mode = [ "Input"] # Example - If you want to capture input only

endpoint_config_response = sagemaker_client.create_endpoint_config(
    EndpointConfigName=endpoint_config_name,
    # List of ProductionVariant objects, one for each model that you want to host at
 this endpoint.
    ProductionVariants=[
        {
            "VariantName": variant_name,
            "ModelName": model_name,
            "InstanceType": instance_type, # Specify the compute instance type.
            "InitialInstanceCount": initial_instance_count # Number of instances to
 launch initially.
        }
    ],
    DataCaptureConfig= {
        'EnableCapture': True, # Whether data should be captured or not.
        'InitialSamplingPercentage' : initial_sampling_percentage,
        'DestinationS3Uri': s3_capture_upload_path,
        'CaptureOptions': [{"CaptureMode" : capture_mode} for capture_mode in
 capture_modes] # Example - Use list comprehension to capture both Input and Output
    }
)
```

有关其他终端节点配置选项的更多信息，请参阅CreateEndpointConfig中的 API亚马逊 SageMaker 服务 API 参考指南.

SageMaker Python SDK

导入`DataCaptureConfig`来自sagemaker.model_监控器模块。通过设置启用数据捕获`EnableCapture`布尔值`True`.

可选为以下参数提供参数：

- `SamplingPercentage`：一个整数值，对应于要采样的数据的百分比。如果您没有提供抽样百分比， SageMaker 将采样默认值为 20 (20%) 个数据。
- `DestinationS3Uri`：Amazon S3 URI SageMaker 将用于存储捕获的数据。如果你不提供一个，SageMaker 将在中存储捕获的数据`"s3://<default-session-bucket>/ model-monitor/data-capture"`.

```
from sagemaker.model_monitor import DataCaptureConfig

# Set to True to enable data capture
```

```
enable_capture = True

# Optional - Sampling percentage. Choose an integer value between 0 and 100
sampling_percentage = <int>
# sampling_percentage = 30 # Example 30%

# Optional - The S3 URI of where to store captured data in S3
s3_capture_upload_path = 's3://<bucket-name>/<data_capture_s3_key>'

# Specify either Input, Output or both.
capture_modes = ['REQUEST','RESPONSE'] # In this example, we specify both
# capture_mode = ['REQUEST'] # Example - If you want to only capture input.

# Configuration object passed in when deploying Models to SM endpoints
data_capture_config = DataCaptureConfig(
    enable_capture = enable_capture,
    sampling_percentage = sampling_percentage, # Optional
    destination_s3_uri = s3_capture_upload_path, # Optional
    capture_options = [{"CaptureMode": capture_mode} for capture_mode in capture_modes]
)
```

部署模型

使用部署模型并创建 HTTPS 终端节点`DataCapture`已启用.

Amazon SDK for Python (Boto3)

向 SageMaker 提供端点配置。该服务会启动 ML 计算实例，并按照配置中的规定部署一个或多个模型。

拥有模型和终端节点配置后，请使用`CreateEndpoint`用于创建终端节点的 API。在中，终端节点名称必须唯一。Amazon您的区域Amazonaccount.

以下内容使用在请求中指定的终端节点配置创建终端节点。亚马逊 SageMaker 使用终端节点来配置资源和部署模型。

```
# The name of the endpoint. The name must be unique within an AWS Region in your AWS
 account.
endpoint_name = '<endpoint-name>'

# The name of the endpoint configuration associated with this endpoint.
endpoint_config_name='<endpoint-config-name>'

create_endpoint_response = sagemaker_client.create_endpoint(
                                            EndpointName=endpoint_name,
                                            EndpointConfigName=endpoint_config_name)
```

有关更多信息，请参阅 `CreateEndpoint` API。

SageMaker Python SDK

为终端节点定义名称。此为可选步骤。如果你不提供一个， SageMaker 将为你创建一个唯一的名字：

```
from datetime import datetime

endpoint_name = f"DEMO-{datetime.utcnow():%Y-%m-%d-%H%M}"
print("EndpointName =", endpoint_name)
```

使用模型对象的内置将模型部署到实时的 HTTPS 终端节点`deploy()`方法。提供要将此模型部署到的 Amazon EC2 实例类型的名称`instance_type`此字段以及要在上运行终端节点的终端节点的实例的初始数量以及`initial_instance_count`字段：

```
initial_instance_count=<integer>
# initial_instance_count=1 # Example

instance_type='<instance-type>'
# instance_type='ml.m4.xlarge' # Example

# Uncomment if you did not define this variable in the previous step
#data_capture_config = <name-of-data-capture-configuration>

model.deploy(
    initial_instance_count=initial_instance_count,
    instance_type=instance_type,
    endpoint_name=endpoint_name,
    data_capture_config=data_capture_config
)
```

查看捕获的数据

从中创建预测器对象 SageMaker Python 开发工具包预测器类。您将使用返回的对象`Predictor`在 future 的步骤中调用终端节点的类。提供终端节点的名称（之前定义为`endpoint_name`），以及分别用于序列化器和反序列化器的序列化器和反序列化器对象。有关序列化器类型的信息，请参阅序列化器中的课SageMaker Python 开发工具包.

```
from sagemaker.predictor import Predictor
from sagemaker.serializers import <Serializer>
from sagemaker.deserializers import <Deserializers>

predictor = Predictor(endpoint_name=endpoint_name,
                      serializer = <Serializer_Class>,
                      deserializer = <Deserializer_Class>)

# Example
#from sagemaker.predictor import Predictor
#from sagemaker.serializers import CSVSerializer
#from sagemaker.deserializers import JSONDeserializer

#predictor = Predictor(endpoint_name=endpoint_name,
#                      serializer=CSVSerializer(),
#                      deserializer=JSONDeserializer())
```

在接下来的代码示例场景中，我们使用我们在本地存储在名为的 CSV 文件中的示例验证数据调用终端节点。`validation_with_predictions`. 我们的示例验证集包含每个输入的标签。

WITH 语句的前几行首先打开验证集 CSV 文件，然后用逗号字符拆分文件中的每一行`","`，然后将两个返回的对象存储到标签和 input_cols 变量中。对于每一行，输入 (`input_cols`) 被传递给预测变量的 (`predictor`) 对象内置方法`Predictor.predict()`.

假设模型返回概率。整数值为 0 和 1.0 之间的概率范围。如果模型返回的概率大于 80% (0.8)，我们将预测分配一个整数值标签 1。否则，我们为预测分配一个整数值标签 0。

```
from time import sleep

validate_dataset = "validation_with_predictions.csv"

# Cut off threshold of 80%
cutoff = 0.8

limit = 200  # Need at least 200 samples to compute standard deviations
i = 0
with open(f"test_data/{validate_dataset}", "w") as validation_file:
```

```
    validation_file.write("probability,prediction,label\n")  # CSV header
    with open("test_data/validation.csv", "r") as f:
        for row in f:
            (label, input_cols) = row.split(",", 1)
            probability = float(predictor.predict(input_cols))
            prediction = "1" if probability > cutoff else "0"
            baseline_file.write(f"{probability},{prediction},{label}\n")
            i += 1
            if i > limit:
                break
            print(".", end="", flush=True)
            sleep(0.5)
print()
print("Done!")
```

由于您在前面的步骤中启用了数据捕获，因此，请求和响应负载将与其他一些元数据一起保存到您在中指定的 Amazon S3 位置。DataCaptureConfig. 向 Amazon S3 交付捕获数据可能需要几分钟时间。

通过列出 Amazon S3 中存储的数据捕获文件来查看捕获的数据。Amazon S3 路径的格式为：s3:///*{endpoint-name}/{variant-name}/yyyy/mm/dd/hh/filename*.jsonl.

预计会看到根据调用发生时间组织的来自各个时段的不同文件。运行以下命令以打印出单个捕获文件的内容：

```
print("\n".join(capture_file[-3:-1]))
```

这将返回 SageMaker 特定的 JSLINE 格式化的文件。以下是从我们使用调用的实时终端节点中获取的响应示例csv/textDATA:

```
{"captureData":{"endpointInput":{"observedContentType":"text/csv","mode":"INPUT",
"data":"69,0,153.7,109,194.0,105,256.1,114,14.1,6,1,0,0,0
```

```
    "endpointOutput": {
      "observedContentType": "text/csv; charset=character-encoding",
      "mode": "OUTPUT",
      "data": "0.023190177977085114",
      "encoding": "CSV"
    }
  },
  "eventMetadata": {
    "eventId": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "inferenceTime": "2022-02-14T17:25:06Z"
  },
  "eventVersion": "0"
}
```

计划监控作业

亚马逊 SageMaker 模型监控器使用，您可以按计划持续监控从终端节点中收集的数据。可以使用 `CreateMonitoringSchedule` API 创建具有预定义的周期性间隔的监控计划。例如，每 x 小时（x 的范围是 1 到 23）。

随着监控计划， SageMaker 可以按指定频率启动处理作业来分析给定时段内收集到的数据。 SageMaker 提供了用于对表格数据集执行分析的预构建容器。在处理作业中， SageMaker 将当前分析的数据集与基准统计数据、提供的约束进行比较，并生成违规情况报告。此外， CloudWatch 为正在分析的每个要素发出指标。或者，您可以选择自带容器，如构建您自己的容器 (p. 2204)主题中所述。

可以为先前创建的终端节点创建模型监控计划。使用基准资源（约束和统计数据）与实时流量进行比较。对于此示例，上传已用于训练本示例中包含的预训练模型的训练数据集。如果您在 Amazon S3 中已拥有此数据点，则可直接指向它。

```
# copy over the training dataset to Amazon S3 (if you already have it in Amazon S3, you
 could reuse it)
baseline_prefix = prefix + '/baselining'
baseline_data_prefix = baseline_prefix + '/data'
baseline_results_prefix = baseline_prefix + '/results'

baseline_data_uri = 's3://{}/{}'.format(bucket,baseline_data_prefix)
baseline_results_uri = 's3://{}/{}'.format(bucket, baseline_results_prefix)
print('Baseline data uri: {}'.format(baseline_data_uri))
print('Baseline results uri: {}'.format(baseline_results_uri))
```

```
training_data_file = open("test_data/training-dataset-with-header.csv", 'rb')
s3_key = os.path.join(baseline_prefix, 'data', 'training-dataset-with-header.csv')
boto3.Session().resource('s3').Bucket(bucket).Object(s3_key).upload_fileobj(training_data_file)
```

使用基准约束和统计数据为终端节点创建模型监控计划，从而与实时流量进行比较。

```
from sagemaker.model_monitor import CronExpressionGenerator
from time import gmtime, strftime

mon_schedule_name = 'DEMO-xgb-churn-pred-model-monitor-schedule-' + strftime("%Y-%m-%d-%H-
%M-%S", gmtime())
my_default_monitor.create_monitoring_schedule(
    monitor_schedule_name=mon_schedule_name,
    endpoint_input=predictor.endpoint,
    post_analytics_processor_script=s3_code_postprocessor_uri,
    output_s3_uri=s3_report_path,
    statistics=my_default_monitor.baseline_statistics(),
    constraints=my_default_monitor.suggested_constraints(),
    schedule_cron_expression=CronExpressionGenerator.hourly(),
    enable_cloudwatch_metrics=True,
```

```
)
```

描述并检查时间表：在描述之后，请注意MonitoringScheduleStatus在MonitoringScheduleSummary 返回的ListMonitoringSchedulesAPI 更改Scheduled.

```
desc_schedule_result = my_default_monitor.describe_schedule()
print('Schedule status: {}'.format(desc_schedule_result['MonitoringScheduleStatus']))
```

用于监控时间表的 cron 表达式

要提供监控计划的详细信息，请使用 ScheduleConfig，它是一个描述有关监控计划的详细信息的 cron 表达式。

亚马逊 SageMaker 模型监控器支持以下条件cron表达式：

- 若要将作业设置为每小时启动一次，请使用以下命令：

  ```
  Hourly: cron(0 * ? * * *)
  ```
- 要每天运行作业，请使用以下命令：

  ```
  cron(0 [00-23] ? * * *)
  ```

例如，以下是有效的 cron 表达式：

- 每天凌晨 12 点 (UTC)：`cron(0 12 ? * * *)`
- 每天中午 12 点 (UTC)：`cron(0 0 ? * * *)`

为了支持每 6、12 小时运行一次，模型监控器支持以下表达式：

```
cron(0 [00-23]/[01-24] ? * * *)
```

例如，以下是有效的 cron 表达式：

- 每 12 小时一次，从下午 5 点 (UTC) 开始：`cron(0 17/12 ? * * *)`
- 每 2 小时一次，从中午 12 点 (UTC) 开始：`cron(0 0/2 ? * * *)`

> **注意**
>
> - 尽管 cron 表达式设置为下午 5 点 (UTC) 开始，但请注意，从实际请求的时间到运行执行可能有 0-20 分钟的延迟。
> - 如果要按每日计划运行，请不要提供此参数。 SageMaker 选择每天跑步的时间。
> - 目前， SageMaker 仅支持 1 到 24 小时之间的小时整数费率。

亚马逊 SageMaker 模型监控器预构建容器

SageMaker 提供了一个名为sagemaker-model-monitor-analyzer它为您提供一系列模型监控功能，包括约束建议、统计数据生成、基准约束验证和发出 Amazon CloudWatch 指标。此图像基于 Spark，并使用构建德渠.

> **Note**
>
> 你不能拉内置sagemaker-model-monitor-analyzer直接图像。您可以使用sagemaker-model-monitor-analyzer使用其中一个提交基准处理或监控作业时的映像Amazon开发工具包。

使用 SageMaker Python 开发工具包（参阅image_uris.retrieve中的SageMaker Python SDK 参考指南）为您生成 ECR 镜像 URI，或直接指定 ECR 镜像 URI。预构建的映像 SageMaker 可按如下方式访问模型监控器：

```
<ACCOUNT_ID>.dkr.ecr.<REGION_NAME>.amazonaws.com/sagemaker-model-monitor-analyzer
```

例如：`159807026194.dkr.ecr.us-west-2.amazonaws.com/sagemaker-model-monitor-analyzer`。

如果您在Amazon在中国的地区，预构建的图像 SageMaker 可按如下方式访问模型监控器：

```
<ACCOUNT_ID>.dkr.ecr.<REGION_NAME>.amazonaws.com.cn/sagemaker-model-monitor-analyzer
```

下表列出了账户 ID 和相应账户 ID 支持的值。Amazon区域名称。

ACCOUNT_ID	REGION_NAME
156813124566	us-east-1
777275614652	us-east-2
890145073186	us-west-1
159807026194	us-west-2
875698925577	af-south-1
001633400207	ap-east-1
574779866223	ap-northeast-1
709848358524	ap-northeast-2
990339680094	ap-northeast-3
126357580389	ap-south-1
245545462676	ap-southeast-1
563025443158	ap-southeast-2
536280801234	ca-central-1
453000072557	cn-north-1
453252182341	cn-northwest-1
048819808253	eu-central-1
895015795356	eu-north-1
933208885752	eu-south-1
468650794304	eu-west-1
749857270468	eu-west-2
680080141114	eu-west-3
607024016150	me-south-1

ACCOUNT_ID	REGION_NAME
539772159869	sa-east-1
362178532790	us-gov-west-1

要编写您自己的分析容器，请参阅自定义监控 (p. 2201)中描述的容器约定。

解释结果

在运行基准处理作业并获得数据集的统计数据和约束后，可以执行监控作业，以便计算统计数据并列出相对于基准约束遇到的任何违规情况。亚马逊 CloudWatch 默认情况下，指标也会在您的账户中报告。有关在亚马逊中查看监控结果的信息 SageMaker 工作室，请参阅在亚马逊可视化结果 SageMaker 工作室 (p. 2195).

列出执行

计划按指定间隔开始监控作业。以下代码列表：最新的五执行。如果您在创建小时计划后运行此代码，则执行可能为空，并且您可能必须等到跨越小时边界 (UTC) 才能看到执行开始。下面的代码包含等待的逻辑。

```
mon_executions = my_default_monitor.list_executions()
print("We created a hourly schedule above and it will kick off executions ON the hour (plus
 0 - 20 min buffer.\nWe will have to wait till we hit the hour...")

while len(mon_executions) == 0:
    print("Waiting for the 1st execution to happen...")
    time.sleep(60)
    mon_executions = my_default_monitor.list_executions()
```

检查特定执行

在上一步中，您选择了最新的已完成或失败的计划执行。您可以探索探索什么是正确的，什么是错误的。终端状态包括：

- `Completed`— 监控执行已完成，未在违规报告中找到任何问题。
- `CompletedWithViolations`— 执行已完成，但检测到约束违反情况。
- `Failed`— 监控执行失败，可能是因客户端错误（例如，角色问题）或基础设施问题导致的。要确定原因，请参阅 `FailureReason` 和 `ExitMessage`。

```
latest_execution = mon_executions[-1] # latest execution's index is -1, previous is -2 and
 so on..
time.sleep(60)
latest_execution.wait(logs=False)

print("Latest execution status: {}".format(latest_execution.describe()
['ProcessingJobStatus']))
print("Latest execution result: {}".format(latest_execution.describe()['ExitMessage']))

latest_job = latest_execution.describe()
if (latest_job['ProcessingJobStatus'] != 'Completed'):
        print("====STOP==== \n No completed executions to inspect further. Please wait till
 an execution completes or investigate previously reported failures.")
```

```
report_uri=latest_execution.output.destination
print('Report Uri: {}'.format(report_uri))
```

列出生成的报告

列出生成的报告使用以下代码列出生成的报告。

```
from urllib.parse import urlparse
s3uri = urlparse(report_uri)
report_bucket = s3uri.netloc
report_key = s3uri.path.lstrip('/')
print('Report bucket: {}'.format(report_bucket))
print('Report key: {}'.format(report_key))

s3_client = boto3.Session().client('s3')
result = s3_client.list_objects(Bucket=report_bucket, Prefix=report_key)
report_files = [report_file.get("Key") for report_file in result.get('Contents')]
print("Found Report Files:")
print("\n ".join(report_files))
```

违规报告

如果与基准相比存在违规情况，则会在违规情况报告中生成它们。使用以下代码列出违规行为。

```
violations = my_default_monitor.latest_monitoring_constraint_violations()
pd.set_option('display.max_colwidth', -1)
constraints_df = pd.io.json.json_normalize(violations.body_dict["violations"])
constraints_df.head(10)
```

这仅适用于包含表格式数据的数据集。以下架构文件指定计算的统计数据和监视的违规情况。

表格式数据集的输出文件

文件名称	描述
statistics.json	包含所分析数据集中每个特征的列式统计数据。请参阅下一个主题中此文件的架构。 Note 创建此文件仅用于数据质量监控。
constraints_violations.json	包含在当前数据集中找到的相对于 `baseline_statistics` 路径中指定的基准统计数据文件和 `baseline_constaints` 路径中指定的约束文件的所有违规情况。

这些区域有：亚马逊 SageMaker 模型监控器预构建容器 (p. 2192)默认情况下，为每项功能保存一组 Amazon CloudWatch 指标。

容器代码可以发出 CloudWatch 此位置的指标：`/opt/ml/output/metrics/cloudwatch`.

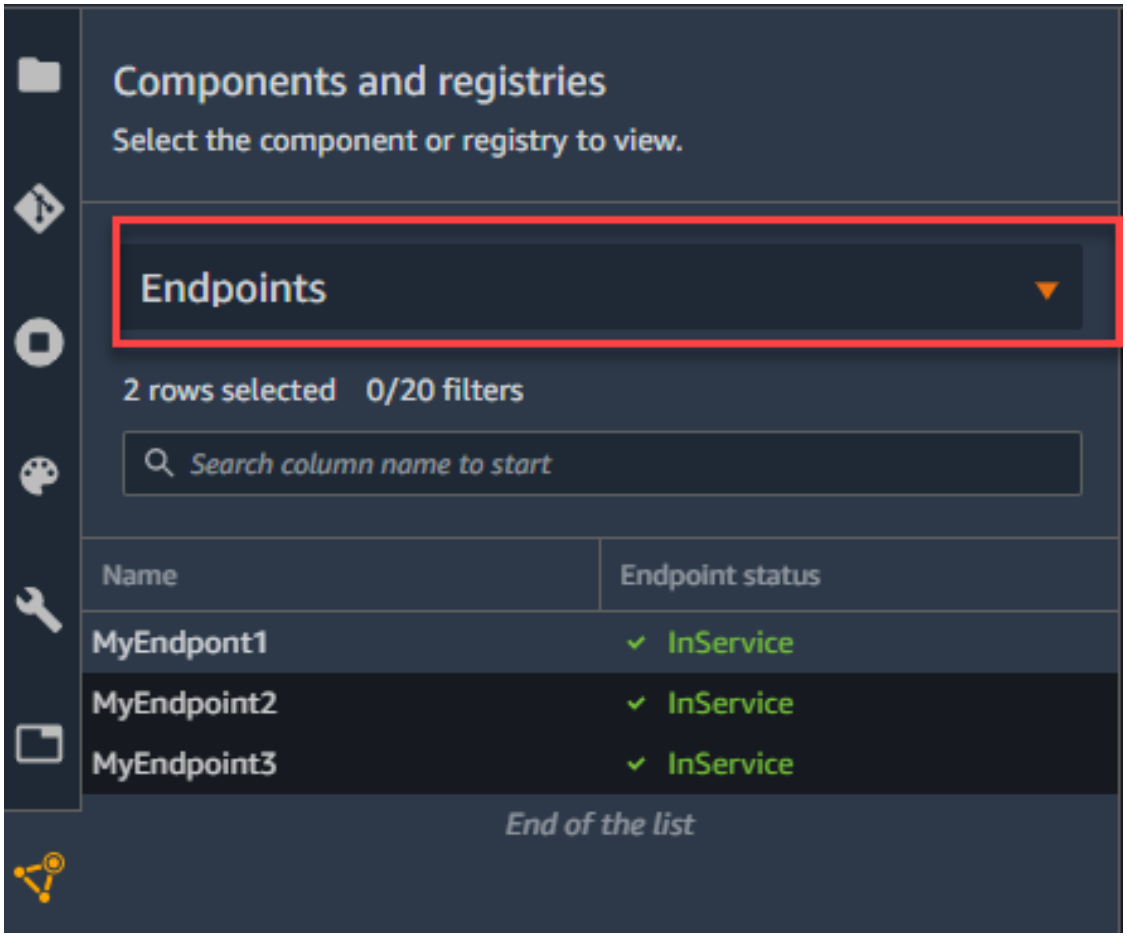
在亚马逊可视化结果 SageMaker 工作室

您还可以在亚马逊中可视化监控结果。 SageMaker Studio。您可以查看任何监控作业运行的详细信息，也可以创建图表来显示监控作业计算的任何指标的基线和捕获的值。

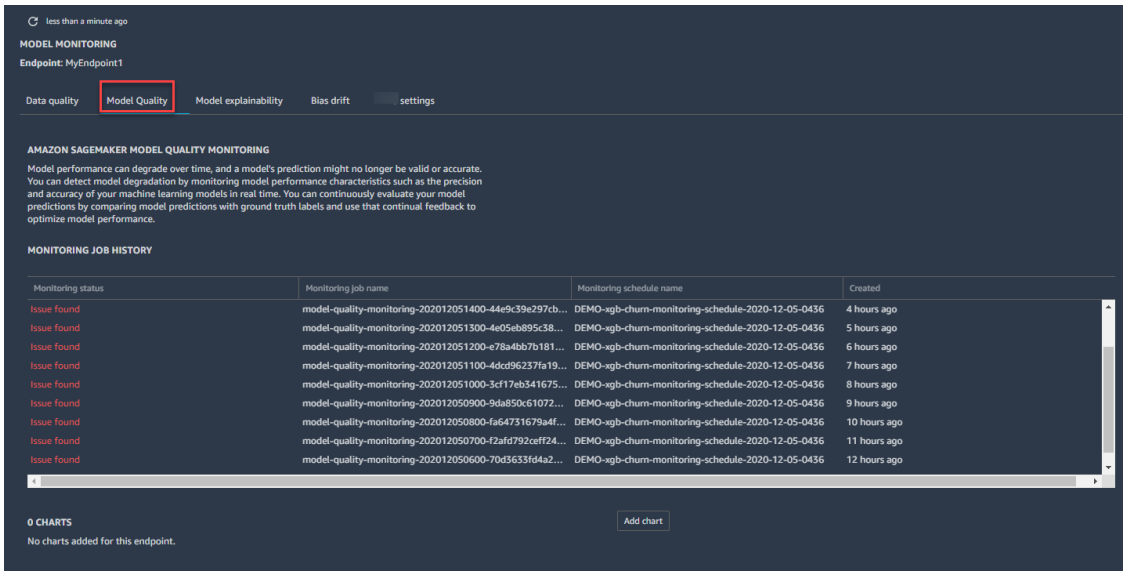
查看监控作业的详细结果

1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。

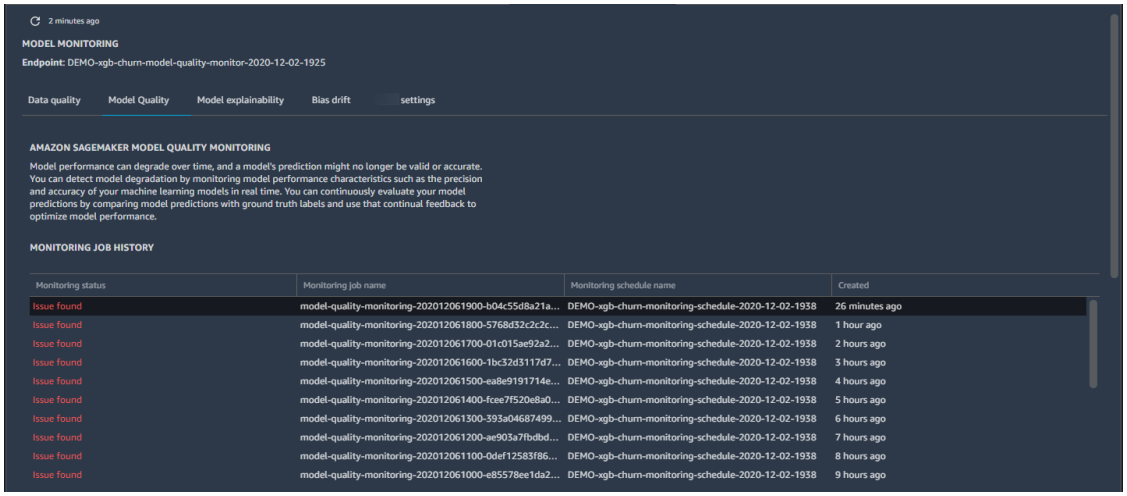
2. 在左侧导航窗格中，选择组件和注册表图标 ( )。
3. 选择终端节点在下拉菜单中。



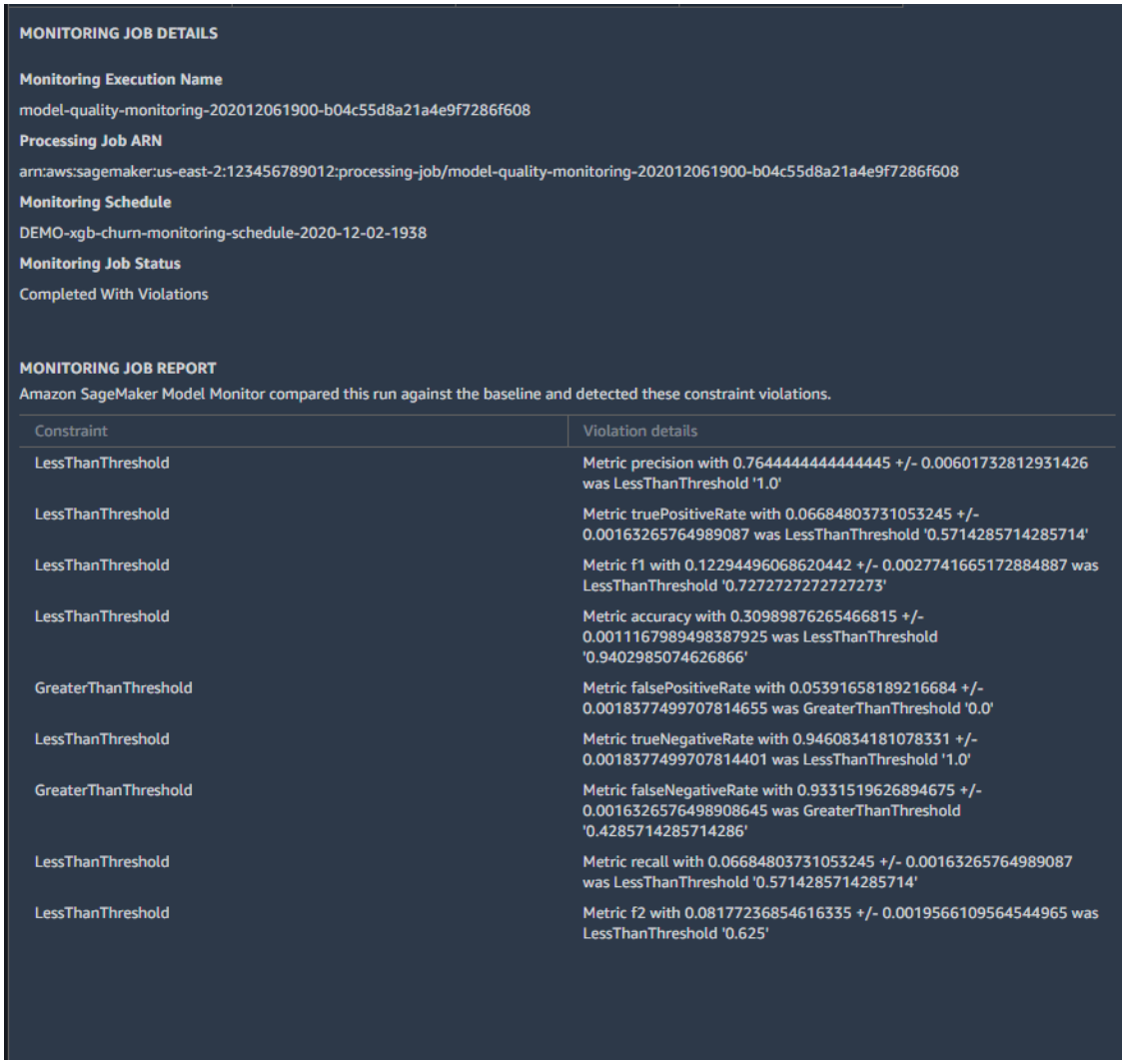
4. 在端点选项卡上，选择要查看其作业详细信息的监控类型。



5. 从监视作业列表中选择要查看其详细信息的监控作业运行的名称。



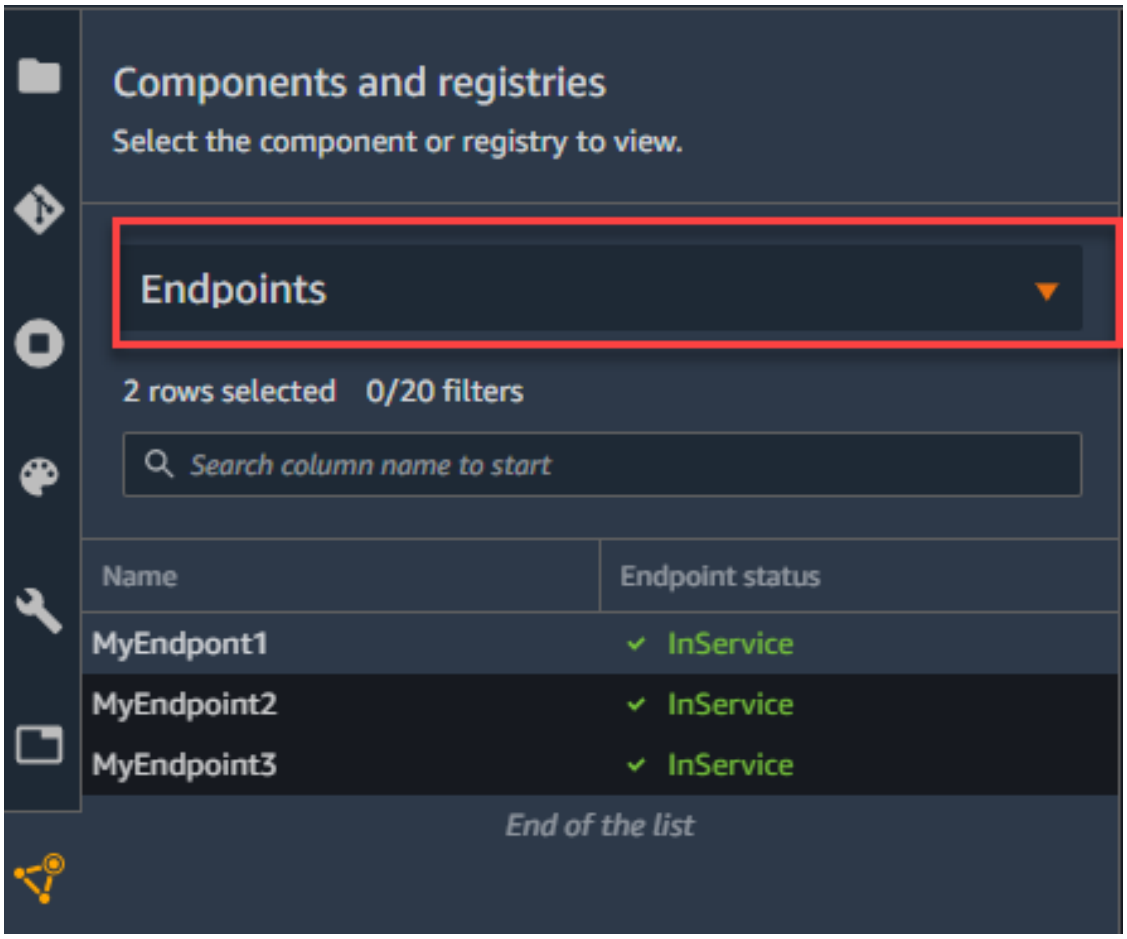
6. 这些区域有：监控任务细节选项卡打开，其中包含监控作业的详细报告。



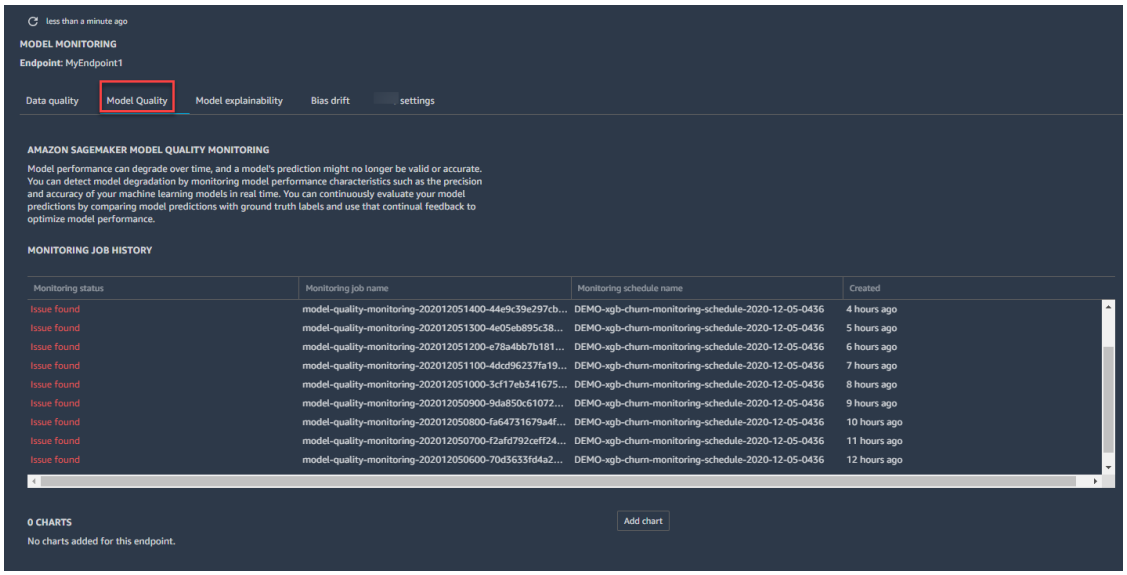
您可以创建一个图表，显示一段时间内的基线和捕获的指标。

在中创建图表 SageMaker Studio 可视化监控结果

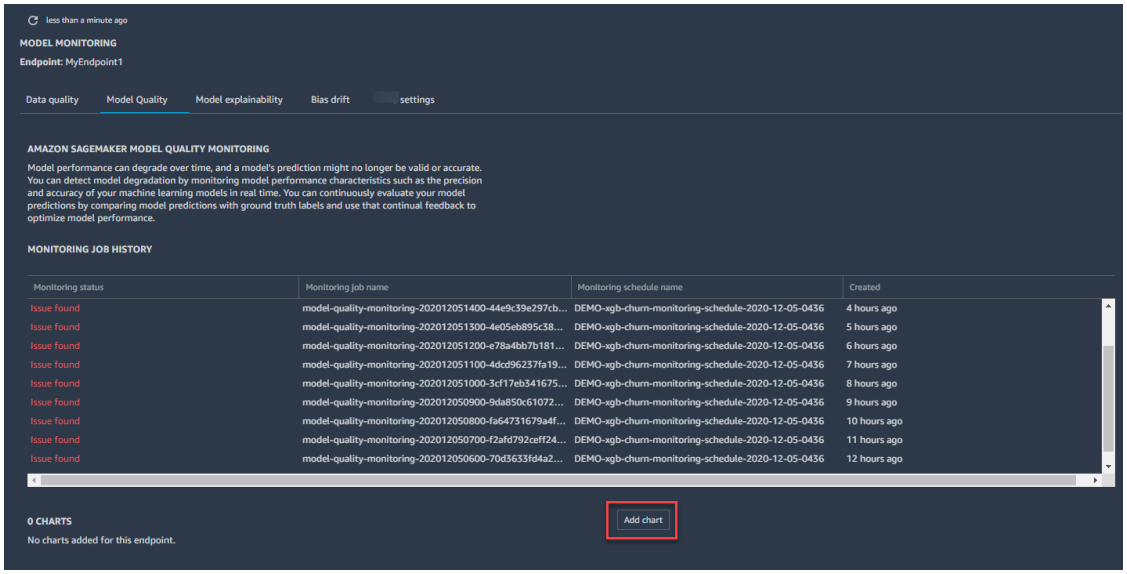
1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
2. 在左侧导航窗格中，选择组件和注册表图标 ( )。
3. 选择终端节点在下拉菜单中。



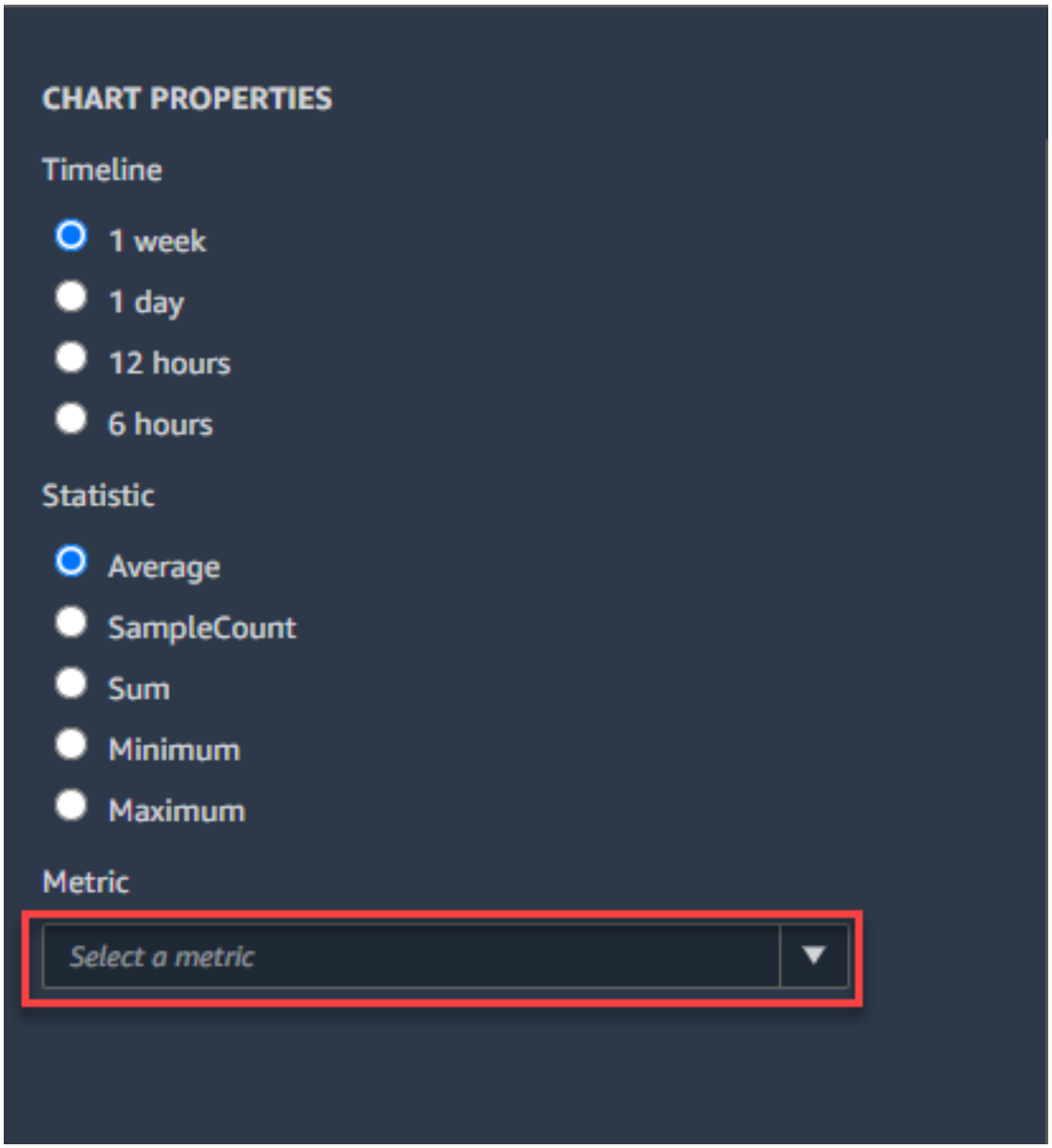
4. 在存储库的端点选项卡中，选择要为其创建图表的监控类型。此示例显示了模型质量监控类型。



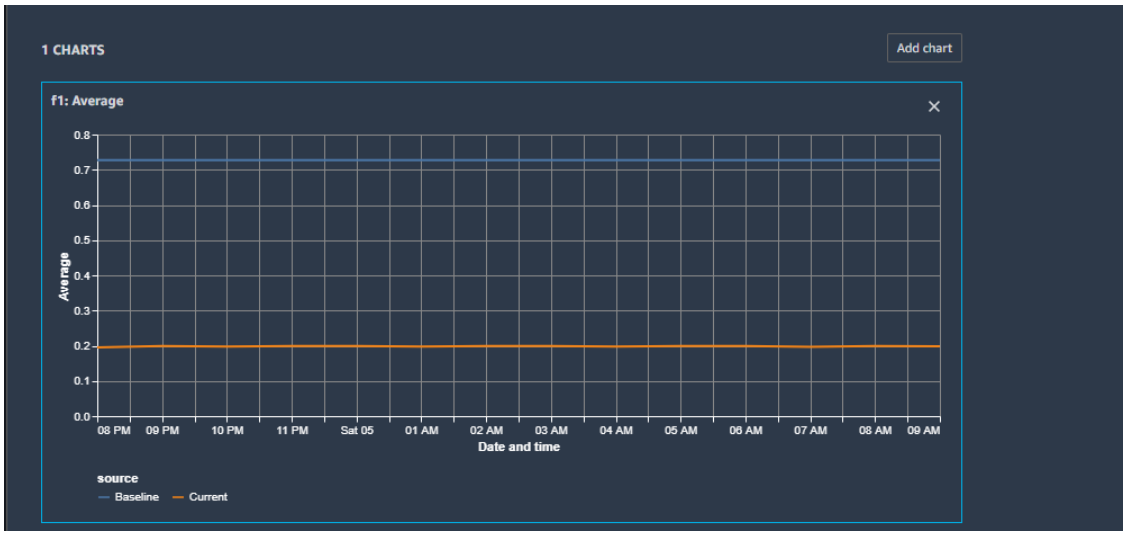
5. 选择添加图表.



6. 在存储库的图表属性选项卡中，选择要绘制图表的时间段、统计数据和量度。此示例显示了一个图表时间表的1 周，Average Statistic的、和F1 指标.



7. 显示您在上一步中选择的基线和当前指标统计数据的图表显示在端点选项卡。



高级主题

以下部分包含更高级的任务，这些任务介绍如何使用预处理和后处理脚本来自定义监控、如何构建您自己的容器以及如何使用。Amazon CloudFormation以创建监控计划。

主题

- 自定义监控 (p. 2201)
- 使用 Amazon CloudFormation 自定义资源创建监控计划 (p. 2210)

自定义监控

除了使用内置监控机制之外，还可以使用预处理和后处理脚本，或通过使用或构建您自己的容器来创建您自己的自定义监控计划和过程。

主题

- 预处理和后处理 (p. 2201)
- 构建您自己的容器 (p. 2204)

预处理和后处理

除了使用内置机制之外，还可以使用预处理和后处理脚本来扩展代码。

主题

- 后处理脚本 (p. 2201)
- 预处理脚本 (p. 2202)

后处理脚本

您可以通过遵循此约定来使用后处理脚本扩展代码：

```
def postprocess_handler():
    print("Hello from post-proc script!")
```

将其指定为 Amazon Simple Storage Service (Amazon S3) 中的路径。`CreateMonitoringSchedule`请求，如下所示：

```
.MonitoringAppSpecification.PostAnalyticsProcessorSourceUri.
```

预处理脚本

Amazon SageMaker 模型监控器容器仅适用于表格式或平面 JSON 结构。我们为转换数据集所需的一些小更改提供了每记录预处理器。例如，如果输出是数组 [1.0, 2.1]，则需要将其转换为平面 JSON，如 {“预测 0”: 1.0, “预测 1”: 2.1"}。示例实施可能与以下内容类似：

```
def preprocess_handler(inference_record):
    input_data = inference_record.endpoint_input.data
    output_data = inference_record.endpoint_output.data.rstrip("\n")
    data = output_data + "," + input_data
    return { str(i).zfill(20) : d for i, d in enumerate(data.split(",")) }
```

将其指定为 Amazon S3 中的路径`CreateMonitoringSchedule`请求：

```
.MonitoringAppSpecification.RecordPreprocessorSourceUri.
```

inference_record 的结构定义如下所示：

```
KEY_EVENT_METADATA = "eventMetadata"
KEY_EVENT_METADATA_EVENT_ID = "eventId"
KEY_EVENT_METADATA_EVENT_TIME = "inferenceTime"
KEY_EVENT_METADATA_CUSTOM_ATTR = "customAttributes"

KEY_EVENTDATA = "captureData"
KEY_EVENTDATA_INPUT = "endpointInput"
KEY_EVENTDATA_OUTPUT = "endpointOutput"
KEY_EVENTDATA_ENCODING = "encoding"
KEY_EVENTDATA_DATA = "data"
KEY_EVENTDATA_OBSERVED_CONTENT_TYPE = "observedContentType"
KEY_EVENTDATA_MODE = "mode"

KEY_EVENT_VERSION = "eventVersion"

"""
{
  "captureData": {
    "endpointInput": {
      "observedContentType": "text/csv",
      "mode": "INPUT",
      "data": "132,25,113.2,96,269.9,107,,0,0,0,0,0,0,1,0,1,0,0,1",
      "encoding": "CSV"
    },
    "endpointOutput": {
      "observedContentType": "text/csv; charset=utf-8",
      "mode": "OUTPUT",
      "data": "0.01076381653547287",
      "encoding": "CSV"
    }
  },
  "eventMetadata": {
    "eventId": "feca1ab1-8025-47e3-8f6a-99e3fdd7b8d9",
    "inferenceTime": "2019-11-20T23:33:12Z"
  },
  "eventVersion": "0"
}
```

```
"""

class EventConfig:
    def __init__(self, endpoint, variant, start_time, end_time):
        self.endpoint = endpoint
        self.variant = variant
        self.start_time = start_time
        self.end_time = end_time


class EventMetadata:
    def __init__(self, event_metadata_dict):
        self.event_id = event_metadata_dict.get(KEY_EVENT_METADATA_EVENT_ID, None)
        self.event_time = event_metadata_dict.get(KEY_EVENT_METADATA_EVENT_TIME, None)
        self.custom_attribute =
 event_metadata_dict.get(KEY_EVENTDATA_OBSERVED_CONTENT_TYPE, None)


class EventData:
    def __init__(self, data_dict):
        self.encoding = data_dict.get(KEY_EVENTDATA_ENCODING, None)
        self.data = data_dict.get(KEY_EVENTDATA_DATA, None)
        self.observedContentType = data_dict.get(KEY_EVENTDATA_OBSERVED_CONTENT_TYPE, None)
        self.mode = data_dict.get(KEY_EVENTDATA_MODE, None)

    def as_dict(self):
        ret = {
            KEY_EVENTDATA_ENCODING: self.encoding,
            KEY_EVENTDATA_DATA: self.data,
            KEY_EVENTDATA_OBSERVED_CONTENT_TYPE: self.observedContentType,
        }
        return ret


class CapturedData:
    def __init__(self, event_dict):
        self.event_metadata = None
        self.endpoint_input = None
        self.endpoint_output = None
        self.event_version = None
        self.event_dict = event_dict
        self._event_dict_postprocessed = False
        if KEY_EVENT_METADATA in event_dict:
            self.event_metadata = EventMetadata(event_dict[KEY_EVENT_METADATA])
        if KEY_EVENTDATA in event_dict:
            if KEY_EVENTDATA_INPUT in event_dict[KEY_EVENTDATA]:
                self.endpoint_input = EventData(event_dict[KEY_EVENTDATA]
[KEY_EVENTDATA_INPUT])
            if KEY_EVENTDATA_OUTPUT in event_dict[KEY_EVENTDATA]:
                self.endpoint_output = EventData(event_dict[KEY_EVENTDATA]
[KEY_EVENTDATA_OUTPUT])
        if KEY_EVENT_VERSION in event_dict:
            self.event_version = event_dict[KEY_EVENT_VERSION]

    def as_dict(self):
        if self._event_dict_postprocessed is True:
            return self.event_dict
        if KEY_EVENTDATA in self.event_dict:
            if KEY_EVENTDATA_INPUT in self.event_dict[KEY_EVENTDATA]:
                self.event_dict[KEY_EVENTDATA][KEY_EVENTDATA_INPUT] =
 self.endpoint_input.as_dict()
            if KEY_EVENTDATA_OUTPUT in self.event_dict[KEY_EVENTDATA]:
                self.event_dict[KEY_EVENTDATA][
                    KEY_EVENTDATA_OUTPUT
                ] = self.endpoint_output.as_dict()
        self._event_dict_postprocessed = True
```

```
        return self.event_dict
```

构建您自己的容器

亚马逊 SageMaker Model Monitor 提供了一个预构建的容器，能够针对表格式数据集分析从终端节点捕获的数据。如果要创建您自己的容器，Model Monitor 提供了可利用的扩展点。

在引擎盖下，当您创建一个`MonitoringSchedule`，模型监视器最终将启动处理作业。因此，容器需要了解构建您自己的处理容器（高级方案）(p. 736)主题中记录的处理作业约定。请注意，模型监控器将按照计划代表您启动处理作业。在调用时，Model Monitor 会为您设置额外的环境变量，以便您的容器具有足够的上下文来处理已计划监控的特定执行的数据。有关容器输入的其他信息，请参阅 容器约定输入 (p. 2204)。

在容器中，通过使用上述环境变量/上下文，您现在可以在自定义代码中分析当前周期的数据集。在此分析完成后，您可以选择发出要上传到 S3 存储桶的报告。预构建的容器所生成的报告记录在中。容器约定输出 (p. 2205). 如果您想在可视化中使用报告 SageMaker Studio，您应该遵循相同的格式。还可以选择发出完全自定义的报告。

你还发出 CloudWatch 按照中的说明操作，从容器中获取指标自备容器的 CloudWatch 指标 (p. 2209).

主题

- 容器约定输入 (p. 2204)
- 容器约定输出 (p. 2205)
- 自备容器的 CloudWatch 指标 (p. 2209)

容器约定输入

Amazon SageMaker Model Monitor 平台根据指定的计划调用您的容器代码。如果您选择编写您自己的容器代码，则以下环境变量可用于您的容器代码。在此上下文中，可以分析当前数据集或评估约束（如果您选择这样做）并发出指标（如果适用）。

```
"Environment": {
 "dataset_format": "{\"sagemakerCaptureJson\": {\"captureIndexNames\": [\"endpointInput\",
\"endpointOutput\"]}}",
 "dataset_source": "/opt/ml/processing/endpointdata",
 "end_time": "2019-12-01T16: 20: 00Z",
 "output_path": "/opt/ml/processing/resultdata",
 "publish_cloudwatch_metrics": "Disabled",
 "sagemaker_endpoint_name": "endpoint-name",
 "sagemaker_monitoring_schedule_name": "schedule-name",
 "start_time": "2019-12-01T15: 20: 00Z"
}
```

参数

参数名称	描述
`dataset_format`	对于从由 `Endpoint` 支持的 `MonitoringSchedule` 开始的作业，这是具有捕获索引 `endpointInput` 和/或 `endpointOutput` 的 `sageMakerCaptureJson`。
`dataset_source`	与由 `start_time` 和 `end_time` 指定的监控周期对应的数据所在的本地路径。在此路径中，数据在 `/{endpoint-name}/{variant-name}/yyyy/mm/dd/hh` 中可用。

参数名称	描述
	有时，我们下载的内容超出了开始时间和结束时间所指定的范围。由容器代码根据需要决定解析数据。
output_path	用于写入输出报告和其他文件的本地路径。您必须在 CreateMonitoringSchedule 请求中将该参数指定为 MonitoringOutputConfig.MonitoringOutput[0].LocalP 它将上传到 MonitoringOutputConfig.MonitoringOutput[0].S3Uri 中指定的 S3Uri 路径。
publish_cloudwatch_metrics	对于由 CreateMonitoringSchedule 启动的作业，此参数设置为 Enabled。容器可以选择写入 Amazon CloudWatch 输出文件[filepath].
sagemaker_endpoint_name	为其启动此计划作业的 Endpoint 的名称。
sagemaker_monitoring_schedule_name	启动了此作业的 MonitoringSchedule 的名称。
sagemaker_endpoint_datacapture_prefix	Endpoint 的 DataCaptureConfig 参数中指定的前缀。如果容器需要直接访问的数据多于下载的数据，则容器可以使用此选项。 SageMaker 在dataset_source路径。
start_time, end_time	此分析的运行时段。例如，对于计划在 05:00 UTC 运行的作业和在 20/02/2020 运行的作业，start_time 为 2020-02-19T06:00:00Z，end_time 为 2020-02-20T05:00:00Z
baseline_constraints:	BaselineConfig.ConstraintResource.S3Uri 中指定的基准约束文件的本地路径。这仅在 CreateMonitoringSchedule 请求中指定了此参数时可用。
baseline_statistics	BaselineConfig.StatisticsResource.S3Uri 中指定的基准统计数据文件的本地路径。这仅在 CreateMonitoringSchedule 请求中指定了此参数时可用。

容器约定输出

容器可以分析 *dataset_source* 路径中可用的数据，并将报告写入 *output_path*. 中的路径。容器代码可以编写任何报告来满足您的需求。

如果您使用以下结构和约定，则某些输出文件会由进行特殊处理。 SageMaker 在可视化和 API 中。这适用于表格式数据集。

表格式数据集的输出文件

文件名称	描述
statistics.json	此文件应具有所分析数据集中每个特征的列式统计数据。此文件的架构在下一部分中提供。

文件名称	描述
constraints.json	此文件应对观察到的特征有约束。此文件的架构在下一部分中提供。
constraints_violations.json	此文件应包含在当前数据集中找到的相对于 baseline_statistics 路径中指定的基准统计数据文件和 baseline_constaints 路径中指定的约束文件的违规情况的列表。

此外，如果publish_cloudwatch_metrics值是"Enabled"容器代码可以发出亚马逊 CloudWatch 此位置的指标：/opt/ml/output/metrics/cloudwatch. 以下部分中描述了这些文件的架构。

主题

- 统计数据的架构（statistics.json 文件）(p. 2206)
- 约束的架构（constraints.json 文件）(p. 2207)

统计数据的架构（statistics.json 文件）

statistics.json 文件中定义的架构指定要为基准和捕获的数据计算的统计参数。它还将存储桶配置为由 KLL（一个带有延迟压缩方案的非常紧凑的分位数草图）使用。

```
{
    "version": 0,
    # dataset level stats
    "dataset": {
        "item_count": number
    },
    # feature level stats
    "features": [
        {
            "name": "feature-name",
            "inferred_type": "Fractional" | "Integral",
            "numerical_statistics": {
                "common": {
                    "num_present": number,
                    "num_missing": number
                },
                "mean": number,
                "sum": number,
                "std_dev": number,
                "min": number,
                "max": number,
                "distribution": {
                    "kll": {
                        "buckets": [
                            {
                                "lower_bound": number,
                                "upper_bound": number,
                                "count": number
                            }
                        ],
                        "sketch": {
                            "parameters": {
                                "c": number,
                                "k": number
                            },
                            "data": [
                                [
                                    num,
```

```
                                    num,
                                    num,
                                    num
                                ],
                                [
                                    num,
                                    num
                                ][
                                    num,
                                    num
                                ]
                            ]
                        }#sketch
                    }#KLL
                }#distribution
            }#num_stats
        },
        {
            "name": "feature-name",
            "inferred_type": "String",
            "string_statistics": {
                "common": {
                    "num_present": number,
                    "num_missing": number
                },
                "distinct_count": number,
                "distribution": {
                    "categorical": {
                         "buckets": [
                                {
                                    "value": "string",
                                    "count": number
                                }
                             ]
                         }
                }
            },
            #provision for custom stats
        }
    ]
}
```

注意

- 指定的指标由以下位置识别。 SageMaker 在以后的可视化更改中。如果需要，容器可以发出更多的指标。
- KLL 草图是公认的草图。自定义容器可以编写自己的表示形式，但无法识别它。 SageMaker 可视化项中。
- 默认情况下，分配将具体化到 10 个存储桶中。无法更改此设置。

约束的架构（constraints.json 文件）

constraints.json 文件用于表示数据集必须满足的约束。亚马逊 SageMaker 模型监控器容器可以使用 constraints.json 文件来评估数据集。利用预构建的容器，您可以自动为基准数据集自动生成 constraints.json 文件。如果您创建了自己的容器，则可以为它提供类似的功能，也可以通过其他方式创建 constraints.json 文件。以下是预构建的容器使用的约束文件的架构。自己携带容器可以采用相同的格式或根据需要对其进行增强。

```
{
```

```
  "version" : 0,

    "features": [
      {
          "name": "string",
          "inferred_type": "Integral" | "Fractional" |
                    | "String" | "Unknown",
          "completeness": number, # denotes observed non-null value percentage
          "num_constraints" : {
            "is_non_negative": boolean,
          },
          "string_constraints" : {
             "domains": [
                  "list of",
                  "observed values",
                  "for small cardinality"
              ],
           },
           "monitoringConfigOverrides" : {

           }#monitoringConfigOverrides
      }#feature
  ]#features

  # options to control monitoring for this feature with monitoring jobs
  # See the following table for notes on what each constraint is doing.
"monitoring_config": {
    "evaluate_constraints": "Enabled",
    "emit_metrics": "Enabled",
    "datatype_check_threshold": 0.1,
    "domain_content_threshold": 0.1,
    "distribution_constraints": {
        "perform_comparison": "Enabled",
        "comparison_threshold": 0.1,
        "comparison_method": "Simple"||"Robust"
    }
}}#schema
```

监控约束

限制	描述
`evaluate_constraints`	在为 `Enabled` 时，评估正在分析的当前数据集是否满足将 constraints.json 文件中指定的约束作为基准。 有效值：`Enabled` 或 `Disabled` 默认值：`Enabled`
`emit_metrics`	何时`Enabled`发射 CloudWatch 文件中包含的数据的指标。 有效值：`Enabled` 或 `Disabled` 默认值：`Enabled`
`datatype_check_threshold`	如果阈值高于指定的 `datatype_check_threshold` 的值，则会导致在违规情况报告中被视为违规情况的失败。如果当前执行中的数据类型与基准数据集中的数据类型不同，则此阈值用于评估是否需要将其标记为违规情况。

限制	描述
	在基准步骤中，生成的约束会为每个列建议推断的数据类型。可以调整 `datatype_check_threshold` 参数，以便调整标记为违规时的阈值。 有效值：浮点值 默认值：0.1
`domain_content_threshold`	如果当前数据集中的字符串字段的未知值多于基准数据集中的未知值，则此阈值可用于指定是否需要将其标记为违规情况。 有效值：浮点值 默认值：0.1
`distribution_constraints`	`perform_comparison` 为 `Enabled` 时，此标志指示代码在基准分布与当前数据集观察到的分布之间执行分布比较。 有效值：`Enabled` 或 `Disabled` 默认值：`Enabled`
	`comparison_threshold` 如果阈值高于为 `comparison_threshold` 设置的值，则会导致在违规情况报告中被视为违规情况的失败。通过获取两个分布的累积分布函数之间的最大绝对差来计算距离。 有效值：浮点值 默认值：0.1
	`comparison_method` 是否计算 `linf_simple` 或 `linf_robust`。`linf_simple` 基于两个分布的累积分布函数之间的最大绝对差。计算 `linf_robust` 基于 `linf_simple`，但仅在样本不足时使用它。`linf_robust` 公式基于 Two-sample Kolmogorov–Smirnov 测试。 有效值：`linf_simple` 或 `linf_robust`.

自备容器的 CloudWatch 指标

如果`publish_cloudwatch_metrics`值是`Enabled`中的`Environment`中的映射`/opt/ml/processing/processingjobconfig.json`文件，容器代码发出亚马逊 CloudWatch 此位置的指标：`/opt/ml/output/metrics/cloudwatch`.

此文件的架构严格基于 CloudWatch`PutMetrics`API。此处未指定命名空间。其默认值为 `/aws/sagemaker/Endpoint/data-metrics`。不过，您可以指定维度。建议您至少添加 `Endpoint` 和 `MonitoringSchedule` 维度。

```
{
    "MetricName": "", # Required
    "Timestamp": "2019-11-26T03:00:00Z", # Required
    "Dimensions" : [{"Name":"Endpoint","Value":"endpoint_0"},
{"Name":"MonitoringSchedule","Value":"schedule_0"}]
    "Value": Float,
    # Either the Value or the StatisticValues field can be populated and not both.
    "StatisticValues": {
        "SampleCount": Float,
        "Sum": Float,
        "Minimum": Float,
        "Maximum": Float
    },
    "Unit": "Count", # Optional
}
```

使用 Amazon CloudFormation 自定义资源创建监控计划

要使用 Amazon CloudFormation 创建监控计划，请使用 Amazon CloudFormation 自定义资源。自定义资源位于 Python 中。要部署它，请参阅 Python Lambda 部署。

自定义资源

首先，将自定义资源添加到您的 Amazon CloudFormation 模板。这指向了Amazon Lambda您在下一个步骤中创建的函数。

利用此资源，您可以自定义监控计划的参数。您可以通过修改Amazon CloudFormation资源和以下示例资源中的 Lambda 函数。

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "MonitoringSchedule": {
            "Type": "Custom::MonitoringSchedule",
            "Version": "1.0",
            "Properties": {
                "ServiceToken": "arn:aws:lambda:us-west-2:111111111111:function:lambda-
name",
                "ScheduleName": "YourScheduleName",
                "EndpointName": "YourEndpointName",
                "BaselineConstraintsUri": "s3://your-baseline-constraints/
constraints.json",
                "BaselineStatisticsUri": "s3://your-baseline-stats/statistics.json",
                "PostAnalyticsProcessorSourceUri": "s3://your-post-processor/
postprocessor.py",
                "RecordPreprocessorSourceUri": "s3://your-preprocessor/preprocessor.py",
                "InputLocalPath": "/opt/ml/processing/endpointdata",
                "OutputLocalPath": "/opt/ml/processing/localpath",
                "OutputS3URI": "s3://your-output-uri",
                "ImageURI": "111111111111.dkr.ecr.us-west-2.amazonaws.com/your-image",
                "ScheduleExpression": "cron(0 * ? * * *)",
                "PassRoleArn": "arn:aws:iam::111111111111:role/AmazonSageMaker-
ExecutionRole"
            }
        }
    }
}
```

Lambda 自定义资源代码

该Amazon CloudFormation自定义资源使用自定义资源辅助 Amazon库，可使用使用 pip 安装该库`pip install crhelper`.

在创建和删除堆栈的过程中，Amazon CloudFormation 将调用此 Lambda 函数。此 Lambda 函数负责创建和删除监控计划，并使用上一部分中描述的自定义资源中定义的参数。

```
import boto3
import botocore
import logging

from crhelper import CfnResource
from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)
sm = boto3.client('sagemaker')

# cfnhelper makes it easier to implement a CloudFormation custom resource
helper = CfnResource()

# CFN Handlers

def handler(event, context):
    helper(event, context)


@helper.create
def create_handler(event, context):
    """
    Called when CloudFormation custom resource sends the create event
    """
    create_monitoring_schedule(event)


@helper.delete
def delete_handler(event, context):
    """
    Called when CloudFormation custom resource sends the delete event
    """
    schedule_name = get_schedule_name(event)
    delete_monitoring_schedule(schedule_name)


@helper.poll_create
def poll_create(event, context):
    """
    Return true if the resource has been created and false otherwise so
    CloudFormation polls again.
    """
    schedule_name = get_schedule_name(event)
    logger.info('Polling for creation of schedule: %s', schedule_name)
    return is_schedule_ready(schedule_name)

@helper.update
def noop():
    """
    Not currently implemented but crhelper will throw an error if it isn't added
    """
    pass

# Helper Functions

def get_schedule_name(event):
    return event['ResourceProperties']['ScheduleName']

def create_monitoring_schedule(event):
    schedule_name = get_schedule_name(event)
    monitoring_schedule_config = create_monitoring_schedule_config(event)
```

```
    logger.info('Creating monitoring schedule with name: %s', schedule_name)

    sm.create_monitoring_schedule(
        MonitoringScheduleName=schedule_name,
        MonitoringScheduleConfig=monitoring_schedule_config)

def is_schedule_ready(schedule_name):
    is_ready = False

    schedule = sm.describe_monitoring_schedule(MonitoringScheduleName=schedule_name)
    status = schedule['MonitoringScheduleStatus']

    if status == 'Scheduled':
        logger.info('Monitoring schedule (%s) is ready', schedule_name)
        is_ready = True
    elif status == 'Pending':
        logger.info('Monitoring schedule (%s) still creating, waiting and polling
 again...', schedule_name)
    else:
        raise Exception('Monitoring schedule ({}) has unexpected status:
 {}'.format(schedule_name, status))

    return is_ready

def create_monitoring_schedule_config(event):
    props = event['ResourceProperties']

    return {
        "ScheduleConfig": {
            "ScheduleExpression": props["ScheduleExpression"],
        },
        "MonitoringJobDefinition": {
            "BaselineConfig": {
                "ConstraintsResource": {
                    "S3Uri": props['BaselineConstraintsUri'],
                },
                "StatisticsResource": {
                    "S3Uri": props['BaselineStatisticsUri'],
                }
            },
            "MonitoringInputs": [
                {
                    "EndpointInput": {
                        "EndpointName": props["EndpointName"],
                        "LocalPath": props["InputLocalPath"],
                    }
                }
            ],
            "MonitoringOutputConfig": {
                "MonitoringOutputs": [
                    {
                        "S3Output": {
                            "S3Uri": props["OutputS3URI"],
                            "LocalPath": props["OutputLocalPath"],
                        }
                    }
                ],
            },
            "MonitoringResources": {
                "ClusterConfig": {
                    "InstanceCount": 1,
                    "InstanceType": "ml.t3.medium",
                    "VolumeSizeInGB": 50,
                }
            },
```

```
            "MonitoringAppSpecification": {
                "ImageUri": props["ImageURI"],
                "RecordPreprocessorSourceUri": props['PostAnalyticsProcessorSourceUri'],
                "PostAnalyticsProcessorSourceUri":
 props['PostAnalyticsProcessorSourceUri'],
            },
            "StoppingCondition": {
                "MaxRuntimeInSeconds": 300
            },
            "RoleArn": props["PassRoleArn"],
        }
    }


def delete_monitoring_schedule(schedule_name):
    logger.info('Deleting schedule: %s', schedule_name)
    try:
        sm.delete_monitoring_schedule(MonitoringScheduleName=schedule_name)
    except ClientError as e:
        if e.response['Error']['Code'] == 'ResourceNotFound':
            logger.info('Resource not found, nothing to delete')
        else:
            logger.error('Unexpected error while trying to delete monitoring schedule')
            raise e
```

调用实时端节点

使用 SageMaker 托管服务部署模型后，可以通过发送测试数据在该端点上测试模型。您可以使用 Amazon SageMaker Studio 测试终端节点，Amazon CLI，或者Amazon开发工具包。

使用 Amazon SageMaker Studio 测试您的终端节点

将模型部署到终端节点后（请参阅创建终端节点并部署模型 (p. 2103)) 您可以使用 Amazon SageMaker Studio 检查该终端节点。

> Note
>
> 注意：SageMaker 仅支持使用 Amazon SageMaker Studio 进行实时终端节点的端点测试。

1. 启动 Amazon SageMaker Studio。
2. SelectSageMaker 组件和注册表左侧栏上的图标。
3. 从下拉列表中，选择终端节点.
4. 按名称搜索终端节点，然后双击终端节点的名称。中列出的终端节点名称SageMaker 资源在您部署模型时定义了面板。您可以通过多种方式部署模型：
   - SageMaker Python SDK 模型类`sagemaker.model.Model.deploy`.
   - 这些区域有：Amazon SDK for Python (Boto3)SageMaker 服务客户端 APICreateEndpoint.
   - SageMaker 控制台。Select推理在左侧面板中，然后选择端节点配置. 在中提供终端节点名称终端节点名称字段中返回的子位置类型。
5. （可选）您可以选择提供自定义 URL 以将请求发送到。在配置终端节点 URL 和标头字段提供模型托管位置的 URL。如果您使用的是 SageMaker 终端节点，请将此字段留空。您还可以选择添加多个键值标头以将其他信息传递给推理请求。
6. Studio 工作区中将填充一个新的选项卡。Select测试推理选项卡。
7. 通过提供 JSON 格式的示例数据，向终端节点发送请求。使用 JSON 编辑器向终端节点提交请求。
8. Select发送请求.
9. 当你发送请求时推理输出卡将显示在控制台的右侧。

卡片顶部将显示发送到终端节点的请求类型（目前只接受 JSON）。卡片中有四个主要领域：Status、Execution Length、Request Time, 和Result Time.

- 状态：显示请求状态。有三种类型的状态：
  - Complete-如果请求成功，状态将显示Complete执行时间将通过跟踪请求时间来计算。
  - Failed-如果由于任何原因请求失败。失败响应将显示在失败原因手风琴。
  - Pending-推理请求待处理期间，将出现旋转的圆形图标。
- 执行长度：调用需要多长时间（结束时间减去开始时间），以毫秒为单位。
- 请求时间：自请求发送以来已经过去了多少分钟。
- 结果时间：自结果返回以来已经过去了多少分钟。

使用测试终端节点Amazon软件开发工具包

将模型部署到终端节点后（请参阅创建终端节点并部署模型 (p.)) 你可以使用InvokeEndpointAPI 使用其中一个Amazon开发工具包。如果成功的调用，SageMaker 将返回 HTTP 200 响应。您可以为返回的响应字典对象编制索引，以了解有关响应标题的更多信息。有关更多信息，请参阅 InvokeEndpoint。

以下内容演示了如何使用InvokeEndpoint使用Amazon SDK for Python (Boto3). 提供终端节点的名称。这是你指定的名字EndpointName当你使用创建终端节点时CreateEndpoint.

在 Body 字段中提供输入数据，以便 SageMaker 传递给模型。数据的格式必须与用于培训的格式相同。

```
import boto3

# Create a low-level client representing Amazon SageMaker Runtime
sagemaker_runtime = boto3.client("sagemaker-runtime", region_name=<aws_region>)

# The name of the endpoint. The name must be unique within an Amazon Region in your Amazon
 account.
endpoint_name='<endpoint-name>'

# After you deploy a model into production using SageMaker hosting
# services, your client applications use this API to get inferences
# from the model hosted at the specified endpoint.
response = sagemaker_runtime.invoke_endpoint(
                            EndpointName=endpoint_name,
                            Body=bytes('{"features": ["This is great!"]}', 'utf-8') #
 Replace with your own data.
                            )

# Optional - Print the response body and decode it so it is human read-able.
print(response['Body'].read().decode('utf-8'))
```

一旦你有了响应对象（在上述示例中）它就存储在一个名为response) 你可以索引它来检查 HTTP 状态，即已部署模型的名称（InvokedProductionVariant) 和其他字段。

正在进行的代码片段将打印HTTPStatusCode存储在response变量：

```
print(response["HTTPStatusCode"])
```

有关更多信息，请参阅 。InvokeEndpoint请参阅 SageMaker API 参考指南。

使用测试终端节点Amazon CLI

以下内容演示了如何使用InvokeEndpoint使用Amazon CLI. 提供终端节点的名称。这是你指定的名字EndpointName当你使用创建终端节点时CreateEndpoint.

在 Body 字段中提供输入数据，以便 SageMaker 传递给模型。数据的格式必须与用于培训的格式相同。示例代码模板显示了如何向终端节点发送二进制数据。

```
aws sagemaker-runtime invoke-endpoint \
    --endpoint-name endpoint_name \
    --body fileb://$file_name \
    output_file.txt
```

有关何时使用的更多信息`file://`结束了`fileb://`将文件的内容传递给的参数时Amazon CLI，请参阅本地文件参数的最佳实践.

请参阅`invoke-endpoint`中的Amazon CLI命令参考有关可以传递的其他参数的更多信息的指南。

如果上面的调用显示如下的片段，则表示命令已成功执行。否则，请检查输入有效负载格式是否正确。

```
{
    "ContentType": "<content_type>; charset=utf-8",
    "InvokedProductionVariant": "<Variant>"
}
```

通过检查文件输出文件查看调用的输出 (`output_file.txt`在这个例子中）。

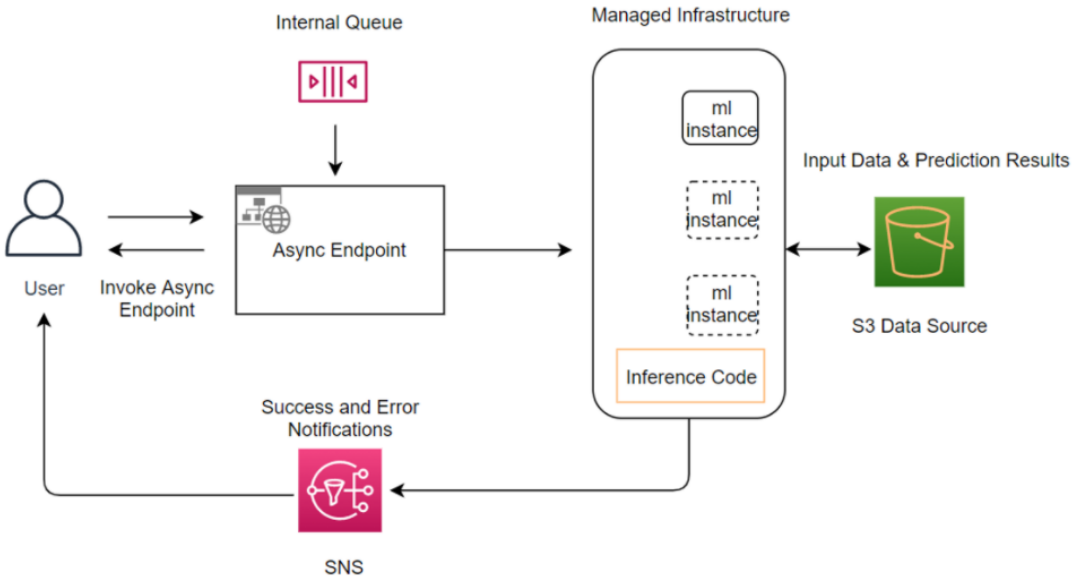
```
more output_file.txt
```

异步推理

Amazon SageMaker 异步推理是 SageMaker 中的一项新功能，可以对传入的请求进行排队并异步处理它们。此选项非常适合具有大负载大小（最多 1GB）、处理时间长（最长 15 分钟）和接近实时延迟要求的请求。异步推理使您能够在没有处理请求时自动将实例计数自动缩放到零来节省成本，因此您只需在终端节点处理请求时付费。

工作方式

创建异步推理端点类似于创建实时推理端点。您可以使用现有的 SageMaker 模型，而且只需指定`AsyncInferenceConfig`在创建终端节点配置时，请使用`EndpointConfig`字段中的`CreateEndpointConfig`API。下图演示了异步推理的架构和工作流程。



要调用终端节点，您需要将请求有效负载放在 Amazon S3 中，并提供指向此有效负载的指针作为`InvokeEndpointAsync`请求. 调用后，SageMaker 将请求排队以进行处理，并返回标识符和输出位置作为响应。处理后，SageMaker 会将结果放置在 Amazon S3 位置。您可以选择使用 Amazon SNS 接收成功或错误通知。有关如何设置异步通知的更多信息，请参阅检查预测结果 (p. 2226).

> **Note**
>
> 异步推理配置的存在 (`AsyncInferenceConfig`) 端点配置中的对象意味着终端节点只能接收异步调用。

怎样入门？

如果您是首次接触 Amazon SageMaker 异步推断的用户，我们建议您执行以下操作：

- Read创建、调用和更新异步终端节点 (p. 2216)以获取有关如何创建、调用、更新和删除异步终端节点的信息。
- 探索异步推理示例笔记本中的aws/亚马逊sagemaker-示例GitHub 存储库。

请注意，如果您的终端节点使用此中列出的任何功能Exclusions (p. 2251)页面中，您无法使用异步推理。

创建、调用和更新异步终端节点

本指南演示了创建异步终端节点必须满足的先决条件，以及如何创建、调用和删除异步终端节点。您可以使用创建、更新、删除和调用异步终端节点Amazon开发工具包和Amazon SageMaker Python 开发工具包.

主题

- 先决条件 (p. 2217)
- 创建异步推理端点 (p. 2219)
- 调用异步终端节点 (p. 2222)
- 更新异步终端节点 (p. 2222)
- 删除异步终端节点 (p. 2223)

先决条件

要使用异步终端节点，请首先确保满足这些先决条件。

1. 为 Amazon SageMaker 创建 IAM 角色。

   异步推理需要访问 Amazon S3 存储桶 URI。为了实现这一点，请创建一个 IAM 角色，该角色可以运行 SageMaker 并有权访问 Amazon S3 和 Amazon SNS。使用此角色，SageMaker 可以在您的账户下运行并访问您的 Amazon S3 存储桶和 Amazon SNS 主题。

   您可以使用 IAM 控制台创建 IAM 角色，Amazon SDK for Python (Boto3)，或者Amazon CLI. 以下是如何创建 IAM 角色并使用 IAM 控制台附加必要的策略的示例。

   a. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
   b. 在 IAM 控制台的导航窗格中，选择 Roles，然后选择 Create role。
   c. 对于选择受信任实体的类型，选择 Amazon 服务。
   d. 选择您希望允许其承担此角色的服务。在本例中，选择SageMaker. 接下来，选择 Next (下一步)：Permissions (下一步：权限)。

      - 这会自动创建 IAM 策略，该策略授予对 Amazon S3、Amazon ECR 和 CloudWatch Logs 等相关服务的访问权限。

   e. 选择 Next:。标签。
   f. （可选）通过以键值对的形式附加标签来向角色添加元数据。有关将在 IAM 中使用标签的更多信息，请参阅 Tagging IAM resources（标记 IAM 资源）。
   g. 选择 Next:。审核。
   h. 输入Role name (角色名称).
   i. 如果可能，键入角色名称或角色名称后缀。角色名称在您的 Amazon 账户内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 `PRODROLE` 和 `prodrole` 的角色。由于其他 Amazon 资源可能引用该角色，角色创建完毕后无法编辑角色名称。
   j. (可选) 对于 Role description，键入新角色的描述。
   k. 检查角色，然后选择 Create role。

      请注意 SageMaker 角色 ARN。要使用控制台查找角色 ARN，请执行以下操作：

      i. 转到 IAM 控制台：https://console.aws.amazon.com/iam/
      ii. Select角色.
      iii. 通过在搜索字段中键入角色的名称，搜索刚才创建的角色。
      iv. 选择角色。
      v. 角色 ARN 位于摘要页.

2. 将 Amazon SageMaker、Amazon S3 和 Amazon SNS 权限添加到您的 IAM 角色。

   创建角色后，向您的 IAM 角色授予 SageMaker、Amazon S3 以及可选的 Amazon SNS 权限。

   选择角色在 IAM 控制台中。通过在搜索字段中返回的子位置类型。

   a. 选择你的角色。
   b. 下一步，选择附加策略.
   c. Amazon SageMaker 异步推理需要权才能执行以下操作：`"sagemaker:CreateModel"`、`"sagemaker:CreateEndpointConfig"`、`"sagemaker:CreateEndp`和`"sagemaker:InvokeEndpointAsync"`.

这些操作包含在AmazonSageMakerFullAccess政策。将此策略添加到您的 IAM 角色。搜索AmazonSageMakerFullAccess中的搜索字段中返回的子位置类型。Select AmazonSageMakerFullAccess.

d. 选择 Attach policy（附上策略）。

e. 下一步，选择附加策略以添加 Amazon S3 权限。

f. Select创建策略.

g. 选择 JSON 选项卡。

h. 添加以下策略语句：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::bucket_name/*"
        }
    ]
}
```

i. 选择 Next:。标签。

j. 输入策略名称.

k. 选择 Create policy（创建策略）。

l. 重复您完成的相同步骤以添加 Amazon S3 权限以添加 Amazon SNS 权限。对于政策声明，请附上以下内容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:sns:<region>:<Account_ID>:<SNS_Topic>"
        }
    ]
}
```

3. 将推理数据（例如机器学习模型、示例数据）上传到Amazon S3.

4. 选择预构建的 Docker 推理映像或创建自己的推理 Docker 映像。

   SageMaker 为其内置算法提供了容器，并为一些最常见的机器学习框架提供了预构建的 Docker 映像，如 Apache MXNet、TensorFlow、PyTorch 和 Chainer。有关可用 SageMaker 映像的完整列表，请参阅可用的 Deep Learning Containers 映像.

   如果现有的 SageMaker 容器都不能满足您的需求，并且您没有自己的现有容器，则可能需要创建一个新的 Docker 容器。请参阅使用您自己的推理代码 (p. 2435)了解有关如何创建 Docker 映像的信息。

5. 创建 Amazon SNS 主题（可选）

   创建 Amazon Simple Notification Service (Amazon SNS) 主题，用于发送有关已完成处理的请求的通知。Amazon SNS 是面向消息的应用程序的通知服务，多个订阅者通过各种传输协议（包

括 HTTP、Amazon SQS 和电子邮件）请求和接收时间关键型消息的“推送”通知。您可以在创建`EndpointConfig`指定时的对象`AsyncInferenceConfig`使用`EndpointConfig`API。

按照以下步骤创建并订阅 Amazon SNS 主题。

a. 使用 Amazon SNS 控制台创建主题。有关说明，请参阅创建 Amazon SNS 主题中的Amazon Simple Notification Service 开发人员指南.
b. 订阅至主题。有关说明，请参阅订阅 Amazon SNS 主题中的Amazon Simple Notification Service开发人员指南.
c. 当您收到要求您确认订阅该主题的电子邮件时，请确认订阅。
d. 记下主题的 Amazon 资源名称 (ARN)。您创建的 Amazon SNS 主题是您的Amazon账户，它有唯一的 ARN。ARN 采用以下格式：

```
arn:aws:sns:aws-region:account-id:topic-name
```

有关 Amazon SNS 的更多信息，请参阅Amazon SNS 开发人员指南.

创建异步推理端点

创建异步终端节点的方式与使用 SageMaker 托管服务创建终端节点的方式相同：

- 使用在 SageMaker 中创建模型`CreateModel`.
- 使用创建端点配置`CreateEndpointConfig`.
- 使用创建 HTTPS 终端节点`CreateEndpoint`.

要创建终端节点，首先需要使用`CreateModel`，您可以在其中指向模型工件和 Docker 注册表路径（图像）。然后，您可以使用`CreateEndpointConfig`您可以在其中指定一个或多个使用`CreateModel`要部署的 API 以及您希望 SageMaker 预配置的资源。使用创建终端节点`CreateEndpoint`使用在请求中指定的终端节点配置。您可以使用`UpdateEndpoint`API。通过以下方式发送和接收来自端点托管的模型的推理请求`InvokeEndpointAsync`. 您可以使用`DeleteEndpoint`API。

有关可用 SageMaker 映像的完整列表，请参阅可用的 Deep Learning Containers 映像. 请参阅使用您自己的推理代码 (p. 2435)了解有关如何创建 Docker 映像的信息。

创建模型

以下示例显示了如何使用Amazon SDK for Python (Boto3). 前几行定义：

- `sagemaker_client`：一个低级 SageMaker 客户端对象，可以轻松向Amazon服务。
- `sagemaker_role`：具有 SageMaker IAM 角色的字符串变量：Amazon Resource Name (ARN)。
- `aws_region`: 具有你的名字的字符串变量Amazon区域。

```
import boto3

# Specify your AWS Region
aws_region='<aws_region>'

# Create a low-level SageMaker service client.
sagemaker_client = boto3.client('sagemaker', region_name=aws_region)

# Role to give SageMaker permission to access Amazon services.
sagemaker_role= "arn:aws:iam::<account>:role/*"
```

下一步，指定存储在 Amazon S3 中的预先训练的模型的位置。在本例中，我们使用的是预先训练的 XGBoost 模型`demo-xgboost-model.tar.gz`. 完整的 Amazon S3 URI 存储在字符串变量中`model_url`：

```
#Create a variable w/ the model S3 URI
s3_bucket = '<your-bucket-name>' # Provide the name of your S3 bucket
bucket_prefix='saved_models'
model_s3_key = f"{bucket_prefix}/demo-xgboost-model.tar.gz"

#Specify S3 bucket w/ model
model_url = f"s3://{s3_bucket}/{model_s3_key}"
```

指定主容器。对于主容器，您可以指定包含推理代码的 Docker 映像、构件（来自先前的培训）以及推理代码在您部署预测模型时使用的自定义环境贴图。

在此示例中，我们指定了 xgBoost 内置算法容器镜像：

```
from sagemaker import image_uris

# Specify an AWS container image.
container = image_uris.retrieve(region=aws_region, framework='xgboost', version='0.90-1')
```

使用在 Amazon SageMaker 中创建模型`CreateModel`. 指定以下内容：

- `ModelName`：模型的名称（在本例中，它存储为字符串变量，名为`model_name`)。
- `ExecutionRoleArn`：IAM 角色的 Amazon 资源名称 (ARN)，Amazon SageMaker 可代入此角色以访问模型项目和 Docker 映像以在 ML 计算实例上进行部署或批量转换作业。
- `PrimaryContainer`：主要 Docker 映像的位置，该映像包含推理代码、关联构件和自定义环境贴图（在部署模型以进行预测时，推理代码将使用此贴图）。

```
model_name = '<The_name_of_the_model>'

#Create model
create_model_response = sagemaker_client.create_model(
    ModelName = model_name,
    ExecutionRoleArn = sagemaker_role,
    PrimaryContainer = {
        'Image': container,
        'ModelDataUrl': model_url,
    })
```

请参阅`CreateModel`有关 API 参数的完整列表，请参阅 SageMaker API 参考指南中的描述。

创建终端节点配置

拥有模型后，使用创建端点配置`CreateEndpointConfig`. Amazon SageMaker 托管服务使用此配置来部署模型。在配置中，您可以标识使用`CreateModel`，以部署您希望 Amazon SageMaker 预配置的资源。指定`AsyncInferenceConfig`对象并提供输出 Amazon S3 位置`OutputConfig`. 您可以选择指定Amazon SNS发送有关预测结果的通知的主题。有关 Amazon SNS 主题的更多信息，请参阅配置 Amazon SNS.

以下示例说明了如何使用创建终端节点配置。Amazon SDK for Python (Boto3)：

```
import datetime
from time import gmtime, strftime

# Create an endpoint config name. Here we create one based on the date
# so it we can search endpoints based on creation time.
endpoint_config_name = f"XGBoostEndpointConfig-{strftime('%Y-%m-%d-%H-%M-%S', gmtime())}"
```

```
# The name of the model that you want to host. This is the name that you specified when
 creating the model.
model_name='<The_name_of_your_model>'

create_endpoint_config_response = sagemaker_client.create_endpoint_config(
    EndpointConfigName=endpoint_config_name, # You will specify this name in a
 CreateEndpoint request.
    # List of ProductionVariant objects, one for each model that you want to host at this
 endpoint.
    ProductionVariants=[
        {
            "VariantName": "variant1", # The name of the production variant.
            "ModelName": model_name,
            "InstanceType": "ml.m5.xlarge", # Specify the compute instance type.
            "InitialInstanceCount": 1 # Number of instances to launch initially.
        }
    ],
    AsyncInferenceConfig={
        "OutputConfig": {
            # Location to upload response outputs when no location is provided in the
 request.
            "S3OutputPath": f"s3://{s3_bucket}/{bucket_prefix}/output"
            # (Optional) specify Amazon SNS topics
            "NotificationConfig": {
                "SuccessTopic": "arn:aws:sns:aws-region:account-id:topic-name",
                "ErrorTopic": "arn:aws:sns:aws-region:account-id:topic-name",
            }
        },
        "ClientConfig": {
            # (Optional) Specify the max number of inflight invocations per instance
            # If no value is provided, Amazon SageMaker will choose an optimal value for
 you
            "MaxConcurrentInvocationsPerInstance": 4
        }
    }
)

print(f"Created EndpointConfig: {create_endpoint_config_response['EndpointConfigArn']}")
```

在上述示例中，您可以指定以下键`OutputConfig`(对于)`AsyncInferenceConfig`字段：

- `S3OutputPath`：请求中未提供位置时上传响应输出的位置。
- `NotificationConfig`: (可选) 推理请求成功时向您发布通知的 SNS 主题 (`SuccessTopic`) 或者如果失败了 (`ErrorTopic`)。

您也可以指定以下可选参数`ClientConfig`中的`AsyncInferenceConfig`字段：

- `MaxConcurrentInvocationsPerInstance`：（可选）SageMaker 客户端向模型容器发送的并发请求的最大数量。

创建终端节点

拥有模型和终端节点配置后，请使用`CreateEndpoint`用于创建终端节点的 API。终端节点名称在Amazon您的区域Amazonaccount.

以下操作将使用在请求中指定的终端节点配置创建终端节点。Amazon SageMaker 使用终端节点来配置资源和部署模型。

```
# The name of the endpoint.The name must be unique within an AWS Region in your AWS
 account.
```

```
endpoint_name = '<endpoint-name>'

# The name of the endpoint configuration associated with this endpoint.
endpoint_config_name='<endpoint-config-name>'

create_endpoint_response = sagemaker_client.create_endpoint(
                                            EndpointName=endpoint_name,
                                            EndpointConfigName=endpoint_config_name)
```

当你打电话给`CreateEndpoint`API，Amazon SageMaker 异步推理会发送测试通知，以检查您是否配置了 Amazon SNS 主题。这可以让 SageMaker 检查您是否拥有所需的权限。可以简单地忽略该通知。测试通知具有以下形式：

```
{
    "eventVersion":"1.0",
    "eventSource":"aws:sagemaker",
    "eventName":"TestNotification"
}
```

调用异步终端节点

从异步终端节点托管的模型中获取推论`InvokeEndpointAsync`.

> **Note**
>
> 如果您尚未这样做，请将推理数据（例如机器学习模型、示例数据）上传到 Amazon S3。

在中指定推理数据的位置`InputLocation`字段和终端节点的名称`EndpointName`：

```
# Create a low-level client representing Amazon SageMaker Runtime
sagemaker_runtime = boto3.client("sagemaker-runtime", region_name=<aws_region>)

# Specify the location of the input. Here, a single SVM sample
input_location = "s3://bucket-name/test_point_0.libsvm"

# The name of the endpoint. The name must be unique within an AWS Region in your AWS
 account.
endpoint_name='<endpoint-name>'

# After you deploy a model into production using SageMaker hosting
# services, your client applications use this API to get inferences
# from the model hosted at the specified endpoint.
response = sagemaker_runtime.invoke_endpoint_async(
                            EndpointName=endpoint_name,
                            InputLocation=input_location)
```

您会收到一个 JSON 字符串的响应，其中包含请求 ID 和 Amazon S3 存储桶的名称，该存储桶将在处理 API 调用后响应该调用。

更新异步终端节点

使用更新异步终端节点点`UpdateEndpoint`API。当您更新终端节点时，SageMaker 首先预配置并切换到您指定的新端点配置，然后再删除之前的终端节点配置中预配的资源。请勿删除`EndpointConfig`使用处于活动状态或在`UpdateEndpoint`要么`CreateEndpoint`正在终端上执行操作。

```
# The name of the endpoint. The name must be unique within an Amazon Region in your Amazon
 account.
endpoint_name='<endpoint-name>'
```

```
# The name of the endpoint configuration associated with this endpoint.
endpoint_config_name='<endpoint-config-name>'

sagemaker_client.update_endpoint(
                                  EndpointConfigName=endpoint_config_name,
                                  EndpointName=endpoint_name
                                  )
```

当 Amazon SageMaker 收到请求时，它将终端节点的状态设置为。正在更新. 更新异步终端节点后，它将状态设置为InService. 要检查终端节点的状态，请使用DescribeEndpointAPI。有关更新终端节点时可指定的完整参数列表，请参阅UpdateEndpointAPI。

删除异步终端节点

删除异步终端节点的方式与删除 SageMaker 托管终端节点的方式类似DeleteEndpointAPI。指定要删除的异步终端节点的名称。当您删除终端节点时，SageMaker 释放在创建终端节点时部署的所有资源。删除模型不会删除模型项目、推理代码或在创建模型时指定的 IAM 角色。

使用删除 SageMaker 模型DeleteModelAPI 或使用 SageMaker 控制台。

Boto3

```
import boto3

# Create a low-level SageMaker service client.
sagemaker_client = boto3.client('sagemaker', region_name=<aws_region>)
sagemaker_client.delete_endpoint(EndpointName='<endpoint-name>')
```

SageMaker console

1. 导航到 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 展开推理下拉列表。
3. 选择终端节点。
4. 在中搜索终端节点搜索端点搜索栏。
5. 选择终端节点。
6. 选择 Delete（删除）。

除了删除异步终端节点之外，您可能还想清除用于创建终端节点的其他资源，例如 Amazon ECR 存储库（如果您创建了自定义推理映像）、SageMaker 模型以及异步终端节点配置本身。

监控异步端节点

您可以使用 Amazon CloudWatch 监控 SageMaker，Amazon CloudWatch 会收集原始数据，并将数据处理为便于读取的近乎实时 使用 Amazon CloudWatch，您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。有关 Amazon CloudWatch 的更多信息，请参阅什么是 Amazon CloudWatch？

使用 CloudWatch 进行监控

以下指标是异步终端节点指标的详尽列表。如果终端节点启用异步推断，则不会发布下面未列出的任何指标。此类指标包括（但不限于）：

- 开销延迟
- 调用

- InvocationsPerInstance

常见端节点指标

这些指标与今天为实时终端节点发布的指标相同。有关 Amazon CloudWatch 中其他指标的更多信息，请参阅使用 Amazon CloudWatch 监控 SageMaker.

指标名称	描述	Unit /统计信息
`Invocation4XXErrors`	模型在其中返回 4xx HTTP 响应代码的 请求的数量。对于每个 4xx 响应，发送 1；否则，发送 0。	单位：无 有效统计数据：Average、Sum
`Invocation5XXErrors`	模型在其中返回 5xx HTTP 响应代码的 InvokeEndpoint 请求的数量。对于每个 5xx 响应，发送 1；否则，发送 0。	单位：无 有效统计数据：Average、Sum
`ModelLatency`	从 SageMaker 观察到的模型做出响应所用的时间间隔。此时间间隔包括发送请求以及从模型容器提取响应的本地通信时间，以及在容器中完成推理所用的时间。	单位：微秒 有效统计数据：Average、Sum、Min、Min、Max、Sample Count

异步推理终端节点指标

这些指标是针对启用异步推理的终端节点发布的。以下指标与`EndpointName`维度：

指标名称	描述	Unit /统计信息
`ApproximateBacklogSize`	终端节点队列中当前正在处理或尚未处理的项目数。	单位：计数 有效统计数据：平均、最大、最小
`ApproximateBacklogSizePerInstance`	队列中的项目数除以终端节点后面的实例数。此指标主要用于为启用异步的终端节点设置应用程序自动扩展。	单位：计数 有效统计数据：平均、最大、最小
`ApproximateAgeOfOldestRequest`	队列中最早请求的年龄。	单位：秒 有效统计数据：平均、最大、最小

以下指标与`EndpointName`和`VariantName`维度：

指标名称	描述	Unit /统计信息
`RequestDownloadFailures`	当由于从 Amazon S3 下载请求时出现问题而导致推理失败时。	单位：计数 有效统计数据：总计

指标名称	描述	Unit /统计信息
ResponseUploadFailures	当由于将响应上传到 Amazon S3 出现问题而导致推理失败时。	单位：计数 有效统计数据：总计
NotificationFailures	当出现问题时，发布通知。	单位：计数 有效统计数据：总计
RequestDownloadLatency	下载请求负载的总时间。	单位：微秒 有效统计数据： Average、Sum、Min、Min、Max、Sample Count
ResponseUploadLatency	上传响应有效负载的总时间。	单位：微秒 有效统计数据： Average、Sum、Min、Min、Max、Sample Count
ExpiredRequests	队列中因达到指定请求 TTL 而失败的请求数。	单位：计数 有效统计数据：总计
InvocationFailures	如果调用因任何原因失败。	单位：计数 有效统计数据：总计
InvocationsProcesssed	终端节点处理的异步调用数量。	单位：计数 有效统计数据：总计
TimeInBacklog	处理请求之前排队的总时间。这不包括实际处理时间（即下载时间、上传时间、模型延迟）。	单位：微秒 有效统计数据： Average、Sum、Min、Min、Max、Sample Count
TotalProcessingTime	SageMaker 收到推理请求到请求完成处理的时间。这包括积压时间以及上传和发送回复通知的时间（如果有）。	单位：毫秒 有效统计数据： Average、Sum、Min、Min、Max、Sample Count

Amazon SageMaker 异步推理还包括主机级指标。有关主机级指标的信息，请参阅SageMaker 作业和端点指标.

日志

除了模型容器日志在账户中发布到 Amazon CloudWatch 中，您还可以获得一个用于跟踪和调试推理请求的新平台日志。

新日志发布在端点日志组下：

```
/aws/sagemaker/Endpoints/[EndpointName]
```

日志流名称包括：

```
[production-variant-name]/[instance-id]/data-log.
```

日志行包含请求的推理 ID，以便可以轻松地将错误映射到特定请求。

检查预测结果

您可以使用多种方法检查异步终端节点的预测结果。一些选项包括：

1. Amazon SNS 主题。
2. 检查 Amazon S3 存储桶中的输出。

Amazon SNS 主题

Amazon SNS 是面向消息的应用程序的通知服务，多个订阅者通过各种传输协议（包括 HTTP、Amazon SQS 和电子邮件）请求和接收时间关键型消息的“推送”通知。当您使用创建终端节点时，Amazon SageMaker 异步推理会发布通知`CreateEndpointConfig`并指定 Amazon SNS 主题。

> **Note**
>
> 要接收 Amazon SNS 通知，您的 IAM 角色必须具有`sns:Publish`权限。请参阅先决条件 (p. 2217)有关使用异步推理必须满足的要求的信息。

要使用 Amazon SNS 检查异步终端节点的预测结果，您首先需要创建主题、订阅该主题、确认订阅主题，然后记下该主题的亚马逊资源名称 (ARN)。有关如何创建、订阅和查找 Amazon SNS 主题的 Amazon ARN 的详细信息，请参阅配置 Amazon SNS.

在中提供 Amazon SNS 主题 ARN`AsyncInferenceConfig`使用创建终端配置时使用的终端配置`CreateEndpointConfig`. 您可以同时指定 Amazon SNS`ErrorTopic`和`SuccessTopic`.

```
import boto3

sagemaker_client = boto3.client('sagemaker', region_name=<aws_region>)

sagemaker_client.create_endpoint_config(
    EndpointConfigName=<endpoint_config_name>, # You specify this name in a CreateEndpoint
 request.
    # List of ProductionVariant objects, one for each model that you want to host at this
 endpoint.
    ProductionVariants=[
        {
            "VariantName": "variant1", # The name of the production variant.
            "ModelName": "model_name",
            "InstanceType": "ml.m5.xlarge", # Specify the compute instance type.
            "InitialInstanceCount": 1 # Number of instances to launch initially.
        }
    ],
    AsyncInferenceConfig={
        "OutputConfig": {
            # Location to upload response outputs when no location is provided in the
 request.
            "S3OutputPath": "s3://<bucket>/<output_directory>"
            "NotificationConfig": {
                "SuccessTopic": "arn:aws:sns:aws-region:account-id:topic-name",
                "ErrorTopic": "arn:aws:sns:aws-region:account-id:topic-name",
            }
        }
    }
)
```

创建终端节点并调用终端节点后，您会收到来自 Amazon SNS 主题的通知。例如，如果您订阅了接收来自主题的电子邮件通知，则每次调用终端节点时都会收到一封电子邮件通知。以下示例显示成功调用电子邮件通知的 JSON 内容。

```
{
   "awsRegion":"us-east-1",
   "eventTime":"2022-01-25T22:46:00.608Z",
   "receivedTime":"2022-01-25T22:46:00.455Z",
   "invocationStatus":"Completed",
   "requestParameters":{
      "contentType":"text/csv",
      "endpointName":"<example-endpoint>",
      "inputLocation":"s3://<bucket>/<input-directory>/input-data.csv"
   },
   "responseParameters":{
      "contentType":"text/csv; charset=utf-8",
      "outputLocation":"s3://<bucket>/<output_directory>/prediction.out"
   },
   "inferenceId":"11111111-2222-3333-4444-555555555555",
   "eventVersion":"1.0",
   "eventSource":"aws:sagemaker",
   "eventName":"InferenceResult"
}
```

检查你的 S3 存储桶

当你调用终端节点时`InvokeEndpointAsync`，它返回响应对象。您可以使用响应对象获取存储输出的 Amazon S3 URI。使用输出位置，您可以使用 SageMaker Python SDK SageMaker 会话类以编程方式检查输出。

下面存储的输出字典`InvokeEndpointAsync`作为名为响应的变量。然后使用响应变量，您可以获取 Amazon S3 输出 URI 并将其存储为名为的字符串变量`output_location`.

```
import uuid
import boto3

sagemaker_runtime = boto3.client("sagemaker-runtime", region_name=<aws_region>)

# Specify the S3 URI of the input. Here, a single SVM sample
input_location = "s3://bucket-name/test_point_0.libsvm"

response = sagemaker_runtime.invoke_endpoint_async(
    EndpointName='<endpoint-name>',
    InputLocation=input_location,
    InferenceId=str(uuid.uuid4()),
    ContentType="text/libsvm" #Specify the content type of your data
)

output_location = response['OutputLocation']
print(f"OutputLocation: {output_location}")
```

有关支持的内容类型的信息，请参阅用于推理的常见数据格式 (p. 1583).

然后，您可以使用 Amazon S3 输出位置SageMaker Python SDK SageMaker 会话类读取 Amazon S3 文件。以下代码示例演示如何创建函数 (`get_ouput`) 它反复尝试从 Amazon S3 输出位置读取文件：

```
import sagemaker
import urllib, time
from botocore.exceptions import ClientError

sagemaker_session = sagemaker.session.Session()
```

```
def get_output(output_location):
    output_url = urllib.parse.urlparse(output_location)
    bucket = output_url.netloc
    key = output_url.path[1:]
    while True:
        try:
            return sagemaker_session.read_s3_file(
                                        bucket=output_url.netloc,
                                        key_prefix=output_url.path[1:])
        except ClientError as e:
            if e.response['Error']['Code'] == 'NoSuchKey':
                print("waiting for output...")
                time.sleep(2)
                continue
            raise

output = get_output(output_location)
print(f"Output: {output}")
```

自动缩放异步端点

Amazon SageMaker 支持自动扩展（自动扩展）异步终端节点。自动扩展 动态调整为模型预配置的实例数，以响应工作负荷的变化。与 Amazon SageMaker 支持的其他托管模型不同，借助异步推理，您还可以将异步终端节点实例缩小到零。在终端节点扩展后，在没有实例时收到的请求将排队等待处理。

要自动扩展异步终端节点，您至少必须：

- 注册已部署的模型（生产变体）。
- 定义扩展策略。
- 应用自动扩展策略。

在使用自动扩展之前，必须已创建 SageMaker 模型部署。部署的模型称为生产变体。请参阅将模型部署到 SageMaker 托管服务有关部署模型终端节点的更多信息。要为扩展策略指定指标和目标值，请配置目标跟踪扩展策略。有关如何定义扩展策略的信息，请参阅删除扩展策略. 注册模型并定义扩展策略后，将扩展策略应用于已注册的模型。有关如何应用扩展策略的信息，请参阅应用扩展策略.

有关与自动扩展结合使用的其他先决条件和组件的详细信息，先决条件SageMaker 自动缩放文档中的部分。

删除扩展策略

要为扩展策略指定指标和目标值，请配置目标跟踪扩展策略。在文本文件中将扩展策略定义为 JSON 块。在调用 Amazon CLI 或 Application Auto Scaling API 时，您可以使用该文本文件。有关策略配置语法的更多信息，请参阅`TargetTrackingScalingPolicyConfiguration`在应用程序 Auto Scaling API 参考中。

对于异步终端节点，SageMaker 强烈建议您为变体的目标跟踪扩展创建策略配置。在此配置示例中，我们使用自定义指标，`CustomizedMetricSpecification`，调用`ApproximateBacklogSizePerInstance`.

```
TargetTrackingScalingPolicyConfiguration={
        'TargetValue': 5.0, # The target value for the metric. Here the metric is:
 SageMakerVariantInvocationsPerInstance
        'CustomizedMetricSpecification': {
            'MetricName': 'ApproximateBacklogSizePerInstance',
            'Namespace': 'AWS/SageMaker',
            'Dimensions': [
                {'Name': 'EndpointName', 'Value': <endpoint_name> }
            ],
            'Statistic': 'Average',
        }
```

```
    }
```

定义扩展到 0 的扩展策略

下面介绍了如何使用应用程序自动扩缩来定义和注册终端节点变体Amazon SDK for Python (Boto3). 在使用 Boto3 定义表示应用程序自动扩展的低级客户端对象之后，我们使用`RegisterScalableTarget`注册生产变体的方法。我们设置`MinCapacity`之所以为 0，因为异步推理使您能够在没有处理请求时自动缩放到 0。

```
# Common class representing application autoscaling for SageMaker
client = boto3.client('application-autoscaling')

# This is the format in which application autoscaling references the endpoint
resource_id='endpoint/' + <endpoint_name> + '/variant/' + <'variant1'>

# Define and register your endpoint variant
response = client.register_scalable_target(
    ServiceNamespace='sagemaker',
    ResourceId=resource_id,
    ScalableDimension='sagemaker:variant:DesiredInstanceCount', # The number of EC2
 instances for your Amazon SageMaker model endpoint variant.
    MinCapacity=0,
    MaxCapacity=5
)
```

有关应用程序自动扩缩 API 的详细描述，请参阅应用程序扩展 Boto3文档中）。

使用批量转换

当您需要执行以下操作时，请使用批量转换：

- 预处理数据集以从数据集中删除可能干扰训练或推理的噪声或偏差。
- 从大型数据集获取推理。
- 当您不需要持续性终端节点时运行推理。
- 将输入记录与推理相关联，以帮助解释结果。

要在执行推理前筛选输入数据，或要将输入记录与有关这些记录的推理相关联，请参阅关联预测结果使用输入记录 (p. 2231)。例如，您可以筛选输入数据，为创建和解释有关输出数据的报告提供上下文。

主题

- 使用批量转换获取大型数据集的推理 (p. 2229)
- 加快批量转换作业 (p. 2231)
- 使用批量转换测试生产变体 (p. 2231)
- 批量转换错误 (p. 2231)
- 批量转换示例笔记本 (p. 2231)
- 关联预测结果使用输入记录 (p. 2231)
- Batch 存储转换 (p. 2236)

使用批量转换获取大型数据集的推理

Batch 转换自动管理大型数据集的处理。在指定参数的限制范围内. 例如，假设您有一个数据集文件，`input1.csv`、存储在 S3 存储桶中. 输入文件的内容可能如下示例所示：

```
Record1-Attribute1, Record1-Attribute2, Record1-Attribute3, ..., Record1-AttributeM
Record2-Attribute1, Record2-Attribute2, Record2-Attribute3, ..., Record2-AttributeM
Record3-Attribute1, Record3-Attribute2, Record3-Attribute3, ..., Record3-AttributeM
...
RecordN-Attribute1, RecordN-Attribute2, RecordN-Attribute3, ..., RecordN-AttributeM
```

当批量转换作业启动时，SageMaker 会初始化计算实例，并在它们之间分配推理或预处理工作负载。Batch 转换按照键对输入中的 Amazon S3 对象进行分区，并将 Amazon S3 对象映射到实例。当您有多个文件时，一个实例可能处理 `input1.csv`，而另一个实例可能处理名为 `input2.csv` 的文件。如果您有一个输入文件但初始化多个计算实例，则只有一个实例处理输入文件，其余实例处于空闲状态。

您还可以将输入文件拆分成小批量。例如，您可以通过仅包含其中两个记录来从 `input1.csv` 创建一个小批量。

```
Record3-Attribute1, Record3-Attribute2, Record3-Attribute3, ..., Record3-AttributeM
Record4-Attribute1, Record4-Attribute2, Record4-Attribute3, ..., Record4-AttributeM
```

> **Note**
>
> SageMaker 单独处理每一个输入文件。它不会组合来自不同输入文件的小批次来符合`MaxPayloadInMB` 限制。

要在创建批量转换作业时将输入文件拆分成小批量，请将`SplitType`参数值为`Line`. 如果`SplitType`设置为`None`如果输入文件无法拆分成小批量，SageMaker 会在单个请求中使用整个输入文件。请注意，Batch 转换不支持包含嵌入换行符的 CSV 格式输入。您可以通过使用`BatchStrategy`和`MaxPayloadInMB`参数。`MaxPayloadInMB`不得大于 100 MB。如果你指定可选`MaxConcurrentTransforms`参数，然后为的值`(MaxConcurrentTransforms * MaxPayloadInMB)`也不得超过 100 MB。

如果批量转换作业成功处理输入文件中的所有记录，它会创建一个文件扩展名为 `.out` 的同名输出文件。对于 `input1.csv` 和 `input2.csv` 等多个输入文件，输出文件将分别名为 `input1.csv.out` 和 `input2.csv.out`。批量转换作业将输出文件存储在 Amazon S3 的指定位置，如。`s3://awsexamplebucket/output/`.

输出文件中的预测按与输入文件中对应的记录相同的顺序列出。输出文件 `input1.csv.out` 的内容（基于早前显示的输入文件）如下所示。

```
Inference1-Attribute1, Inference1-Attribute2, Inference1-Attribute3, ..., Inference1-
AttributeM
Inference2-Attribute1, Inference2-Attribute2, Inference2-Attribute3, ..., Inference2-
AttributeM
Inference3-Attribute1, Inference3-Attribute2, Inference3-Attribute3, ..., Inference3-
AttributeM
...
InferenceN-Attribute1, InferenceN-Attribute2, InferenceN-Attribute3, ..., InferenceN-
AttributeM
```

要将多个输出文件的结果合并为单一输出文件，设置`AssembleWith`参数为`Line`.

当输入数据非常大且使用 HTTP 分块编码传输时，要将数据流式传输到算法，请将设置为。`MaxPayloadInMB`到`0`. Amazon SageMaker 内置算法不支持此功能。

有关使用 API 创建批量转换作业的信息，请参阅 `CreateTransformJob` API。有关批量转换输入和输出对象之间关联的更多信息，请参阅。`OutputDataConfig`. 有关如何使用批量转换的示例，请参阅（可选）使用 Batch 变换进行预测 (p. 69)。

加快批量转换作业

如果您使用的是CreateTransformJobAPI，您可以使用最佳的参数值来减少完成批量转换作业所需的时间。例如MaxPayloadInMB、MaxConcurrentTransforms，或者BatchStrategy. 的理想价值MaxConcurrentTransforms等于批处理转换作业中的计算工作人员的数量。如果您使用的是 SageMaker 控制台，则可以在其他配置的 部分Batch 转换作业配置页. SageMaker 会自动找到内置算法的最佳参数设置。对于自定义算法，请通过执行参数终端节点。

使用批量转换测试生产变体

要测试不同的模型或各种超参数设置，请为每个新的模型变体创建单独的转换作业并使用验证数据集。对于每一个转换作业，为输出文件指定一个唯一的模型名称和位于 Amazon S3 中的位置。要分析结果，请使用推理管道日志和指标 (p. 2138)。

批量转换错误

SageMaker 使用 Amazon S3分段上传 API使用将批量转换作业的结果上传到 Amazon S3。如果出现错误，则系统会从 Amazon S3 中删除上传的结果。在某些情况下（如发生网络中断时），未完成的分段上传可能会保留在 Amazon S3 中。为避免产生存储费用，我们建议您将 S3 存储桶策略添加到 S3 存储桶生命周期规则中。此策略会删除可能存储在 S3 存储桶中的未完成分段上传。有关更多信息，请参阅对象生命周期管理。

如果批量转换作业因数据集存在问题而无法处理输入文件，SageMaker 会将该作业标记为。failed. 如果输入文件包含错误记录，则转换作业不会为该输入文件创建输出文件，因为这样做的话，将无法与输入文件中的转换后数据保持相同的顺序。当数据集具有多个输入文件时，即使转换作业无法处理其中一个输入文件，它也会继续处理这些文件。处理后的文件仍会生成可用的结果。

超过了MaxPayloadInMB限制会导致错误。如果大型数据集无法拆分，可能会发生这种情况，SplitType参数设置为none，或者数据集中的单个记录超过限制。

如果您使用的是自己的算法，则当算法在输入文件中找到错误记录时，您可以使用占位符文本，如 ERROR。例如，如果数据集中的最后一条记录是错误的，算法会在输出文件中放入占位符文本来替代该记录。

批量转换示例笔记本

对于将批量转换与主成分分析 (PCA) 模型结合使用以减少用户项查看矩阵中的数据，然后将应用程序的基于密度的空间聚类以及噪声 (DBSCAN) 算法应用于聚类电影的示例笔记本 ，请参阅使用 PCA 和 DBSCAN电影聚类进行批量转换。有关创建和访问可用于在 SageMaker 中运行示例的 Jupyter 笔记本实例的说明，请参阅。使用 Amazon SageMaker 笔记本实例 (p. 147). 创建并打开笔记本实例后，请选择SageMaker 示例选项卡以查看所有 SageMaker 示例的列表。主题建模示例笔记本的主题建模使用NTM 算法位于高级功能部分。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

关联预测结果使用输入记录

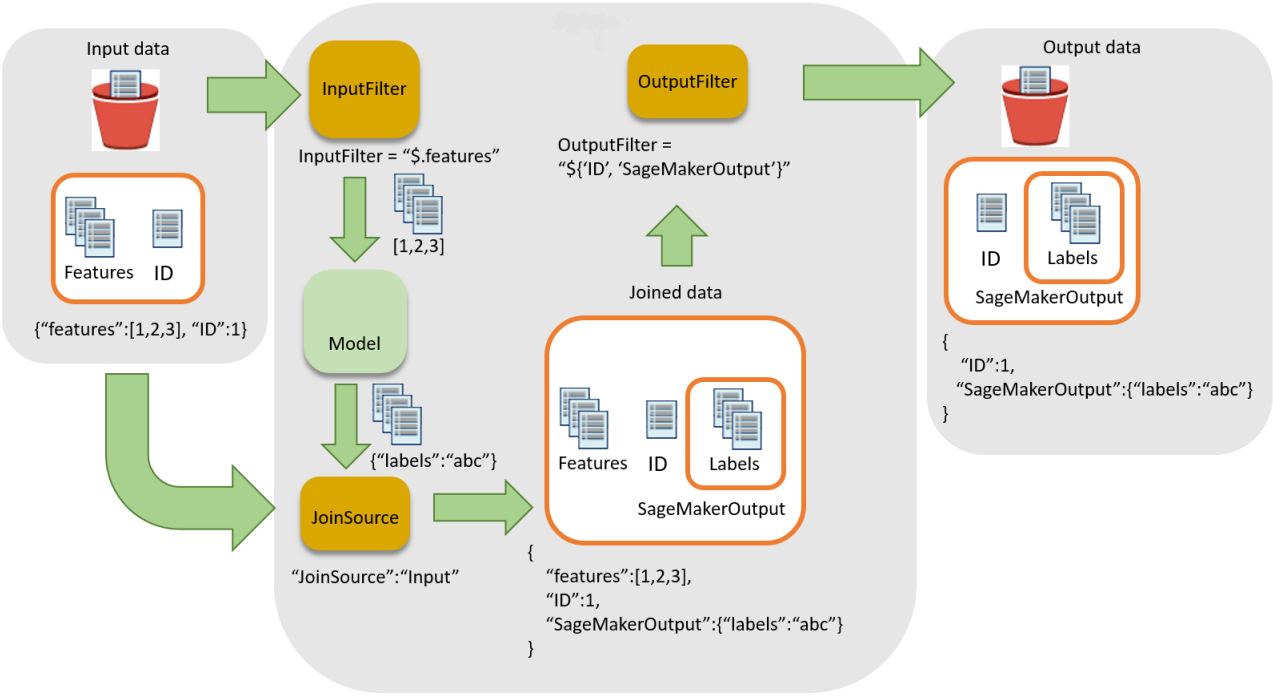
当针对大型数据集进行预测时，可以排除进行预测时不需要的属性。进行预测后，您可以将某些排除的属性和这些预测或报告中的其他输入数据关联起来。通过使用批量转换来执行这些数据处理步骤，您通常可以消除其他预处理或后处理。您只能使用 JSON 和 CSV 格式的输入文件。

主题

- 将推理与输入记录相关联的工作流程 (p. 2232)
- 在批量转换作业中使用数据处理 (p. 2232)
- 支持的 JSONPath 运算符 (p. 2233)
- 批量转换示例 (p. 2233)

将推理与输入记录相关联的工作流程

下图显示将推理与输入记录相关联的工作流程。



要将推理与输入数据相关联，有三个主要步骤：

1. 筛选进行推理时不需要的输入数据，然后再将输入数据传递到批量转换作业。使用`InputFilter`参数使用哪些属性作为模型的输入。
2. 将输入数据和推理结果关联。使用`JoinSource`参数将输入数据与推理结合起来。
3. 筛选联接的数据，以保留用来为解释报告中的预测提供上下文所需的输入。使用`OutputFilter`使用存储输出文件中联接的数据集的指定部分。

在批量转换作业中使用数据处理

当使用创建批量转换作业时`CreateTransformJob`到进程数据：

1. 使用指定要传输到模型的部分。`InputFilter`中的参数`DataProcessing`数据结构。
2. 使用 `JoinSource` 参数联接原始输入数据和转换后的数据。
3. 使用指定批量转换作业中哪部分联接输入数据和转换后的数据要包含在输出文件中，使用`OutputFilter`参数。
4. 选择 JSON 或 CSV 格式的文件作为输入：
   - 对于 JSON 或 JSON 行格式的输入文件，SageMaker 会添加`SageMakerOutput`属性转换为输入文件，或者创建一个新的 JSON 输出文件，使用`SageMakerInput`和`SageMakerOutput`属性。有关更多信息，请参阅 `DataProcessing`。
   - 对于 CSV 格式的输入文件，联接的输入数据后跟转换后的数据，而输出是 CSV 文件。

如果你将算法与`DataProcessing`结构，它必须支持你选择的格式都输入和输出文件。例如，使用`TransformOutput`字段中的`CreateTransformJob`API，你必须同时设置`ContentType`和`Accept`参数为以下任一值：`text/csv`、`application/json`，或者`application/jsonlines`. 在 CSV 文件中指定列的语法与在 JSON 文件中指定属性的语法是不同的。使用错误的语法会导致错误。有关更多信息，请

参阅 批量转换示例 (p. 2233)。有关用于内置算法的输入和输出文件格式的更多信息，请参阅使用 Amazon SageMaker 内置算法或预训练模型 (p. 787)。

输入和输出的记录分隔符也必须符合所选的文件输入。这些区域有：`SplitType`参数指示如何拆分输入数据集中的记录。这些区域有：`AssembleWith`参数指示如何重组记录以进行输出。如果您将输入和输出格式设置为 `text/csv`，还必须将 `SplitType` 和 `AssembleWith` 参数设置为 `line`。如果您将输入和输出格式设置为 `application/jsonlines`，则可以将 `SplitType` 和 `AssembleWith` 这两者设置为 `line`。

对于 CSV 文件，不能使用嵌入的换行符。对于 JSON 文件，属性名称 `SageMakerOutput` 预留用于输出。JSON 输入文件不得具有使用此名称的属性。否则，输入文件中的数据可能被覆盖。

支持的 JSONPath 运算符

要筛选并联接输入数据和推理，请使用 JSONPath 子表达式。SageMaker 仅支持定义的 JSONPath 运算符子集。下表列出了支持的 JSONPath 运算符。对于 CSV 数据，每一行都被视为 JSON 数组，因此只能应用基于索引的 JSONPath，例如 `$[0]`、`$[1:]`。CSV 数据也应遵循 RFC 格式。

JSONPath 运算符	描述	示例
`$`	查询的根元素。所有路径表达式的开头都需要此运算符。	`$`
`.<name>`	一个以点表示的子元素。	`$.id`
`*`	一个通配符。用来代替属性名称或数值。	`$.id.*`
`['<name>' (,'<name>')]`	一个括号表示的元素或多个子元素。	`$['id','SageMakerOutput']`
`[<number> (,<number>)]`	一个索引或索引数组。也支持否定索引值。`-1` 索引指数组中的最后一个元素。	`$[1]`, `$[1,3,5]`
`[<start>:<end>]`	数组 Slice 运算符。数组 slice() 方法提取数组的一部分并返回一个新数组。如果你省略`<start>`，SageMaker 使用数组的第一个元素。如果你省略`<end>`，SageMaker 使用数组的最后一个元素。	`$[2:5]`, `$[:5]`, `$[2:]`

使用括号表示法指定给定字段的多个子元素时，不支持在括号内添加子元素。例如，支持 `$.field1.['child1','child2']` 而不支持 `$.field1.['child1','child2.grandchild']`。

有关 JSONPath 运算符的更多信息，请参阅 GitHub 上的 JsonPath。

批量转换示例

以下示例显示了一些将输入数据与预测结果联接的常用方法。

主题

- 例如：仅输出推理 (p. 2233)
- 例如：输出推断与输入数据联接 (p. 2234)
- 例如：输出推理与输入数据联接，并从输入中排除 ID 列 (CSV) (p. 2235)
- 例如：输出推理与 ID 列联接，并从输入中排除 ID 列 (CSV) (p. 2235)

例如：仅输出推理

默认为`DataProcessing`参数不将推理结果与输入联接。它仅输出推理结果。

如果你想明确指定不能用输入加入结果，u使用Amazon SageMaker Python 开发工具包在转换器通话中指定以下设置。

```
sm_transformer = sagemaker.transformer.Transformer(…)
sm_transformer.transform(…, input_filter="$", join_source= "None", output_filter="$")
```

要使用输出推理Amazon适用于 Python 的开发工具包，将以下代码添加到 CreateTransformJob 请求。下面的代码模仿了默认行为。

```
{
    "DataProcessing": {
        "InputFilter": "$",
        "JoinSource": "None",
        "OutputFilter": "$"
    }
}
```

例如：输出推断与输入数据联接

如果您使用的是Amazon SageMaker Python 开发工具包要将输入数据与输出文件中的推理结合起来，请指定`assemble_with`和`accept`初始化变压器对象时的参数。使用转换调用时，请指定`Input`(对于)`join_source`参数，然后指定`split_type`和`content_type`参数也可以。这些区域有：`split_type`参数必须具有与相同的值`assemble_with`，以及`content_type`参数必须具有与相同的值`accept`. 有关参数及其接受值的更多信息，请参阅转换器中的 “页面”Amazon SageMaker Python 开发工具包.

```
sm_transformer = sagemaker.transformer.Transformer(…, assemble_with="Line", accept="text/
csv")
sm_transformer.transform(…, join_source="Input", split_type="Line", content_type="text/
csv")
```

如果您使用的是Amazon适用于 Python 的 SDK (Boto 3)，通过将以下代码添加到您的推理，联接所有输入数据与推理。`CreateTransformJob`请求. 的值`Accept`和`ContentType`必须匹配，并且`AssembleWith`和`SplitType`还必须匹配。

```
{
    "DataProcessing": {
        "JoinSource": "Input"
    },
    "TransformOutput": {
        "Accept": "text/csv",
        "AssembleWith": "Line"
    },
    "TransformInput": {
        "ContentType": "text/csv",
        "SplitType": "Line"
    }
}
```

对于 JSON 或 JSON 行输入文件，结果位于输入 JSON 文件的 `SageMakerOutput` 键中。例如，如果输入是一个 JSON 文件，其中包含键值对 `{"key":1}`，则数据转换结果可能是 `{"label":1}`。

SageMaker 专卖店都在中的输入文件中`SageMakerInput`键。

```
{
    "key":1,
    "SageMakerOutput":{"label":1}
```

```
}
```

> **Note**
>
> JSON 的联接结果必须是键值对对象。如果输入不是键值对象，SageMaker 会创建一个新的 JSON 文件。在新的 JSON 文件中，输入数据存储在 `SageMakerInput` 键中，而结果存储为 `SageMakerOutput` 值。

对于 CSV 文件，例如，如果记录为`[1,2,3]`，标签结果是`[1]`，那么输出文件将包含`[1,2,3,1]`.

例如：输出推理与输入数据联接，并从输入中排除 ID 列 (CSV)

如果您使用的是Amazon SageMaker Python 开发工具包要将输入数据与推理输出联接，同时从转换器输入中排除 ID 列，请为`input_filter`在你的变压器通话中。例如，如果输入数据包含五列，而第一列是 ID 列，请使用以下转换请求选择除 ID 列以外的所有列作为要素。变压器仍然输出与推断连接的所有输入列。有关参数及其接受值的更多信息，请参阅转换器中的 “页面”Amazon SageMaker Python 开发工具包.

```
sm_transformer = sagemaker.transformer.Transformer(…, assemble_with="Line", accept="text/
csv")
sm_transformer.transform(…, split_type="Line", content_type="text/csv",
 input_filter="$[1:]", join_source="Input")
```

如果您使用的是Amazon适用于 Python 的 SDK (Boto 3)，请将以下代码添加到您的 `CreateTransformJob`请求.

```
{
    "DataProcessing": {
        "InputFilter": "$[1:]",
        "JoinSource": "Input"
    },
    "TransformOutput": {
        "Accept": "text/csv",
        "AssembleWith": "Line"
    },
    "TransformInput": {
        "ContentType": "text/csv",
        "SplitType": "Line"
    }
}
```

要在 SageMaker 中指定列，请使用数组元素的索引。第一列是索引 0，第二列是索引 1，第六列是索引 5。

要从输入中排除第一列，请将`InputFilter`到`"$[1:]"`. 冒号 (`:`) 告知 SageMaker 包含两个值（含这两个值）之间的所有元素。例如，`$[1:4]` 指定第二列到第五列。

如果您省略冒号后的数字，例如 `[5:]`，则子集包含第六列至最后一列的所有列。如果您省略冒号前的数字，例如 `[:5]`，则子集包含第一列（索引 0）至第六列的所有列。

例如：输出推理与 ID 列联接，并从输入中排除 ID 列 (CSV)

如果您使用的是Amazon SageMaker Python 开发工具包，您可以将输出指定为仅将特定输入列（例如 ID 列）与推断联接起来，方法是指定`output_filter`在变压器电话中。这些区域有：`output_filter`使用 jsonPath 子表达式来指定在将输入数据与推理结果联接后作为输出返回的列。以下请求显示了如何在排除 ID 列的同时进行预测，然后将 ID 列加入推断。请注意，在以下示例中，最后一列 (`-1`) 输出中包含推论。如果您使用的是 JSON 文件，则 SageMaker 将推理结果存储在属性中`SageMakerOutput`. 有关参数及其接受值的更多信息，请参阅转换器中的 “页面”Amazon SageMaker Python 开发工具包.

```
sm_transformer = sagemaker.transformer.Transformer(…, assemble_with="Line", accept="text/
csv")
```

```
sm_transformer.transform(…, split_type="Line", content_type="text/csv",
 input_filter="$[1:]", join_source="Input", output_filter="$[0,-1]")
```

如果您使用的是Amazon适用于 Python 的 SDK (Boto 3)，通过将以下代码添加到您的，联接 ID 列和推理。`CreateTransformJob`请求.

```
{
    "DataProcessing": {
        "InputFilter": "$[1:]",
        "JoinSource": "Input",
        "OutputFilter": "$[0,-1]"
    },
    "TransformOutput": {
        "Accept": "text/csv",
        "AssembleWith": "Line"
    },
    "TransformInput": {
        "ContentType": "text/csv",
        "SplitType": "Line"
    }
}
```

Warning

如果您使用的是 JSON 格式的输入文件，该文件不能包含属性名称 `SageMakerOutput`。此属性名称预留供输出文件中的推理使用。如果您的 JSON 格式的输入文件包含具有此名称的属性，则输入文件中的值可能会被推理覆盖。

Batch 存储转换

当您运行批量转换作业时，Amazon SageMaker 将一个 Amazon 弹性块存储库存储卷附加到处理您的任务的 Amazon EC2 实例。卷存储您的型号，存储卷的大小固定为 30 GB。您可以选择在存储卷中静态加密模型。

Note

如果你有大型模型，你可能会遇到`InternalServerError`.

有关 Amazon EBS 存储和功能的更多信息，请参阅以下页面：

- Amazon EBS适用于 Linux 实例的 Amazon EC2 用户指南中的 Amazon EC2 用
- Amazon EBS 卷适用于 Linux 实例的 Amazon EC2 用户指南中的 Amazon EC2 用

Note

g4dn 实例随附自己的本地 SSD 存储。要了解有关 G4DN 实例的更多信息，请参阅Amazon EC2 G4 实例页.

部署防护机制

部署护栏是 Amazon SageMaker Inference 中的一组模型部署选项，用于在生产环境中更新机器学习模型。使用完全托管的部署选项，您可以控制从生产中的当前模型到新模型的切换。流量转移模式（如 Canary 和线性）可让您在更新过程中精确控制从当前模型到新模型的流量转移过程。还有内置的安全措施，例如自动回滚，可帮助您及早发现问题，并在问题对生产产生重大影响之前自动采取纠正措施。

部署防护机制提供以下优势：

- 在更新生产环境时安全部署。对生产环境的回归更新可能会导致计划外停机和业务影响，例如增加模型延迟和高错误率。部署护栏通过提供最佳实践和内置的操作安全护栏来帮助您降低这些风险。
- 完全托管式部署。SageMaker 负责设置和协调这些部署，并将其与端点更新机制集成。您无需构建和维护编排、监控或回滚机制。您可以利用 SageMaker 来设置和安排这些部署，并专注于将机器学习用于应用程序。
- 可见性。您可以借助DescribeEndpointAPI 或通过 Amazon CloudWatch Events（支持的终端 (p. 2251)）。要了解有关 SageMaker 中事件的更多信息，请参阅中的端点部署状态更改部分使用 Amazon EventBridge 自动执行 Amazon SageMaker (p. 2823). 请注意，如果终端节点使用中的任何功能Exclusions (p. 2251)页面中，您无法使用 CloudWatch 事件。

  Note

  部署护栏仅适用于异步推理 (p. 2215)和实时推理 (p. 2102)终端节点类型。

如何开始

我们支持采用多种流量转移模式的蓝/绿部署 (p. 2239). 流量转移模式是指定 SageMaker 如何将端点流量路由到包含更新的新队列的配置。以下流量转移模式为您提供了对终端节点更新过程的不同级别的控制：

- 蓝/绿：一次部署全部 (p. 2240)将所有终端流量从蓝色机队转移到绿色车队。流量转移到绿色队列后，您预先指定的 Amazon CloudWatch 警报开始监控绿色队列一段时间（烘焙时间）。如果在烘焙期间没有警报跳闸，则 SageMaker 将终止蓝色舰队。
- 蓝/绿：加纳利 (p. 2243)让你转移一小部分流量（aCanary）到绿色舰队并监控它一段烘焙期。如果金丝雀在绿色舰队上成功，那么 SageMaker 会在终止蓝色舰队之前将其余流量从蓝色舰队转移到绿色舰队。
- 蓝/绿：线性 (p. 2247)针对流量转移步骤的数量和每个步骤要转移的流量百分比提供了更多的自定义。虽然 Canary 移动可以让你分两步转移流量，但线性移动将其延伸到n线性间隔的步骤。

您可以通过UpdateEndpoint和CreateEndpointSageMaker API 和Amazon Command Line Interface命令。有关如何设置部署的更多详细信息，请参阅各个部署页面。请注意，如果终端节点使用中列出的任何功能Exclusions (p. 2251)页面中，您不能使用部署护栏。

要遵循演示如何使用部署护栏的指导示例，请参阅我们的示例Jupyter 笔记本用于金丝雀和线性交通转移模式。

自动回滚配置和监控

Amazon CloudWatch 警报是在部署护栏中使用烘焙期的先决条件。只有在设置可监控终端节点的 CloudWatch 警报时，才能在部署护栏中使用自动回滚功能。如果您的任何警报在指定监控期间出现故障，SageMaker 会启动对旧终端节点的完全回滚以保护您的应用程序。如果您没有设置任何 CloudWatch 警报来监控终端节点，则自动回滚功能在部署期间不起作用。

要了解有关 Amazon CloudWatch 的更多信息，请参什么是 Amazon CloudWatch？中的Amazon CloudWatch 用户指南.

告警示例

为了帮助你入门，我们提供了以下示例来演示 CloudWatch 警报的功能。除了使用或修改以下示例外，您还可以创建自己的警报并配置警报，以便在一段时间内监控指定车队上的各种指标。要查看可以添加到警报中的更多 SageMaker 指标和维度，请参阅使用 Amazon CloudWatch 监控 Amazon SageMaker (p. 2810).

主题

- 监控新旧车队的调用错误 (p. 2238)
- 监控新车队的模型延迟 (p. 2238)

监控新旧车队的调用错误

以下 CloudWatch 警报监控终端节点的平均错误率。您可以将此警报用于任何部署护栏流量转移类型，以提供对新旧车队的全面监控。如果警报出现，则 SageMaker 会启动回滚到旧机队。

来自旧车队和新车队的调用错误导致了平均错误率。如果平均错误率超过指定的阈值，则警报会跳动。此特殊示例监视部署期间旧队列和新队列上的 4xx 错误（客户端错误）。您还可以使用指标监控 5xx 错误（服务器错误）`Invocation5XXErrors`.

> Note
>
> 对于此警报类型，如果您的旧机队在部署期间跳过警报，SageMaker 将终止您的部署。因此，如果您当前的生产队列已导致错误，请考虑使用或修改以下仅监控新队列中的错误的示例之一。

```
#Applied deployment type: all types
{
    "AlarmName": "EndToEndDeploymentHighErrorRateAlarm",
    "AlarmDescription": "Monitors the error rate of 4xx errors",
    "MetricName": "Invocation4XXErrors",
    "Namespace": "AWS/SageMaker",
    "Statistic": "Average",
    "Dimensions": [
        {
            "Name": "EndpointName",
            "Value": <your-endpoint-name>
        },
        {
            "Name": "VariantName",
            "Value": "AllTraffic"
        }
    ],
    "Period": 600,
    "EvaluationPeriods": 2,
    "Threshold": 1,
    "ComparisonOperator": "GreaterThanThreshold",
    "TreatMissingData": "notBreaching"
}
```

在前面的示例中，请注意以下字段的值：

- 适用于`AlarmName`和`AlarmDescription`中，输入您为警报选择的名称和说明。
- 适用于`MetricName`，使用该值`Invocation4XXErrors`监控终端上的 4xx 错误
- 适用于`Namespace`，使用该值`AWS/SageMaker`. 如果适用，您还可以指定自己的自定义指标。
- 对于 `Statistic`，请使用 `Average`。这意味着在计算错误率是否超过阈值时，警报采用评估期间的平均错误率。
- 对于维度`EndpointName`中，使用要更新的终端节点的名称作为值。
- 对于维度`VariantName`，使用该值`AllTraffic`以指定所有终端节点流量。
- 对于 `Period`，请使用 `600`。这将警报的评估周期设置为 10 分钟。
- 对于 `EvaluationPeriods`，请使用 `2`。此值告诉警报在确定警报状态时考虑最近两个评估周期。

监控新车队的模型延迟

以下 CloudWatch 警报示例监控部署期间新队列的模型延迟。您可以使用此警报仅监控新车队并排除旧车队。警报持续到整个部署。此示例为您提供了对新机队的全面、端到端的监控，如果新机队出现任何响应时间问题，则启动回滚到旧车队。

CloudWatch 发布具有维度的指标`EndpointConfigName:{New-Ep-Config}`在新队列开始接收流量之后，这些指标即使在部署完成后也会持续。

您可以在任何部署类型中使用以下警示例。

```
#Applied deployment type: all types
{
    "AlarmName": "NewEndpointConfigVersionHighModelLatencyAlarm",
    "AlarmDescription": "Monitors the model latency on new fleet",
    "MetricName": "ModelLatency",
    "Namespace": "AWS/SageMaker",
    "Statistic": "Average",
    "Dimensions": [
        {
            "Name": "EndpointName",
            "Value": <your-endpoint-name>
        },
        {
            "Name": "VariantName",
            "Value": "AllTraffic"
        },
        {
            "Name": "EndpointConfigName",
            "Value": <your-config-name>
    ],
    "Period": 300,
    "EvaluationPeriods": 2,
    "Threshold": 100000, # 100ms
    "ComparisonOperator": "GreaterThanThreshold",
    "TreatMissingData": "notBreaching"
}
```

在前面的示例中，请注意以下字段的值：

- 适用于`MetricName`，使用该值`ModelLatency`以监控模型的响应时间。
- 适用于`Namespace`，使用该值`AWS/SageMaker`. 如果适用，您还可以指定自己的自定义指标。
- 对于维度`EndpointName`中，使用要更新的终端节点的名称作为值。
- 对于维度`VariantName`，使用该值`AllTraffic`以指定所有终端节点流量。
- 对于维度`EndpointConfigName`，该值应参考新的或更新的终端节点的终端节点配置名称。

> **Note**
>
> 如果您想监控旧车队而不是新车队，则可以更改维度`EndpointConfigName`以指定旧机队配置的名称。

蓝/绿部署

当您更新终端节点时，Amazon SageMaker 会自动使用蓝/绿部署来最大限度地提高终端节点的可用性。在蓝色/绿色部署中，SageMaker 配置一个包含更新的新机队（绿色机队）。然后，SageMaker 将流量从旧车队（蓝色舰队）转移到绿色舰队。一旦绿色车队在设定的评估期内（称为烘焙期）顺利运行，SageMaker 将终止蓝色舰队。借助蓝/绿部署中的附加功能，您可以利用流量转移模式和自动回滚监控来保护终端免受重大生产影响。

下图描述了 SageMaker 中蓝/绿部署的主要功能：

- 交通转移模式。通过部署护栏的流量转移模式，您可以控制蓝队和绿色车队之间的流量和流量转移步数。此功能使您能够逐步评估绿色车队的表现，而无需完全承诺实现 100% 的流量转移。
- 烘焙时期。烘焙期是在进入下一个部署阶段之前监控绿色车队的固定时间。如果任何预先指定的警报在任何烘焙期间出现，则所有端点流量都会回滚到蓝色队列。烘焙期可以帮助您在使流量转变永久性之前建立对更新的信心。

- 自动回滚。您可以指定 SageMaker 用于监控绿色队列的 Amazon CloudWatch 警报。如果更新代码的问题出现任何警报，SageMaker 会启动对蓝色机队的自动回滚，以保持可用性，从而最大限度地降低风险。

交通转移模式

蓝/绿部署中的各种流量转移模式可让您更精细地控制蓝队和绿色车队之间的流量转移。蓝/绿部署的可用流量转移模式都是一次性、金丝雀和线性的。下表显示了选项的比较。

> **Important**
>
> 对于涉及多阶段流量转移或烘焙周期的蓝色/绿色部署，无论队列流量如何，都需要在更新期间为两个队列付费。这与蓝色/绿色部署形成鲜明对比，所有流量一次性转移且没有烘焙期，在更新过程中，您只需为一个队列付费。

名称	什么是？	Pary	缺点	建议
一次部署全部	只需一步即可将所有流量转移到新车队。	最大限度地缩短整体更新时间。	回归更新会影响 100% 的流量。	使用此选项可以将更新时间和成本降至最低。
Canary	交通转移分两步。第一个（金丝雀）步骤转移了一小部分流量，然后是第二步，这将转移其余流量。	将回归更新的爆炸半径限制为仅限于金丝雀舰队。	两个车队在整个部署中都同时运行。	使用此选项可以在最小化回归更新的爆炸半径和最大限度地减少两个队列的运行时间之间进行平衡。
线性	固定部分流量会以预先指定数量相等间隔的步数转移。	通过将流量转移到几个步骤，最大限度地降低回归更新的风险。	更新时间和成本与步骤数成正比。	使用此选项可以将部署分散到多个步骤，从而最大限度地降低

入门

指定所需的部署配置后，SageMaker 将为您处理配置新实例、终止旧实例和转移流量。您可以通过现有的UpdateEndpoint和CreateEndpointSageMaker API 和Amazon Command Line Interface命令。请注意，如果终端节点使用中列出的任何功能Exclusions (p. 2251)页面中，您不能使用部署护栏。有关如何设置部署的更多详细信息，请参阅各个部署页面：

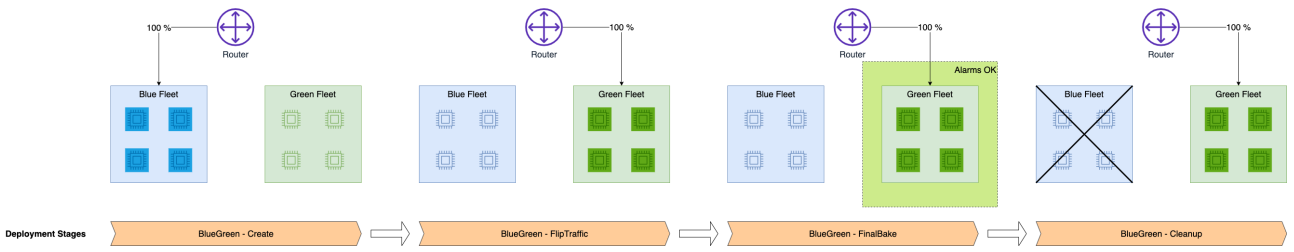
- 蓝/绿更新与一次性流量转移 (p. 2240)
- 带Canary 流量转移的蓝/绿更新 (p. 2243)
- 带线性流量转移的蓝/绿更新 (p. 2243)

要遵循演示如何使用部署护栏的指导示例，请参阅我们的示例Jupyter 笔记本用于金丝雀和线性交通转移模式。

一次性交通转移

随着所有流量一次性转移，您可以使用蓝/绿部署的安全护栏快速推出端点更新。您可以使用此流量转移选项来最大限度地缩短更新持续时间，同时仍可利用蓝/绿部署的可用性保证。烘焙期功能可帮助您在终止旧实例之前监控新实例的性能和功能，确保您的新队列全面运行。

下图显示了所有流量转移如何同时管理新旧车队。



当您同时使用所有流量转移时，SageMaker 会将 100% 的流量路由到新车队（绿色车队）。一旦绿色车队开始接收流量，烘焙期就开始了。烘焙期是预先指定的 Amazon CloudWatch 警报监控绿色队列性能的设定时间。如果在烘焙期间没有警报行程，SageMaker 将终止旧机队（蓝色舰队）。如果在烘焙期间出现任何警报，则会启动自动回滚，100% 的流量转移回蓝队。

先决条件

在设置所有流量一次性转移的部署之前，您必须创建 Amazon CloudWatch 警报以监视来自终端节点的指标。如果任何警报在烘焙期间出现，那么流量将回退到蓝色舰队。要了解如何在终端节点上设置 CloudWatch 警报，请参阅先决条件页面自动回滚配置和监控 (p. 2237). 要了解有关 CloudWatch 警报的更多信息，请使用 Amazon CloudWatch 告警中的Amazon CloudWatch 用户指南.

一次性配置所有流量转移

准备好部署并为终端节点设置 CloudWatch 警报后，您可以使用 SageMakerUpdateEndpointAPI 或更新终端节点中的命令Amazon Command Line Interface以启动部署。

主题

- 如何更新终端节点 (API) (p. 2242)
- 如何使用现有的蓝/绿更新策略 (API) 更新终端节点 (p. 2243)
- 如何更新终端节点 (CLI) (p. 2243)

如何更新终端节点 (API)

以下示例显示如何将所有流量同时转移更新终端节点，使用UpdateEndpoint在 Amazon SageMaker API 中。

```
import boto3
client = boto3.client("sagemaker")

response = client.update_endpoint(
    EndpointName="<your-endpoint-name>",
    EndpointConfigName="<your-config-name>",
    DeploymentConfig={
        "BlueGreenUpdatePolicy": {
            "TrafficRoutingConfiguration": {
                "Type": "ALL_AT_ONCE"
            },
            "TerminationWaitInSeconds": 600,
            "MaximumExecutionTimeoutInSeconds": 1800
        },
        "AutoRollbackConfiguration": {
            "Alarms": [
                {
                    "AlarmName": "<your-cw-alarm>"
                },
            ]
        }
    }
)
```

要配置全部一次性流量转移选项，请执行以下操作：

- 适用于`EndpointName`，请使用要更新的现有终端节点的名称。
- 适用于`EndpointConfigName`，请使用要使用的终端节点配置的名称。
- 在`DeploymentConfig`和`BlueGreenUpdatePolicy`，在`TrafficRoutingConfiguration`中，设置`Type`参数到`ALL_AT_ONCE`. 这指定部署一次使用全部流量转移模式。
- 对于 `TerminationWaitInSeconds`，请使用 `600`。此参数告诉 SageMaker 在绿色队列完全激活后等待指定的时间（以秒为单位），然后终止蓝色队列中的实例。在此示例中，SageMaker 在最后烘焙期结束后等待 10 分钟才能终止蓝色舰队。
- 对于 `MaximumExecutionTimeoutInSeconds`，请使用 `1800`。此参数设置部署在超时之前可以运行的最长时间。在前面的示例中，您的部署的完成时间限制为 30 分钟。
- In`AutoRollbackConfiguration`，在`Alarms`字段中，您可以按名称添加 CloudWatch 警报。创建一个`AlarmName`： *`<your-cw-alarm>`*对于要使用的每个警报的条目。

如何使用现有的蓝/绿更新策略 (API) 更新终端节点

当您使用CreateEndpoint要创建终端节点的 API，您可以选择指定一个部署配置，以便将来的终端节点更新重用。您可以使用相同的`DeploymentConfig`选项如上一个 UpdateEndpoint API 示例。CreateEndpoint API 行为没有变化。指定部署配置不会在终端节点上自动执行蓝/绿更新。

使用之前的部署配置的选项是在使用UpdateEndpoint用于更新终端节点的 API。更新终端节点时，您可以使用`RetainDeploymentConfig`选项来保留您在创建终端节点时指定的部署配置。

调用时UpdateEndpointAPI，设置`RetainDeploymentConfig`到`True`要保留`DeploymentConfig`来自原始终端节点配置的选项。

```
response = client.update_endpoint(
    EndpointName="<your-endpoint-name>",
    EndpointConfigName="<your-config-name>",
    RetainDeploymentConfig=True
)
```

如何更新终端节点 (CLI)

如果您使用Amazon CLI，以下示例展示了如何使用更新终端节点命令。

```
update-endpoint
--endpoint-name <your-endpoint-name>
--endpoint-config-name <your-config-name>
--deployment-config '"BlueGreenUpdatePolicy": {"TrafficRoutingConfiguration": {"Type":
 "ALL_AT_ONCE"},
    "TerminationWaitInSeconds": 600, "MaximumExecutionTimeoutInSeconds": 1800},
    "AutoRollbackConfiguration": {"Alarms": [{"AlarmName": "<your-alarm>"}]}'
```

要配置全部一次性流量转移选项，请执行以下操作：

- 适用于`endpoint-name`，请使用要更新的终端节点的名称。
- 适用于`endpoint-config-name`，请使用要使用的终端节点配置的名称。
- 适用于`deployment-config`，使用BlueGreenUpdatePolicyJSON 对象。

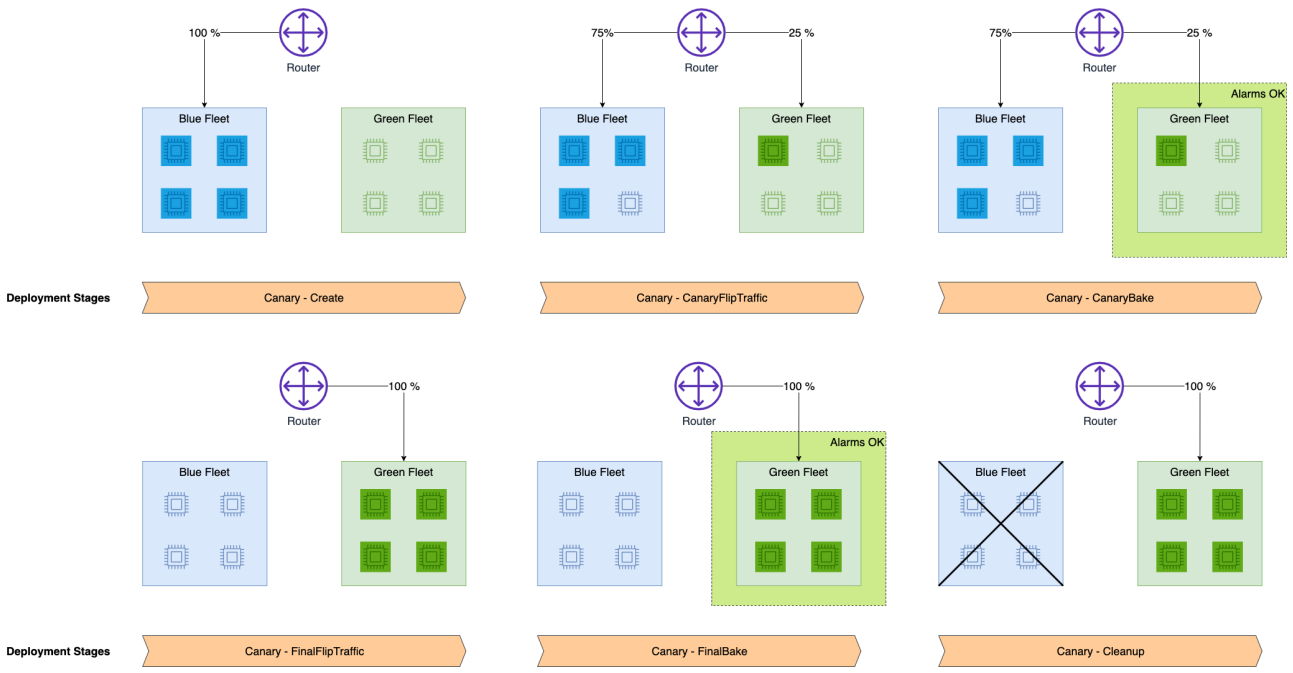
> **Note**
>
> 如果你想将 JSON 对象保存在文件中，请参阅生成Amazon CLI框架和输入参数中的Amazon CLI用户指南.

Canary 流量转移

随着 Canary 流量转移，您可以在新车队上测试一部分端点流量，而旧车队则为其余流量提供服务。此测试步骤是一条安全护栏，在将所有流量转移到新车队之前验证新车队的功能。您仍然可以享受蓝/绿部署的好处，而且添加的 Canary 功能可以让您确保新（绿色）队列在让其处理 100% 的流量之前可以提供推断。

你的绿色舰队中开启接收流量的部分称为金丝雀，你可以选择这个金丝雀的大小。请注意，Canary 规模应小于或等于新队组容量的 50%。烘焙期结束后，没有预先指定的 Amazon CloudWatch 警报行程，其余流量将从旧（蓝色）队列转移到绿色队列。Canary 流量转移为您在部署过程中提供了更高的安全性，因为更新模型的任何问题都只会影响 Canary。

下图显示了金丝雀流量转移如何管理蓝色和绿色车队之间的流量分布。


100 %
Router
Blue Fleet
Green Fleet
75%
25 %
Alarms OK
Deployment Stages
Canary - Create
Canary - CanaryFlipTraffic
Canary - CanaryBake
100 %
Canary - FinalFlipTraffic
Canary - FinalBake
Canary - Cleanup

一旦 SageMaker 配置了绿色车队，SageMaker 会将一部分传入流量（例如 25%）路由到金丝雀。然后烘焙期开始，在此期间，您的 CloudWatch 警报将监控绿色队列的性能。在此期间，蓝色舰队和绿色舰队都处于部分活动状态并接收流量。如果在烘焙期间出现任何警报行程，则 SageMaker 会启动回滚，所有流量都返回蓝队。如果没有警报行程，那么所有的交通都转移到绿色车队，并且有最后的烘焙期。如果最后烘焙期结束时没有跳闸任何警报，那么绿色车队将为所有流量提供服务，SageMaker 将终止蓝色舰队。

先决条件

在使用 Canary 流量转移设置部署之前，您必须创建 Amazon CloudWatch 警报以监控来自终端节点的指标。警报在烘焙期间处于活动状态，如果有警报出现，则所有终端流量都会回滚到蓝色队列。要了解如何在终端节点上设置 CloudWatch 警报，请参阅先决条件页面自动回滚配置和监控 (p. 2237). 要详细了解 CloudWatch 警报，请参阅使用 Amazon CloudWatch 告警中的Amazon CloudWatch 用户指南.

配置 Canary 流量转移

准备好部署并为终端节点设置 Amazon CloudWatch 警报后，您可以使用 Amazon SageMakerUpdateEndpointAPI 或更新终端节点中的命令Amazon CLI以启动部署。

主题

- 如何更新终端节点 (API) (p. 2245)
- 如何使用现有的蓝/绿更新策略 (API) 更新终端节点 (p. 2246)
- 如何更新终端节点 (CLI) (p. 2246)

如何更新终端节点 (API)

下面的示例UpdateEndpointAPI 显示了如何使用 Canary 流量转移来更新终端节点。

```
import boto3
client = boto3.client("sagemaker")

response = client.update_endpoint(
    EndpointName="<your-endpoint-name>",
    EndpointConfigName="<your-config-name>",
    DeploymentConfig={
        "BlueGreenUpdatePolicy": {
            "TrafficRoutingConfiguration": {
                "Type": "CANARY",
                "CanarySize": {
                    "Type": "CAPACITY_PERCENT",
                    "Value": 30
                },
                "WaitIntervalInSeconds": 600
            },
            "TerminationWaitInSeconds": 600,
            "MaximumExecutionTimeoutInSeconds": 1800
        },
        "AutoRollbackConfiguration": {
            "Alarms": [
                {
                    "AlarmName": "<your-cw-alarm>"
                }
            ]
        }
    }
)
```

要配置 Canary 流量转移选项，执行以下操作：

- 适用于EndpointName中，使用要更新的现有终端节点的名称。

- 适用于EndpointConfigName中，使用要使用的终端节点配置的名称。
- 下DeploymentConfig和BlueGreenUpdatePolicy，在TrafficRoutingConfiguration中，设置Type参数CANARY. 这指定了部署使用 Canary 流量转移。
- 在CanarySize字段中，您可以通过修改 CanaryType和Value参数。适用于Type，使用CAPACITY_PERCENT，意思是你想用作金丝雀的绿色舰队的百分比，然后设置Value到30. 在此示例中，您将绿色舰队的 30% 容量用作金丝雀。请注意，金丝雀规模应等于或小于绿色车队容量的 50%。
- 对于 WaitIntervalInSeconds，请使用 600。该参数告诉 SageMaker 在每次间隔移位之间等待指定的时间（以秒为单位）。此时间间隔是金丝雀烘焙期的持续时间。在前面的示例中，SageMaker 在金丝雀转移后等待 10 分钟，然后完成第二个也是最后一次交通转移。
- 对于 TerminationWaitInSeconds，请使用 600。此参数告诉 SageMaker 在绿色队列完全激活后等待指定的时间（以秒为单位），然后终止蓝色队列中的实例。在此示例中，SageMaker 在最后烘焙期结束后等待 10 分钟才能终止蓝色舰队。
- 对于 MaximumExecutionTimeoutInSeconds，请使用 1800。此参数设置部署在超时之前可以运行的最长时间。在前面的示例中，您的部署的完成时间限制为 30 分钟。
- InAutoRollbackConfiguration，在Alarms字段中，您可以按名称添加 CloudWatch 警报。创建一个AlarmName: *<your-cw-alarm>*对于您要使用的每个警报的条目。

如何使用现有的蓝/绿更新策略 (API) 更新终端节点

当您使用CreateEndpoint要创建终端节点的 API，您可以选择指定一个部署配置，以便将来的终端节点更新重用。您可以使用相同的DeploymentConfig选项如上一个 UpdateEndpoint API 示例。CreateEndpoint API 行为没有变化。指定部署配置不会在终端节点上自动执行蓝/绿更新。

使用之前的部署配置的选项是在使用UpdateEndpoint用于更新终端节点的 API。更新终端节点时，您可以使用RetainDeploymentConfig选项来保留您在创建终端节点时指定的部署配置。

当打电话时UpdateEndpoint设置 APIRetainDeploymentConfig到True要保留DeploymentConfig来自原始终端节点配置的选项。

```
response = client.update_endpoint(
    EndpointName="<your-endpoint-name>",
    EndpointConfigName="<your-config-name>",
    RetainDeploymentConfig=True
)
```

如何更新终端节点 (CLI)

如果您使用Amazon CLI，以下示例说明了如何使用更新终端节点命令。

```
update-endpoint
--endpoint-name <your-endpoint-name>
--endpoint-config-name <your-config-name>
--deployment-config '"BlueGreenUpdatePolicy": {"TrafficRoutingConfiguration": {"Type":
 "CANARY",
    "CanarySize": {"Type": "CAPACITY_PERCENT", "Value": 30}, "WaitIntervalInSeconds": 600},
    "TerminationWaitInSeconds": 600, "MaximumExecutionTimeoutInSeconds": 1800},
    "AutoRollbackConfiguration": {"Alarms": [{"AlarmName": "<your-alarm>"}]}'
```

要配置 Canary 流量转移选项，执行以下操作：

- 适用于endpoint-name中，使用要更新的终端节点的名称。
- 适用于endpoint-config-name中，使用要使用的终端节点配置的名称。
- 适用于deployment-config，使用BlueGreenUpdatePolicyJSON 对象。

Note

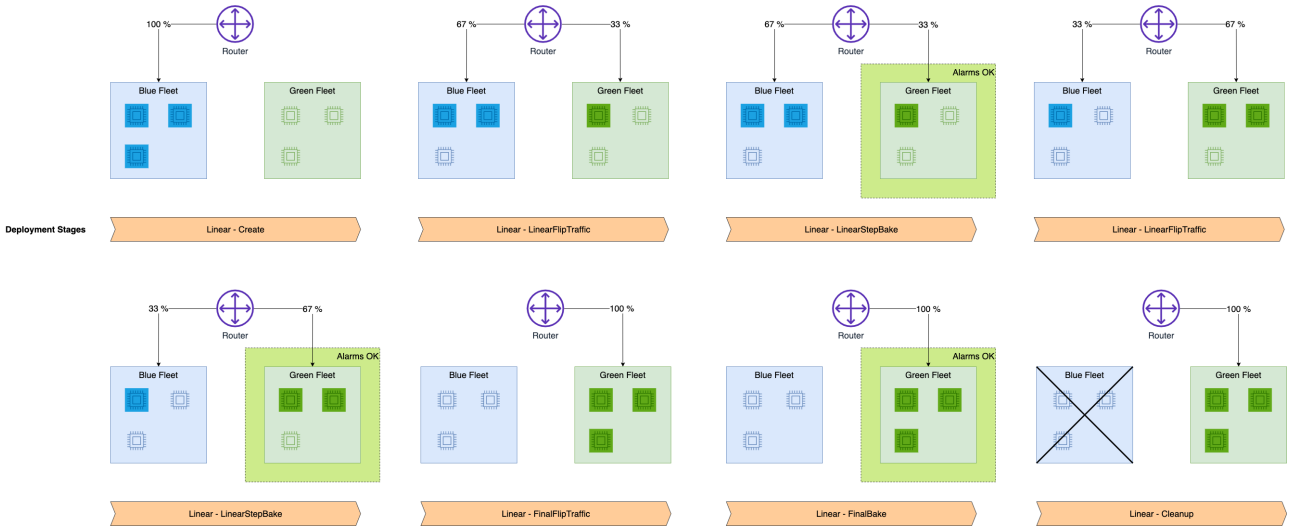
如果你宁愿将 JSON 对象保存在文件中，请参阅生成Amazon CLI框架和输入参数中的Amazon CLI用户指南.

线性交通转移

线性流量转移使您能够逐步将流量从旧车队（蓝色舰队）转移到新车队（绿色车队）。通过线性流量转移，您可以分多个步骤转移流量，从而最大限度地减少终端中断的可能性。此蓝/绿部署选项可让您对流量转移进行最精细的控制。

您可以选择实例数量或绿色队列在每个步骤中激活容量的百分比。每个线性步骤应该只能在绿色舰队容量的10-50% 之间。对于每个步骤，都有一个烘焙期，在此期间，您预先指定的 Amazon CloudWatch 警报会监控绿色队列上的指标。烘焙期结束且没有警报行程后，绿色车队的活跃部分将继续接收流量，并开始新的一步。如果警报在任何烘焙期间出现故障，则 100% 的端点流量会回退到蓝色队列。

下图显示了线性交通如何将流量转移到蓝色和绿色车队。


100 %
Router
Blue Fleet
Green Fleet
67 %
33 %
Alarms OK
Deployment Stages
Linear - Create
Linear - LinearFlipTraffic
Linear - LinearStepBake
Linear - FinalFlipTraffic
Linear - FinalBake
Linear - Cleanup

一旦 SageMaker 配置了新车队，绿色车队的第一部分将开启并接收流量。SageMaker 将停用蓝色舰队中相同尺寸的部分，烘焙期开始。如果有警报出现，所有终端流量都会回滚到蓝色队列。如果烘焙期结束，那么下一步开始。绿色舰队的另一部分激活并接收流量，蓝色舰队的一部分停用，另一个烘焙期开始。同样的过程会重复，直到蓝色舰队完全停用且绿色车队完全激活并接收所有流量。如果警报在任何时候发出，SageMaker 将终止转移过程，100% 的流量回退到蓝色舰队。

先决条件

在设置具有线性流量转移的部署之前，您必须创建 CloudWatch 警报以监控来自终端节点的指标。警报在烘焙期间处于活动状态，如果有警报出现，则所有终端流量都会回滚到蓝色队列。要了解如何在终端节点上设置 CloudWatch 警报，请参阅先决条件页面。自动回滚配置和监控 (p. 2237). 要了解有关 CloudWatch 警报的更多信息，请使用 Amazon CloudWatch 告警中的Amazon CloudWatch 用户指南.

配置线性流量转移

准备好部署并为终端节点设置 CloudWatch 警报后，您可以使用 Amazon SageMakerUpdateEndpointAPI 或更新终端节点中的命令Amazon CLI以启动部署。

主题

- 如何更新终端节点 (API) (p. 2249)
- 如何使用现有的蓝/绿更新策略 (API) 更新终端节点 (p. 2250)
- 如何更新终端节点 (CLI) (p. 2246)

如何更新终端节点 (API)

以下示例中的UpdateEndpointAPI 显示了如何使用线性流量转移更新终端节点。

```
import boto3
client = boto3.client("sagemaker")

response = client.update_endpoint(
    EndpointName="<your-endpoint-name>",
    EndpointConfigName="<your-config-name>",
    DeploymentConfig={
        "BlueGreenUpdatePolicy": {
            "TrafficRoutingConfiguration": {
                "Type": "LINEAR",
                "LinearStepSize": {
                    "Type": "CAPACITY_PERCENT",
                    "Value": 20
                },
                "WaitIntervalInSeconds": 300
            },
            "TerminationWaitInSeconds": 300,
            "MaximumExecutionTimeoutInSeconds": 3600
        },
        "AutoRollbackConfiguration": {
            "Alarms": [
                {
                    "AlarmName": "<your-cw-alarm>"
                }
            ]
        }
    }
)
```

要配置线性流量转移选项，请执行以下操作：

- 适用于EndpointName中，请使用要更新的现有终端节点的名称。

- 适用于EndpointConfigName中，请使用要使用的终端节点配置的名称。
- 在DeploymentConfig和BlueGreenUpdatePolicy，在TrafficRoutingConfiguration中，设置Type要参数LINEAR. 这指定了部署使用线性流量转移。
- 在LinearStepSize字段中，您可以修改步骤的大小，方法是修改步骤的大小。Type和Value参数。适用于Type，使用CAPACITY_PERCENT，意思是希望用作步长大小的绿色车队的百分比，然后设置Value到20. 在此示例中，您为每个流量转移步骤启用绿色车队容量的 20%。请注意，在自定义线性步长时，您应该只使用占绿色车队容量 10-50% 的步骤。
- 对于 WaitIntervalInSeconds，请使用 300。该参数告诉 SageMaker 在每次流量转移之间等待指定的时间（以秒为单位）。此间隔是每个线性步骤之间的烘焙周期的持续时间。在前面的示例中，SageMaker 在每次流量转移之间等待 5 分钟。
- 对于 TerminationWaitInSeconds，请使用 300。此参数告诉 SageMaker 在绿色队列完全激活后等待指定的时间（以秒为单位），然后终止蓝色队列中的实例。在此示例中，SageMaker 在最后烘焙期结束后等待 5 分钟才能终止蓝色舰队。
- 对于 MaximumExecutionTimeoutInSeconds，请使用 3600。此参数设置部署在超时之前可以运行的最长时间。在前面的示例中，您的部署的完成时间限制为 1 小时。
- InAutoRollbackConfiguration，在Alarms字段中，您可以按名称添加 CloudWatch 警报。创建一个AlarmName: *<your-cw-alarm>*对要使用的每个警报的条目。

如何使用现有的蓝/绿更新策略 (API) 更新终端节点

当您使用CreateEndpoint要创建终端节点的 API，您可以选择指定一个部署配置，以便将来的终端节点更新重用。您可以使用相同的DeploymentConfig选项如上一个 UpdateEndpoint API 示例。CreateEndpoint API 行为没有变化。指定部署配置不会在终端节点上自动执行蓝/绿更新。

使用之前的部署配置的选项是在使用UpdateEndpoint用于更新终端节点的 API。更新终端节点时，您可以使用RetainDeploymentConfig选项来保留您在创建终端节点时指定的部署配置。

当打电话时UpdateEndpointAPI，设置RetainDeploymentConfig到True要保留DeploymentConfig来自原始终端节点配置的选项。

```
response = client.update_endpoint(
    EndpointName="<your-endpoint-name>",
    EndpointConfigName="<your-config-name>",
    RetainDeploymentConfig=True
)
```

如何更新终端节点 (CLI)

如果您使用的是Amazon CLI，以下示例说明了如何使用更新终端节点命令。

```
update-endpoint
--endpoint-name <your-endpoint-name>
--endpoint-config-name <your-config-name>
--deployment-config '{"BlueGreenUpdatePolicy": {"TrafficRoutingConfiguration": {"Type":
 "LINEAR",
    "LinearStepSize": {"Type": "CAPACITY_PERCENT", "Value": 20}, "WaitIntervalInSeconds":
 300},
    "TerminationWaitInSeconds": 300, "MaximumExecutionTimeoutInSeconds": 3600},
    "AutoRollbackConfiguration": {"Alarms": [{"AlarmName": "<your-alarm>"}]}'
```

要配置线性流量转移选项，请执行以下操作：

- 适用于endpoint-name中，请使用要更新的终端节点的名称。
- 适用于endpoint-config-name中，请使用要使用的终端节点配置的名称。
- 适用于deployment-config，使用BlueGreenUpdatePolicyJSON 对象。

Note

如果你想将 JSON 对象保存在文件中，请参阅生成Amazon CLI框架和输入参数中的Amazon CLI用户指南.

Exclusions

您只能对新创建的终端节点使用部署护栏。还有一些基于功能的排除项，这使得您的终端节点目前与部署护栏不兼容。如果您的终端节点使用以下任何功能，则无法在终端节点上使用部署护栏，并且您的终端节点将回退到使用蓝/绿部署，所有流量一次性转移而没有最后的烘焙期。

- Marketplace 容器
- 多容器终端节点
- 多模型终端节点
- 多变体终端节点
- 使用 Inf1（基于推理的）实例的终端节点
- 使用 Amazon SageMaker 模型监视器的终端节点（已启用数据捕获）
- Amazon Elastic Inference 终端节点

推理成本优化最佳实践

以下内容提供了优化终端成本的技术和注意事项。您可以使用这些建议来优化新端点和现有终端节点的成本。

最佳实践

要优化您的 SageMaker 请遵循以下最佳实践。

选择工作的最佳推理选择。

SageMaker 提供了 4 种不同的推理选项，为工作提供最佳的推理选项。通过选择最适合您的工作负载的推理选项，您可以节省成本。

- 使用实时推理适用于具有可预测的流量模式的低延迟工作负载，这些模式需要具有一致的延迟特性并且 您为使用实例付费。
- 使用无服务器推理适用于具有尖峰流量模式且可以接受 p99 延迟变化的同步工作负载。无服务器推理会自动扩展以满足您的工作负载流量，因此您无需为任何闲置资源付费。您仅需为推理请求的持续时间付费。同一模型和容器可以用于实时推理和无服务器推理，因此，如果需求发生变化，您可以在这两种模式之间切换。
- 使用异步推理适用于处理高达 1 GB 的数据（例如文本语料库、图像、视频和音频）且对延迟不敏感且成本敏感的异步工作负载。借助异步推理，您可以通过指定固定数量的实例以获得最佳处理速率，而不是为峰值预配置来控制成本。您还可以缩小到零以节省额外成本。
- 使用批量推理对于需要推断离线进程的大量数据集的工作负载（也就是说，您不需要持久终端节点）。您在批量推理作业的持续时间内为实例付费。

选择加入 SageMaker Savings Plan。

- 如果你对所有人都具有一致的使用水平 SageMaker 服务，你可以选择加入 SageMaker Savings Plan 可帮助将成本降低高达 64%。
- 亚马逊 SageMaker Savings Plans为 Amazon SageMaker 提供灵活的定价模式，换取在一年或三年期限内的稳定使用量承诺（以 USD/ 小时为单位衡量）。这些计划自动应用于符合条件 SageMaker ML 实例使用情况包括 SageMaker Studio 笔记本 SageMaker 按需笔记本 SageMaker 处理， SageMaker 数据牧马

人， SageMaker 训练， SageMaker 以及实时推理 SageMaker 无论实例系列、大小或区域如何，都可以 Batch 转换。例如，您可以随时将在美国东部（俄亥俄州）运行的 CPU ml.c5.xlarge 实例更改为美国西部（俄勒冈州）的 ml.Inf1 实例以进行推理工作负载，并自动继续支付 Savings Plans 价格。

优化模型以更好地运行。

- 未优化的模型可能会导致更长的运行时间并使用更多资源。您可以选择使用更多或更大的实例来提高性能；但是，这会导致更高的成本。
- 通过优化模型以提高性能，您可以通过使用更少或更少的实例来降低成本，同时保持相同或更好的性能特征。您可以使用SageMaker Neo和 SageMaker 推理自动优化模型。有关更多细节和示例，请参阅使用 Neo 优化模型性能 (p. 2325).

使用最佳实例类型和大小进行实时推理。

- SageMaker Inference 有 70 多种实例类型和大小可用于部署机器学习模型，其中包括Amazon针对机器学习进行了优化的推理和 Graviton 芯片组。为模型选择合适的实例有助于确保以最低的成本为模型拥有最高性能的实例。

- 使用推理，您可以快速比较不同的实例以了解模型的性能和成本。有了这些结果，您可以选择要部署的实例，并获得最佳投资回报。

通过将多个端点合并为单个端点进行实时推理，提高效率和成本。

- 当您部署多个终端节点时，成本会快速增加，尤其是在终端节点没有充分利用底层实例的情况下。要了解实例是否未充分利用，请检查亚马逊中的利用率指标（CPU、GPU 等） CloudWatch 为了您的实例。如果您有多个这些端点，则可以将这些多个端点上的模型或容器组合到一个端点中。
- 使用多模型终端节点（女士）或多容器终端节点(MCE)，您可以在单个终端节点中部署多个机器学习模型或容器，以便在多个模型或容器之间共享实例，并提高投资回报率。要了解更多信息，请参阅使用亚马逊节省推理成本 SageMaker 多模型终端节点要么使用亚马逊在单个实例上部署多个服务容器 SageMaker 多容器终端节点在AmazonMachine Learning 博客。

设置自动扩展以满足实时和异步推断的工作负载要求。

- 如果没有自动扩展，您需要为峰值流量或风险模型不可用进行预置。除非模型的流量全天保持稳定，否则将有剩余的未使用容量。这导致了低利用率和浪费资源。
- AutoScaling是 out-of-the-box 该功能可监控您的工作负载并动态调整容量，以尽可能低的成本保持稳定且可预测的性能。当工作负载增加时，自动扩展会让更多实例联机。当工作负载减小时，自动扩展会删除不必要的实例，从而帮助您降低计算成本。要了解更多信息，请参阅在 Amazon SageMaker 中配置自动缩放推理终端节点在AmazonMachine Learning 博客。

使用模型注册表注册并部署模型

使用 SageMaker 模型注册表您可以执行以下操作：

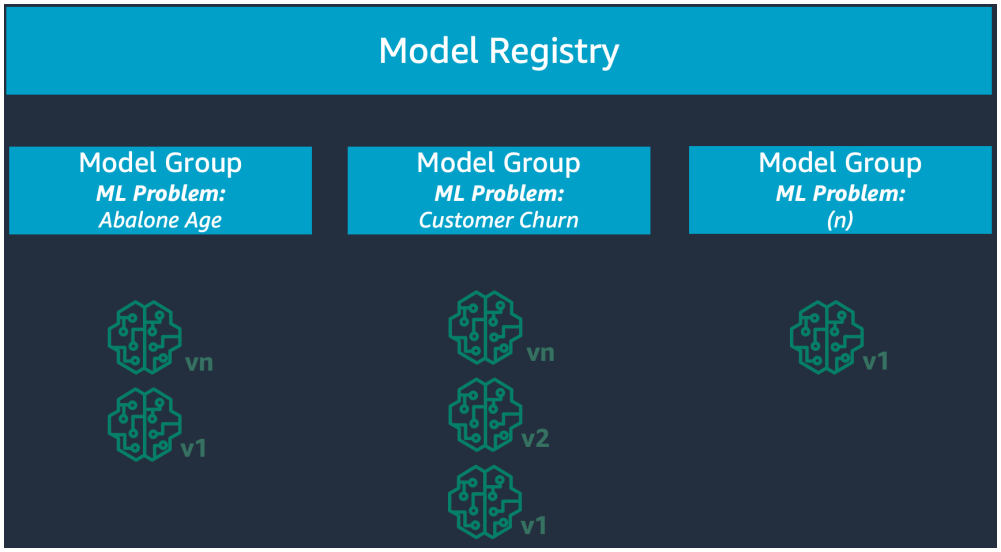
- 用于生产的目录模型。
- 管理模型版本。
- 将元数据（例如训练指标）与模型关联。
- 管理模型的批准状态。
- 将模型部署到生产中。
- 使用 CI/CD 自动化模型部署。

通过创建包含模型不同版本的模型包组对模型进行编目。您可以创建一个模型组来跟踪为解决特定问题而训练的所有模型。然后，您可以注册训练的每个模型，模型注册表将其作为新模型版本添加到模型组中。典型的工作流程可能如下所示：

- 创建模型组。
- 创建训练模型的 ML 管道。有关的信息 SageMaker 管道，请参阅创建和管理 SageMaker 管道 (p. 2478).
- 对于 ML 管道的每次运行，请创建一个模型版本，您在第一步中创建的模型组中注册该模型版本。

模型注册表结构

这些区域有： SageMaker 模型注册表结构为几个模型组，每个组中都有模型包。模型组中的每个模型包对应于训练的模型。每个模型包的版本是一个从 1 开始的数值，随着每个新模型包添加到模型组中而递增。例如，如果将 5 个模型包添加到模型组中，则模型包的版本将为 1、2、3、4 和 5。下图中显示的模型注册表示例包含 3 个模型组，其中每个组包含与特定机器学习问题相关的模型包。



SageMaker 中有两种类型的模型包。一种类型用于AmazonMarketplace，另一个在模型注册表中使用。中使用的模型包AmazonMarketplace 不是可版本的实体，也不与模型注册表中的模型组关联。有关中使用的模型包的更多信息Amazon请参阅 Marketplace在中购买和出售 Amazon SageMaker 算法和模型Amazon Web Services Marketplace (p. 2671).

模型注册表中使用的模型包是版本化的，并且必须与模型组关联。此模型包类型的 ARN 具有以下结构：`'arn:aws:sagemaker:region:account:model-group/version'`

以下主题向您介绍如何使用模型注册表。

主题

- 创建模型组 (p. 2254)
- 注册模型版本 (p. 2257)
- 查看模型组和版本 (p. 2264)
- 查看模型版本的详细信息 (p. 2265)
- 更新模型的批准状态 (p. 2268)
- 部署注册表中的模型 (p. 2271)
- 查看模型的部署历史记录 (p. 2275)
- 亚马逊 SageMaker 模型注册表 (p. 2278)

创建模型组

模型组包含一组版本化模型。通过使用Amazon SDK for Python (Boto3)或 Amazon SageMaker Studio。

创建模型 Package 组 (Boto3)

要使用 Boto3 创建模型组，请调用`create_model_package_group`方法并指定名称和描述作为参数。以下示例显示了如何创建模型组。来自的响应`create_model_package_group`调用是新模型包组的 Amazon 资源名称 (ARN)。

首先，导入所需的软件包并设置 SageMaker Boto3 客户端。

```
import time
import os
from sagemaker import get_execution_role, session
import boto3

region = boto3.Session().region_name

role = get_execution_role()

sm_client = boto3.client('sagemaker', region_name=region)
```

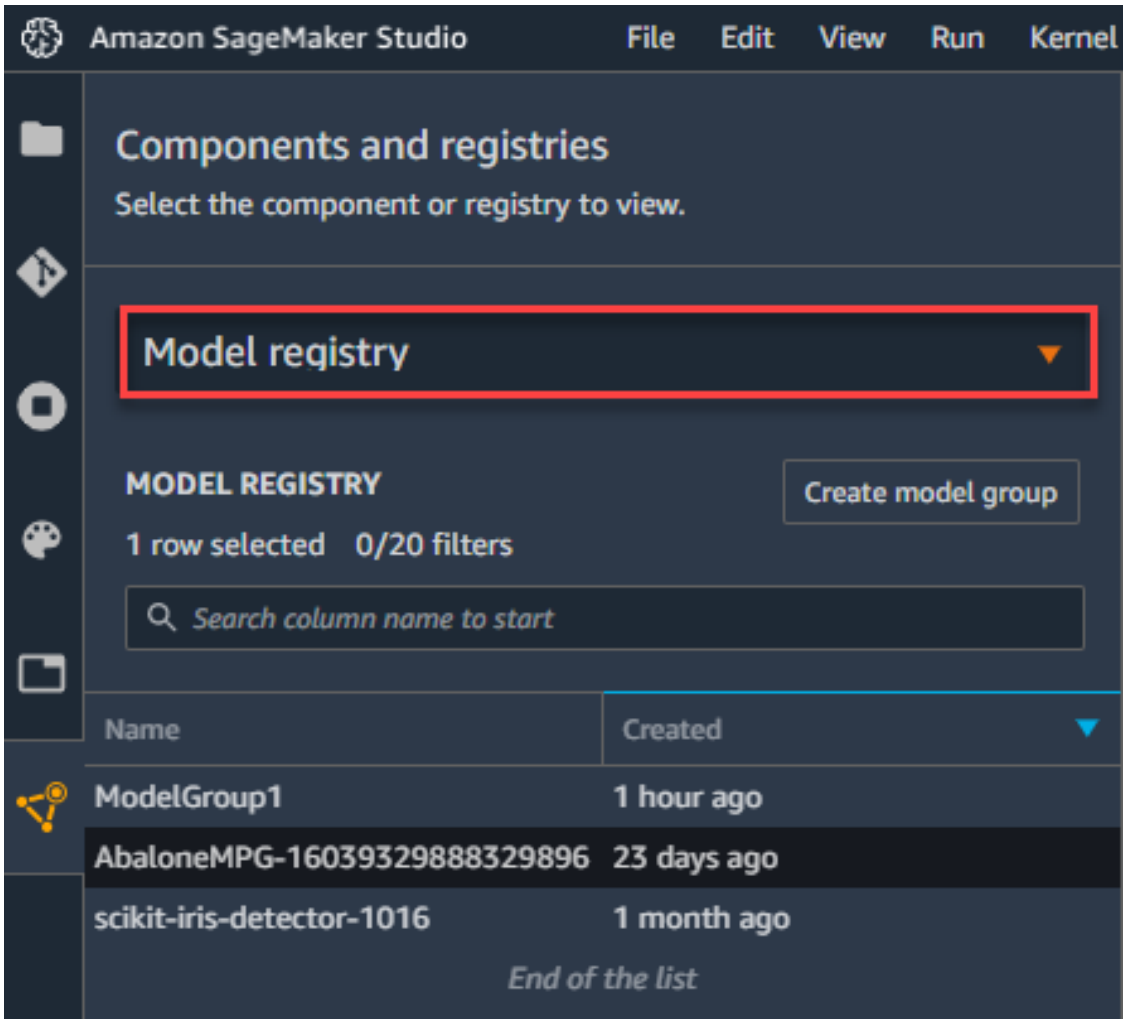
现在创建模型组。

```
import time
model_package_group_name = "scikit-iris-detector-" + str(round(time.time()))
model_package_group_input_dict = {
 "ModelPackageGroupName" : model_package_group_name,
 "ModelPackageGroupDescription" : "Sample model package group"
}

create_model_package_group_response =
 sm_client.create_model_package_group(**model_package_group_input_dict)
print('ModelPackageGroup Arn :
 {}'.format(create_model_package_group_response['ModelPackageGroupArn']))
```

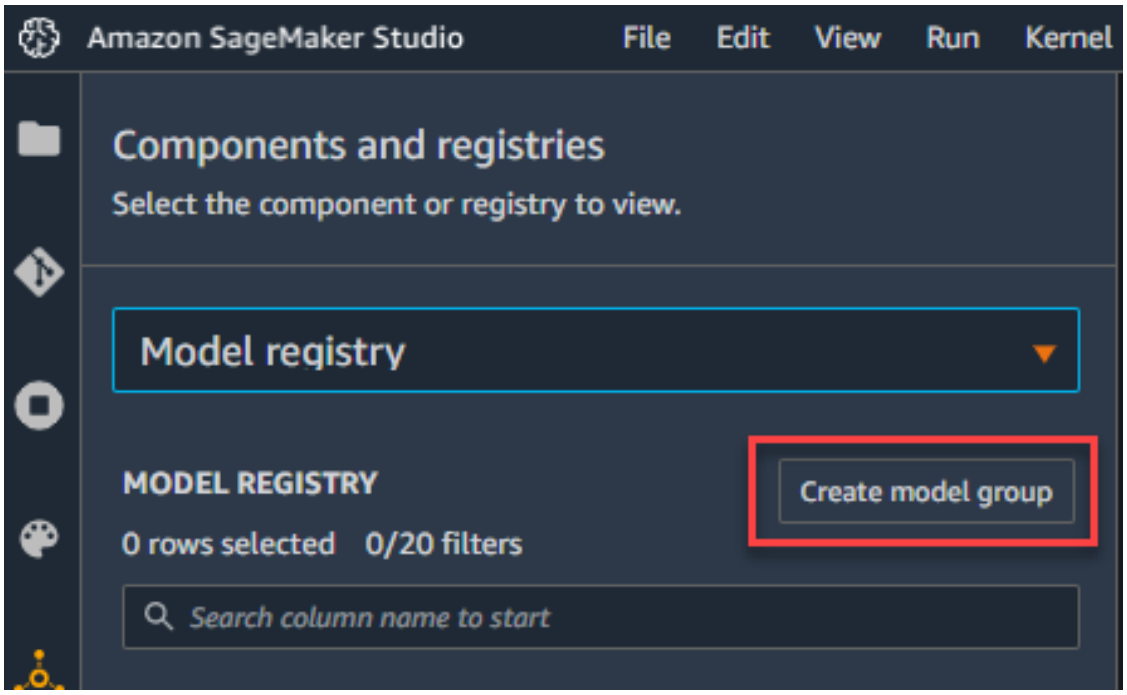
创建模型 Package 组（Amazon） SageMaker) Studio

在 Amazon 中创建模型组 SageMaker Studio，请完成以下步骤。

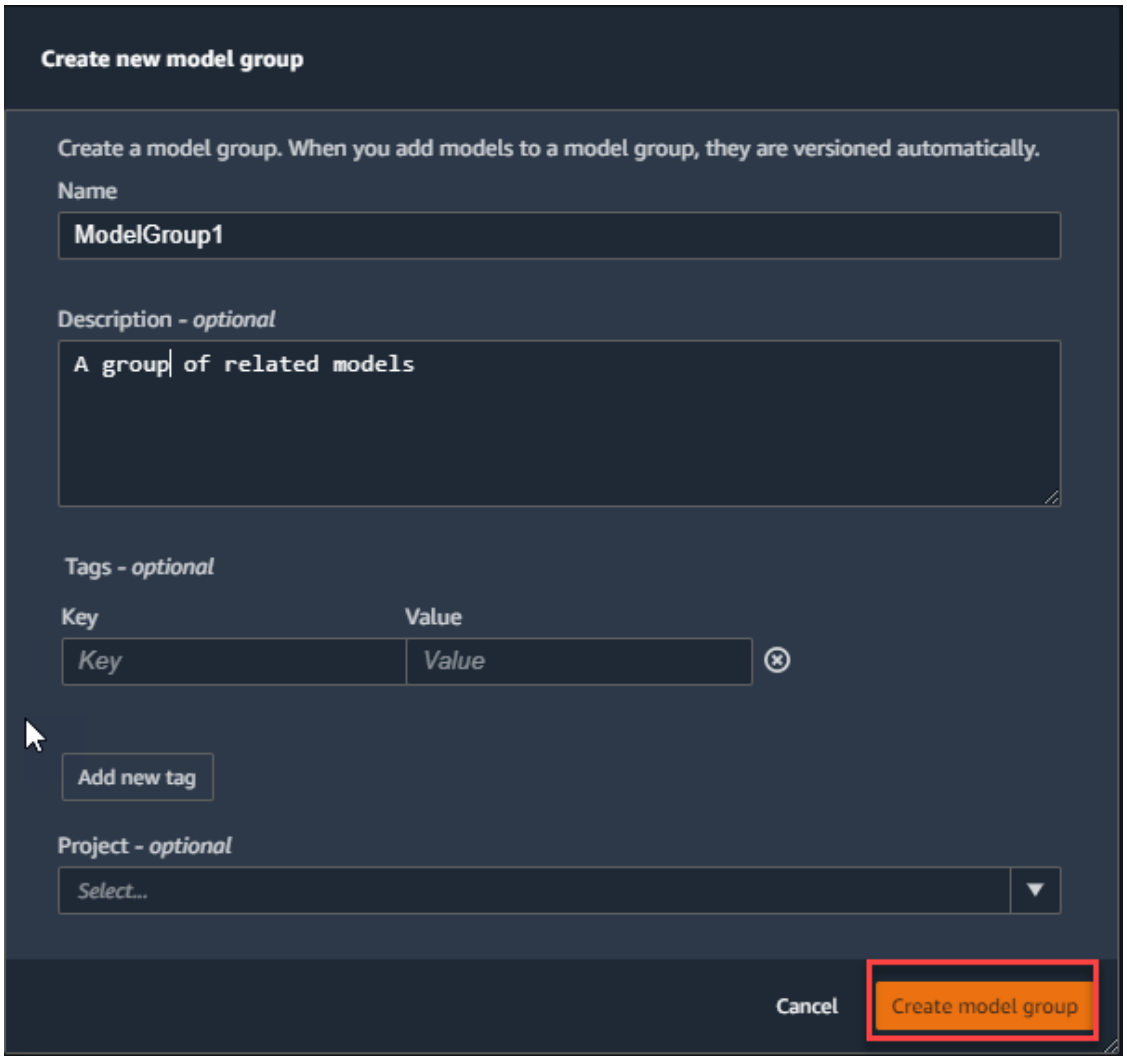
1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
2. 在左侧导航窗格中，选择组件和注册表图标 ( )。
3. 选择模型注册表.


Amazon SageMaker Studio
File
Edit
View
Run
Kernel
Components and registries
Select the component or registry to view.
Model registry
MODEL REGISTRY
Create model group
1 row selected
0/20 filters
Search column name to start
Name
Created
ModelGroup1
1 hour ago
AbaloneMPG-16039329888329896
23 days ago
scikit-iris-detector-1016
1 month ago
End of the list

4. 选择创建模型组.



5. 在创建新模型组对话框中，输入以下信息：
   - 在名称字段中返回的子位置类型。
   - （可选）在中输入模型组的描述说明字段中返回的子位置类型。
   - （可选）在中输入要与模型组关联的任何键-值对。标签字段中返回的子位置类型。有关使用标签的信息，请参阅。标记Amazon资源中的Amazon一般参考.
   - （可选）在项目字段中返回的子位置类型。有关项目的信息，请参阅。使用实现自动化 mLOP SageMaker 项目 (p. 2507).
6. 选择创建模型组.



注册模型版本

你可以注册亚马逊 SageMaker 通过创建模型版本来指定模型组所属的模型组。模型版本必须同时包含模型工件（模型的训练权重）和模型的推理代码。

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的推理管道是 SageMaker 模型由两到十五个容器的线性序列组成，用于处理推理请求。您可以通过指定容器和关联的环境变量来注册推理管道。有关推理管道的更多信息，请参阅。主机模型以及预处理逻辑作为一个端点后面的串行推理管道 (p. 2131).

您可以通过指定容器和关联的环境变量，使用推理管道注册模型。要创建带推理管道的模型版本，请使用 Amazon SDK for Python (Boto3)或者通过在 SageMaker 使用模型构建管道，请使用以下步骤。

主题

- 注册模型版本（SageMaker 管道）(p. 2258)
- 注册模型版本 (Boto3) (p. 2258)
- 从另一账户注册模型版本 (p. 2258)

注册模型版本（SageMaker 管道）

使用注册模型版本 SageMaker 模型构建管道，创建`RegisterModel`走进你的管道。有关创建`RegisterModel`作为管道的一部分，请参阅步骤 8：定义 RegisterModel 创建模型 Package 的步骤 (p. 2488).

注册模型版本 (Boto3)

要使用 Boto3 注册模型版本，请调用`create_model_package`方法。

首先，设置参数字典以传递给`create_model_package`方法。

```
# Specify the model source
model_url = "s3://your-bucket-name/model.tar.gz"

modelpackage_inference_specification =  {
    "InferenceSpecification": {
      "Containers": [
         {
            "Image": '257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-
xgboost:1.2-1',
     "ModelDataUrl": model_url
         }
      ],
      "SupportedContentTypes": [ "text/csv" ],
      "SupportedResponseMIMETypes": [ "text/csv" ],
   }
 }

# Alternatively, you can specify the model source like this:
# modelpackage_inference_specification["InferenceSpecification"]["Containers"][0]
["ModelDataUrl"]=model_url

create_model_package_input_dict = {
    "ModelPackageGroupName" : model_package_group_name,
    "ModelPackageDescription" : "Model to detect 3 different types of irises (Setosa,
 Versicolour, and Virginica)",
    "ModelApprovalStatus" : "PendingManualApproval"
}
create_model_package_input_dict.update(modelpackage_inference_specification)
```

然后，调用`create_model_package`方法，传入刚刚设置的参数字典。

```
create_model_package_response =
 sm_client.create_model_package(**create_model_package_input_dict)
model_package_arn = create_model_package_response["ModelPackageArn"]
print('ModelPackage Version ARN : {}'.format(model_package_arn))
```

从另一账户注册模型版本

将模型版本注册到由不同的模型包组创建的模型包组Amazon账户，你必须添加跨账户Amazon Identity and Access Management资源策略以启用该帐户。例如，一个Amazon组织中的帐户负责培训模型，另一个帐户负责管理、部署和更新模型。您可以创建 IAM 资源策略并将策略应用于要为此案例授予访问权限的特定账户资源。有关跨账户资源策略的更多信息，请参阅Amazon，请参阅跨账户策略评估逻辑中的Amazon Identity and Access Management用户指南.

> **Note**
>
> 您还必须使用 KMS 密钥加密输出数据配置跨账户模型部署期间的操作。

要在 SageMaker 中启用跨账户模型注册表，您必须为包含模型版本的模型组提供跨账户资源策略。以下示例为模型包组创建跨账户策略并将这些策略应用于该特定资源。

必须在源账户中设置以下配置，该账户将模型跨账户注册到模型包组中。在此示例中，源账户是模型培训账户，它将训练模型交叉账户，然后将模型交叉账户注册到模型注册表账户的模型注册表中。

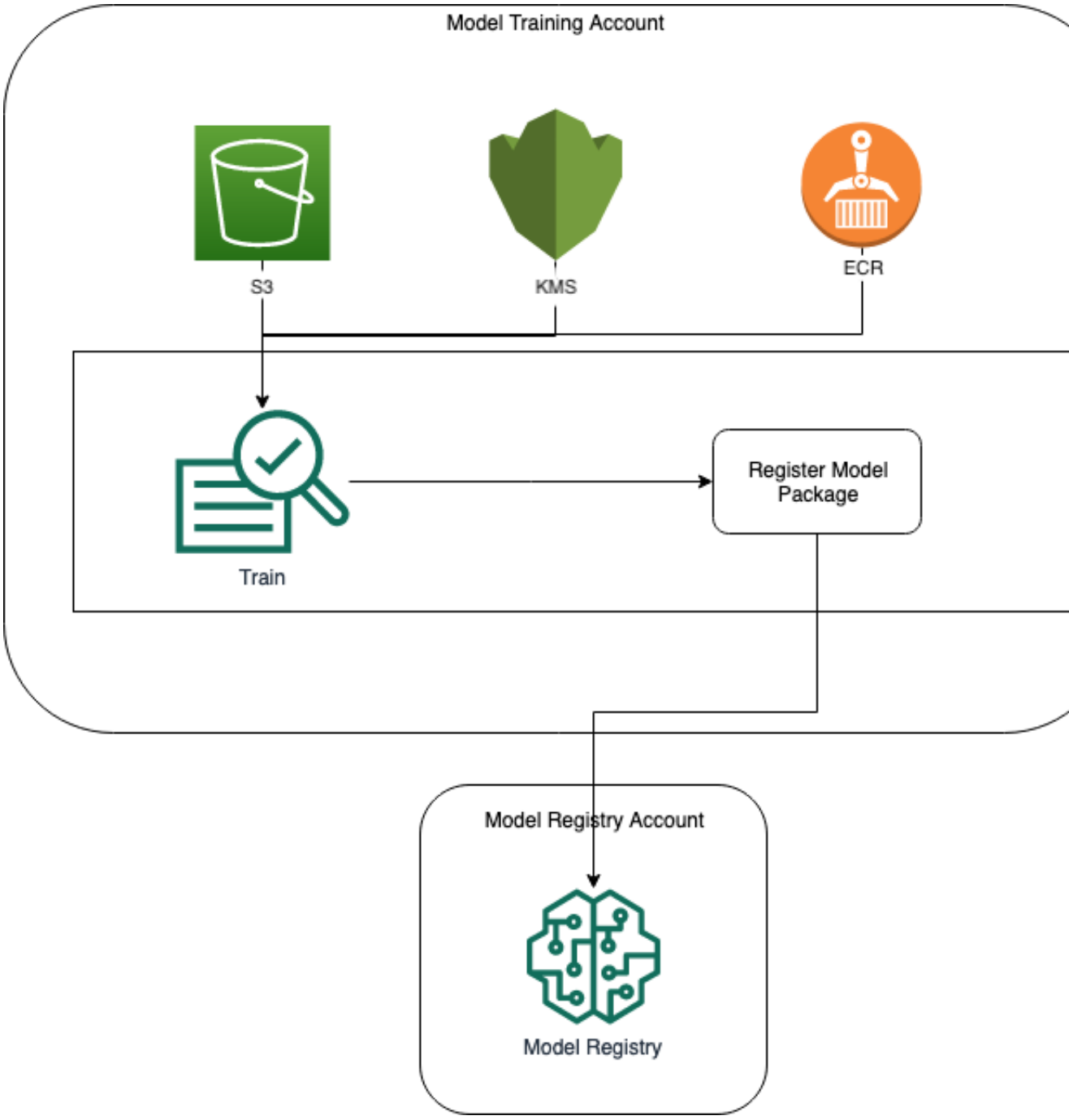
该示例假定您之前定义了以下变量：

- `sm_client`-A SageMaker Boto3 客户端。
- `model_package_group_name`-您要授予访问权限的模型组。
- `model_package_group_arn`-您希望授予跨账户访问权限的模型组 ARN。
- `bucket`-存储模型训练构件的 S3 存储桶。

为了能够部署在其他账户中创建的模型，用户必须拥有对其进行访问的角色。 SageMaker 操作，例如具有`AmazonSageMakerFullAccess`托管策略。有关 SageMaker 托管策略的信息，请参阅AmazonAmazon SageMaker 的托管式策略 (p. 2754)。

必需的 IAM 资源策略

下图描述了允许跨账户模型注册所需的策略。如图所示，这些策略需要在模型训练期间处于活动状态，才能将模型正确注册到模型注册表账户中。



亚马逊 ECR、Amazon S3 和Amazon KMS以下代码示例演示了策略。

示例 Amazon ECR 策略

```
{
  'Version': '2012-10-17',
  'Statement': [
    {
      'Sid': 'AddPerm',
      'Effect': 'Allow',
```

```
      'Principal': {
        'AWS': 'arn:aws:iam::{model_registry_account}:root'
      },
      'Action': [
        'ecr:BatchGetImage',
        'ecr:Describe*'
      ]
    }
  ]
}
```

示例 Amazon S3 策略

```
{
  'Version': '2012-10-17',
  'Statement': [
    {
      'Sid': 'AddPerm',
      'Effect': 'Allow',
      'Principal': {
        'AWS': 'arn:aws:iam::{model_registry_account}:root'
      },
      'Action': [
        's3:GetObject',
        's3:GetBucketAcl',
        's3:GetObjectAcl'
      ],
      'Resource': 'arn:aws:s3:::{bucket}/*'
    }
  ]
}
```

示例Amazon KMS政策

```
{
  'Version': '2012-10-17',
  'Statement': [
    {
      'Sid': 'AddPerm',
      'Effect': 'Allow',
      'Principal': {
        'AWS': 'arn:aws:iam::{model_registry_account}:root'
      },
      'Action': [
        'kms:Decrypt',
        'kms:GenerateDataKey*'
      ],
      'Resource': '*'
    }
  ]
}
```

将资源策略应用于账户

以下策略配置适用上一节中讨论的策略，必须放入模型培训账户中。

```
import json

# The model registry account id of the model package group
model_registry_account = '111111111111'

# The model training account id where training happens
```

```
model_training_account = '222222222222'

# 1. Create a policy for access to the ECR repository
# in the model training account for the model registry account model package group
ecr_repository_policy = {'Version': '2012-10-17',
    'Statement': [{'Sid': 'AddPerm',
        'Effect': 'Allow',
        'Principal': {
          'AWS': f'arn:aws:iam::{model_registry_account}:root'
        },
        'Action': [
          'ecr:BatchGetImage',
          'ecr:Describe*'
        ]
    }]
}

# Convert the ECR policy from JSON dict to string
ecr_repository_policy = json.dumps(ecr_repository_policy)

# Set the new ECR policy
ecr = boto3.client('ecr')
response = ecr.set_repository_policy(
    registryId = model_training_account,
    repositoryName = 'decision-trees-sample',
    policyText = ecr_repository_policy
)

# 2. Create a policy in the model training account for access to the S3 bucket
# where the model is present in the model registry account model package group
bucket_policy = {'Version': '2012-10-17',
    'Statement': [{'Sid': 'AddPerm',
        'Effect': 'Allow',
        'Principal': {'AWS': f'arn:aws:iam::{model_registry_account}:root'
        },
        'Action': [
          's3:GetObject',
          's3:GetBucketAcl',
          's3:GetObjectAcl'
        ],
        'Resource': 'arn:aws:s3:::{bucket}/*'
    }]
}

# Convert the S3 policy from JSON dict to string
bucket_policy = json.dumps(bucket_policy)

# Set the new bucket policy
s3 = boto3.client('s3')
response = s3.put_bucket_policy(
    Bucket = bucket,
    Policy = bucket_policy)

# 3. Create the KMS grant for the key used during training for encryption
# in the model training account to the model registry account model package group
client = boto3.client('kms')

response = client.create_grant(
    GranteePrincipal=model_registry_account,
    KeyId=kms_key_id
    Operations=[
        'Decrypt',
        'GenerateDataKey',
    ],
)
```

需要将以下配置放在模型包组所在的模型注册表帐户中。

```
# The model registry account id of the model package group
model_registry_account = '111111111111'

# 1. Create policy to allow the model training account to access the ModelPackageGroup
model_package_group_policy = {'Version': '2012-10-17',
    'Statement': [
        {
            'Sid': 'AddPermModelPackageVersion',
            'Effect': 'Allow',
            'Principal': {'AWS': f'arn:aws:iam::{model_training_account}:root'
            'Action': ["sagemaker:CreateModelPackage"],
            'Resource': f'arn:aws:sagemaker:{region}:{model_registry_account}:model-
package/{model_package_group_name}/*'
        }
    ]
}

# Convert the policy from JSON dict to string
model_package_group_policy = json.dumps(model_package_group_policy)

# Set the new policy
response = sm_client.put_model_package_group_policy(
    ModelPackageGroupName = model_package_group_name,
    ResourcePolicy = model_package_group_policy)
```

最后，使用create_model_package来自模型培训账户的操作，以便在交叉账户中注册模型包。

```
# Specify the model source
model_url = "s3://{bucket}/model.tar.gz"

#Set up the parameter dictionary to pass to the create_model_package method
modelpackage_inference_specification =  {
    "InferenceSpecification": {
        "Containers": [
            {
                "Image": f'{model_training_account}.dkr.ecr.us-east-2.amazonaws.com/
decision-trees-sample:latest',
                "ModelDataUrl": model_url
            }
        ],
        "SupportedContentTypes": [ "text/csv" ],
        "SupportedResponseMIMETypes": [ "text/csv" ],
    }
}

# Alternatively, you can specify the model source like this:
# modelpackage_inference_specification["InferenceSpecification"]["Containers"][0]
["ModelDataUrl"]=model_url

create_model_package_input_dict = {
    "ModelPackageGroupName" : model_package_group_arn,
    "ModelPackageDescription" : "Model to detect 3 different types of irises (Setosa,
 Versicolour, and Virginica)",
    "ModelApprovalStatus" : "PendingManualApproval"
}
create_model_package_input_dict.update(modelpackage_inference_specification)

# Create the model package in the model registry account
```

```
create_model_package_response =
 sm_client.create_model_package(**create_model_package_input_dict)
model_package_arn = create_model_package_response["ModelPackageArn"]
print('ModelPackage Version ARN : {}'.format(model_package_arn))
```

查看模型组和版本

模型组和版本可帮助您组织模型。您可以查看模型组中模型版本的列表。

查看组中的模型版本列表

您可以查看与模型组关联的所有模型版本。如果模型组代表您为解决特定机器学习问题而训练的所有模型，则可以查看所有这些相关模型。

查看组中的模型版本列表 (Boto3)

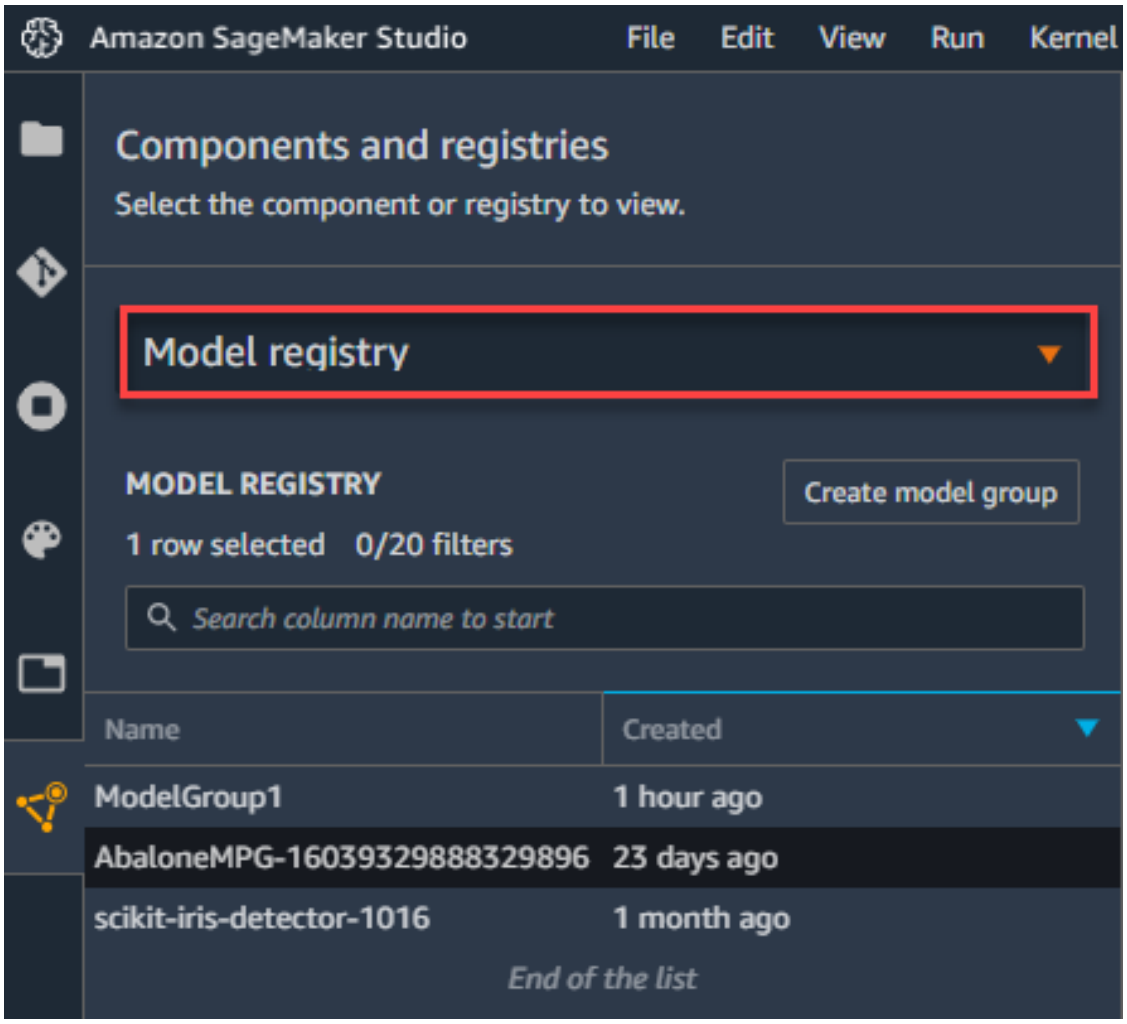
要使用 Boto3 查看与模型组关联的模型版本，请调用`list_model_packages`方法，然后将模型组的名称作为`ModelPackageGroupName`参数。以下代码列出了与您在中创建的模型组相关联的模型版本：创建模型 Package 组 (Boto3) (p. 2254).

```
sm_client.list_model_packages(ModelPackageGroupName=model_package_group_name)
```

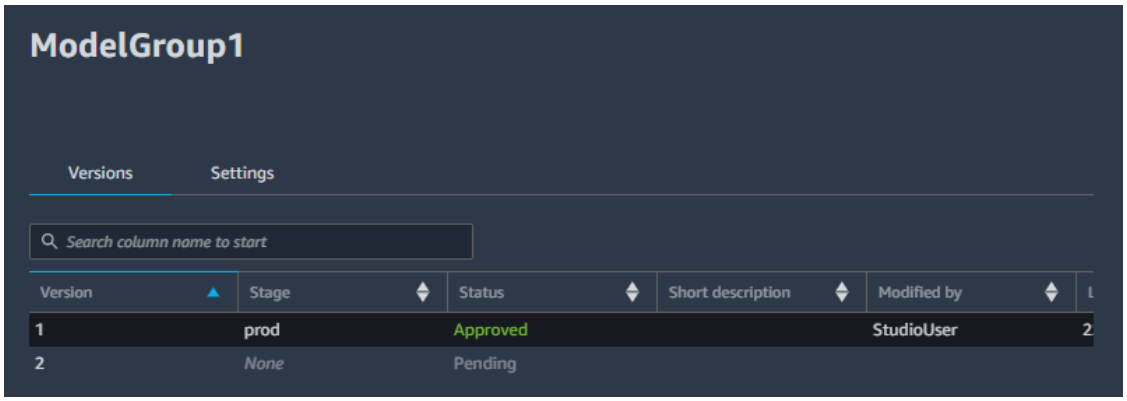
查看组中的模型版本列表（亚马逊） SageMaker) Studio

要查看模型组中模型版本的列表，请完成以下步骤。

1. 登录亚马逊 SageMaker Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
2. 在左侧导航窗格中，选择组件和注册表图标 ( )。
3. 选择模型注册表.



4. 从模型组列表中，选择要查看的模型组。
5. 此时将出现一个新选项卡，其中包含模型组中模型版本的列表，如下图所示。



查看模型版本的详细信息

您可以通过使用Amazon SDK for Python (Boto3)或者通过使用亚马逊 SageMaker Studio。

查看模型版本的详细信息（Boto3）

要使用 Boto3 查看模型版本的详细信息，请完成以下步骤。

1. 调用`list_model_packages`方法来查看模型组中的模型版本。

```
sm_client.list_model_packages(ModelPackageGroupName="ModelGroup1")
```

回应是模型包摘要列表。您可以从此列表中获得模型版本的 Amazon 资源名称 (ARN)。

```
{'ModelPackageSummaryList': [{'ModelPackageGroupName': 'AbaloneMPG-16039329888329896',
   'ModelPackageVersion': 1,
   'ModelPackageArn': 'arn:aws:sagemaker:us-east-2:123456789012:model-package/
ModelGroup1/1',
   'ModelPackageDescription': 'TestMe',
   'CreationTime': datetime.datetime(2020, 10, 29, 1, 27, 46, 46000, tzinfo=tzlocal()),
   'ModelPackageStatus': 'Completed',
   'ModelApprovalStatus': 'Approved'}],
 'ResponseMetadata': {'RequestId': '12345678-abcd-1234-abcd-aabbccddeeff',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '12345678-abcd-1234-abcd-aabbccddeeff',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '349',
   'date': 'Mon, 23 Nov 2020 04:56:50 GMT'},
  'RetryAttempts': 0}}
```

2. Call`describe_model_package`查看模型版本的详细信息。你传入你在调用输出中获得的模型版本的 ARN`list_model_packages`.

```
sm_client.describe_model_package(ModelPackageName="arn:aws:sagemaker:us-
east-2:123456789012:model-package/ModelGroup1/1")
```

此调用的输出是一个 JSON 对象，其中包含模型版本详细信息。

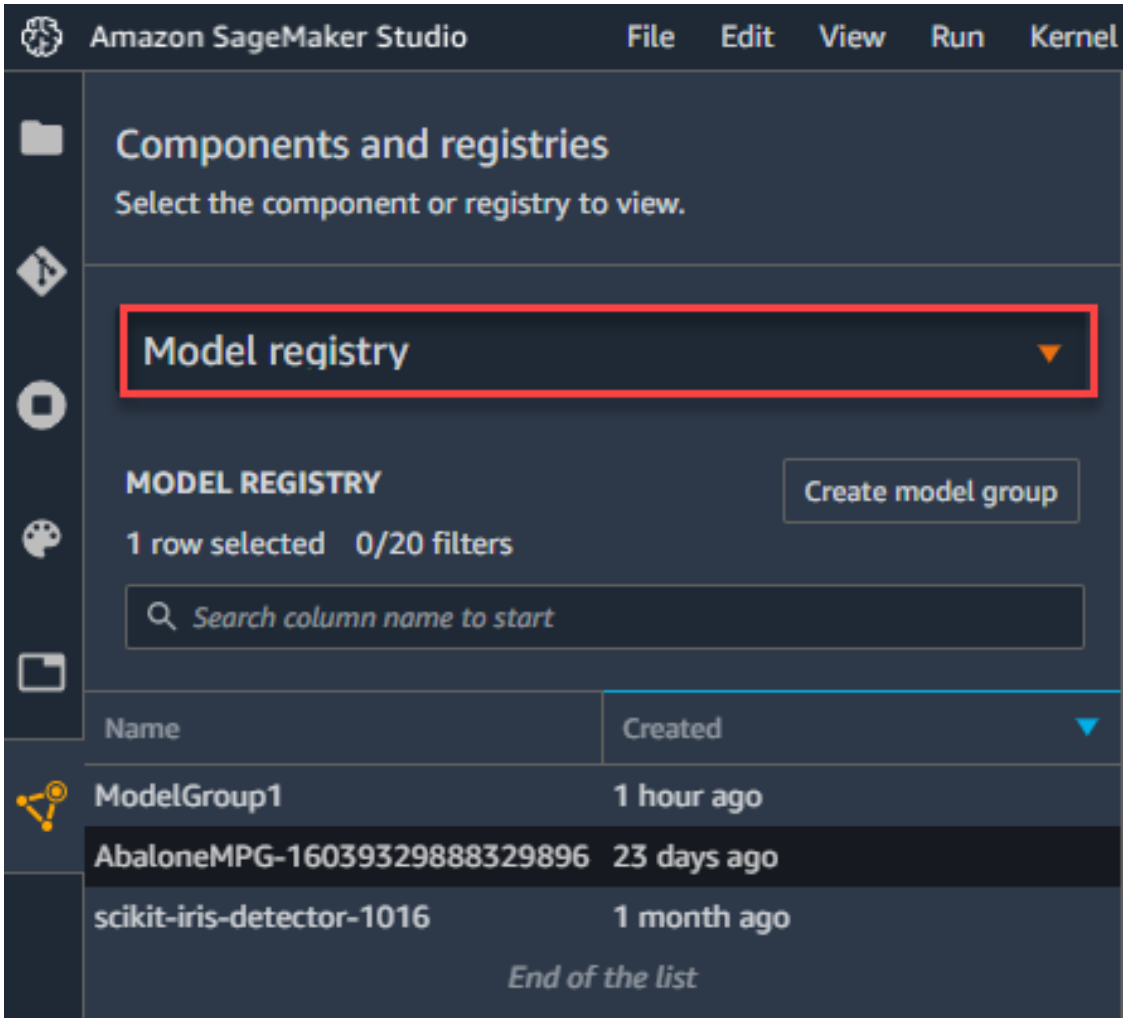
```
{'ModelPackageGroupName': 'ModelGroup1',
 'ModelPackageVersion': 1,
 'ModelPackageArn': 'arn:aws:sagemaker:us-east-2:123456789012:model-package/
ModelGroup/1',
 'ModelPackageDescription': 'Test Model',
 'CreationTime': datetime.datetime(2020, 10, 29, 1, 27, 46, 46000, tzinfo=tzlocal()),
 'InferenceSpecification': {'Containers': [{'Image': '257758044811.dkr.ecr.us-
east-2.amazonaws.com/sagemaker-xgboost:1.0-1-cpu-py3',
    'ImageDigest':
 'sha256:99fa602cff19aee33297a5926f8497ca7bcd2a391b7d600300204eef803bca66',
    'ModelDataUrl': 's3://sagemaker-us-east-2-123456789012/ModelGroup1/
pipelines-0gdonccek7o9-AbaloneTrain-stmiylhtIR/output/model.tar.gz'}],
  'SupportedTransformInstanceTypes': ['ml.m5.xlarge'],
  'SupportedRealtimeInferenceInstanceTypes': ['ml.t2.medium', 'ml.m5.xlarge'],
  'SupportedContentTypes': ['text/csv'],
  'SupportedResponseMIMETypes': ['text/csv']},
 'ModelPackageStatus': 'Completed',
 'ModelPackageStatusDetails': {'ValidationStatuses': [],
  'ImageScanStatuses': []},
 'CertifyForMarketplace': False,
 'ModelApprovalStatus': 'PendingManualApproval',
 'LastModifiedTime': datetime.datetime(2020, 10, 29, 1, 28, 0, 438000,
 tzinfo=tzlocal()),
 'ResponseMetadata': {'RequestId': '12345678-abcd-1234-abcd-aabbccddeeff',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '212345678-abcd-1234-abcd-aabbccddeeff',
   'content-type': 'application/x-amz-json-1.1',
```

```
  'content-length': '1038',
  'date': 'Mon, 23 Nov 2020 04:59:38 GMT'},
 'RetryAttempts': 0}}
```

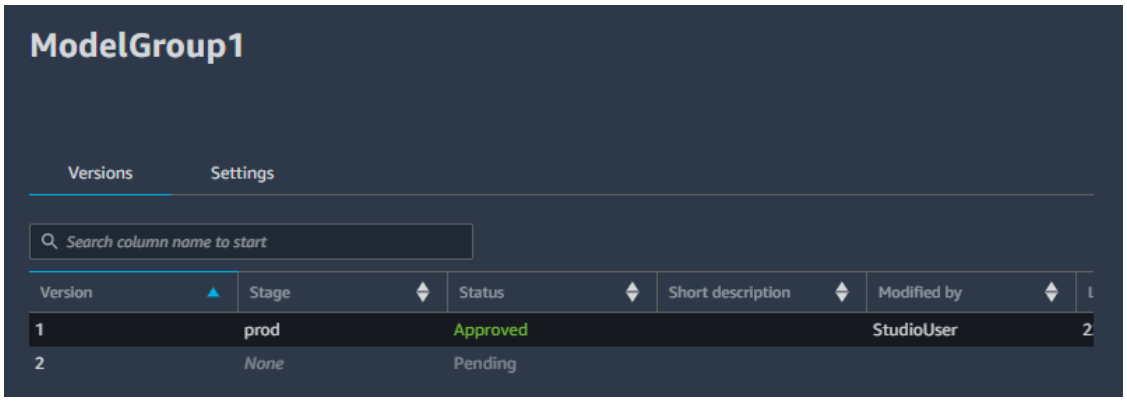
查看模型版本的详细信息（亚马逊） SageMaker）Studio

查看 Amazon 中模型版本的详细信息 SageMaker Studio，请完成以下步骤。

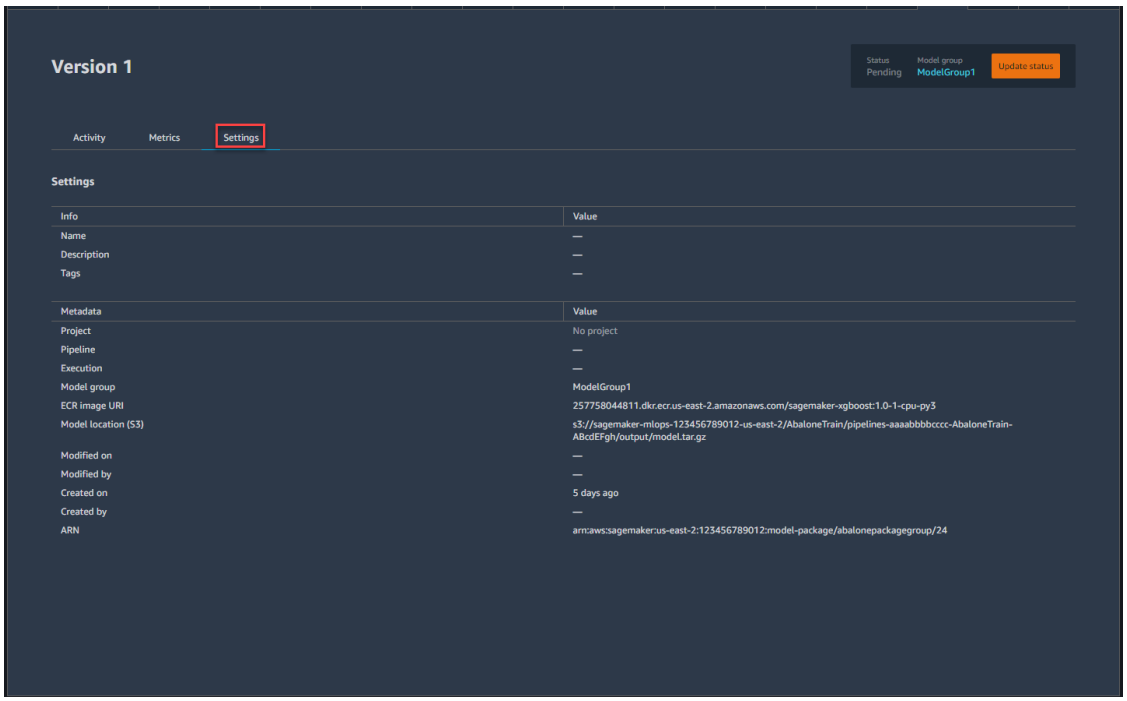
1. 登录亚马逊 SageMaker Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
2. 在左侧导航窗格中，选择组件和注册表图标 ( )。
3. 选择模型注册表.



4. 在模型组列表中，双击要查看的模型组。
5. 此时将出现一个新选项卡，其中包含模型组中模型版本的列表，如下图所示。



6. 在模型版本列表中，双击要查看其详细信息的模型版本。
7. 在打开的模型版本选项卡上，选择以下选项之一以查看有关模型版本的详细信息：
   - 活动：显示模型版本的事件，例如批准状态更新。
   - 指标：显示模型的质量指标。要显示指标，必须使用以下方法为模型启用数据捕获 SageMaker 模型监控器。有关捕获数据的信息，请参阅。捕获数据 (p. 2186).
   - 设置：显示信息，例如与模型版本关联的项目、生成模型的管道、模型组以及模型在 Amazon S3 中的位置。



更新模型的批准状态

创建模型版本后，通常需要在将其部署到生产终端节点之前对其性能进行评估。如果它符合您的要求，您可以将模型版本的批准状态更新为`Approved`. 将状态设置为`Approved`可以为模型启动 CI/CD 部署。如果模型版本不符合您的要求，则可以将批准状态更新为`Rejected`.

您可以在注册模型版本后手动更新该版本的批准状态，也可以在创建 SageMaker 管道时创建条件步骤来评估模型。有关在 SageMaker 管道，请参阅管道步骤 (p. 2450).

当你使用其中一个 SageMaker 如果项目模板和模型版本的批准状态发生变化，则会发生以下操作。仅显示有效的过渡。

- `PendingManualApproval`到`Approved`— 为批准的模型版本启动 CI/CD 部署
- `PendingManualApproval`到`Rejected`— 没有行动
- `Rejected`到`Approved`— 为批准的模型版本启动 CI/CD 部署
- `Approved`到`Rejected`— 启动 CI/CD 以部署最新的模型版本`Approved`状态

您可以使用Amazon SDK for Python (Boto3)或者通过使用亚马逊 SageMaker Studio。您还可以更新模型版本的批准状态，作为条件步骤的一部分 SageMaker 管道。有关使用模型批准步骤的信息 SageMaker 管道，请参阅SageMaker 管道概述 (p. 2446).

更新模型的批准状态 (Boto3)

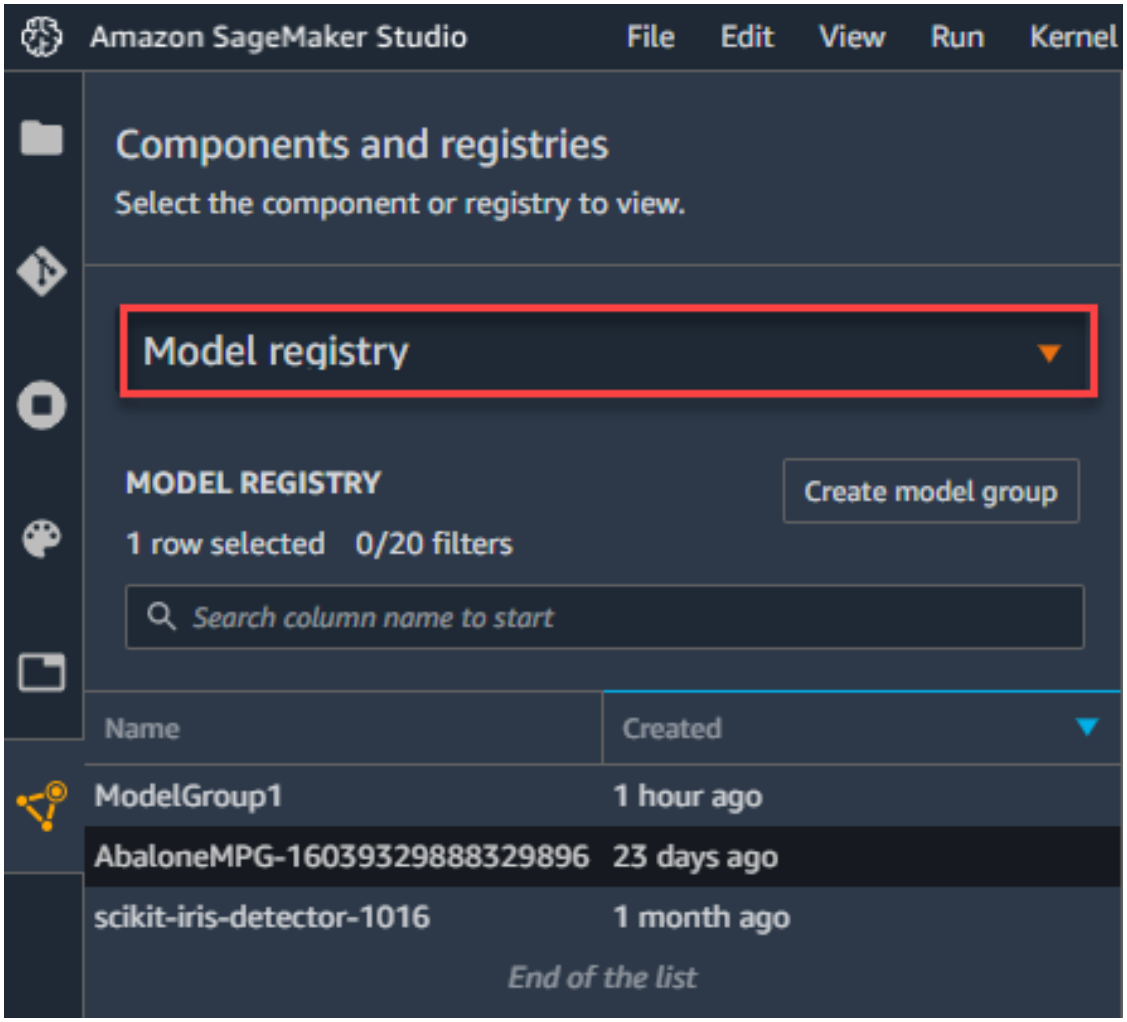
在中创建模型版本时注册模型版本 (p. 2257)，你设置`ModelApprovalStatus`到`PendingManualApproval`. 您可以通过调用来更新模型的批准状态`update_model_package`. 请注意，您可以通过编写代码来自动执行此过程，例如，根据对模型性能的某种评估结果设置模型的批准状态。您还可以在管道中创建一个步骤，在批准新模型版本时自动部署新模型版本。以下代码段演示如何将批准状态手动更改为。`Approved`.

```
model_package_update_input_dict = {
    "ModelPackageArn" : model_package_arn,
    "ModelApprovalStatus" : "Approved"
}
model_package_update_response =
 sm_client.update_model_package(**model_package_update_input_dict)
```

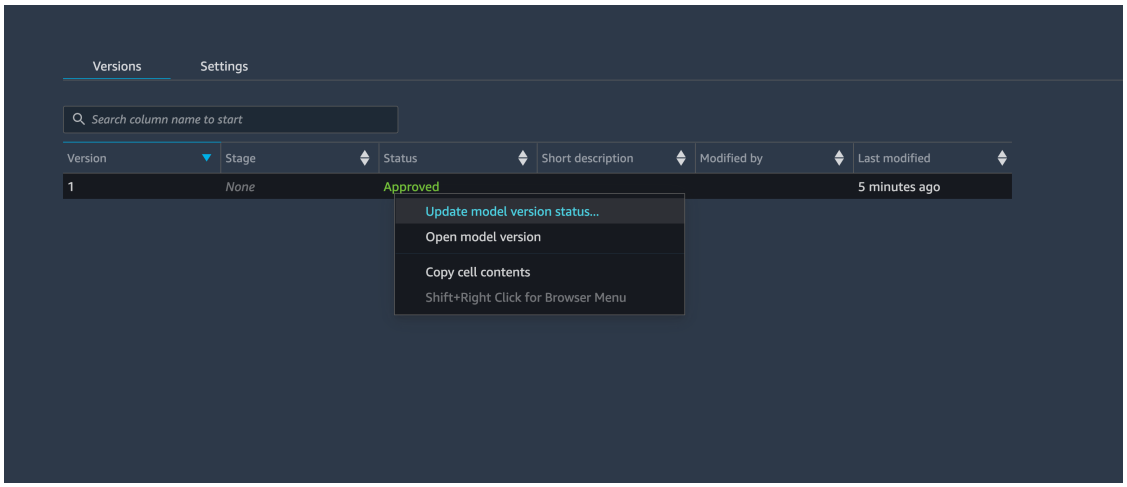
更新模特的批准状态（亚马逊） SageMaker ）Studio

以下过程演示如何从中手动更改批准状态：`Approved`到`Rejected`.

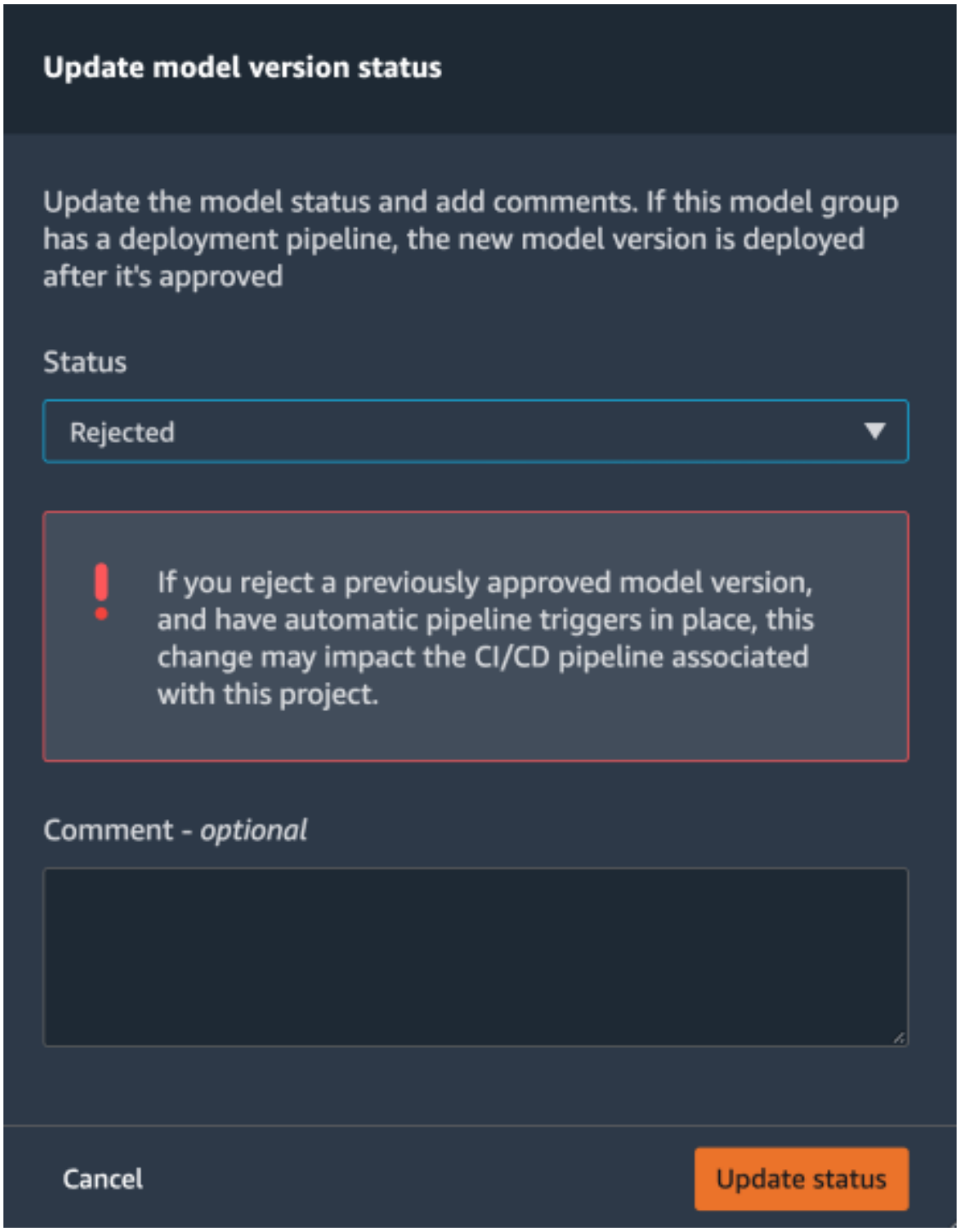
1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
2. 在左侧导航窗格中，选择组件和注册表图标 ( )。
3. 选择模型注册表.



4. 从模型组列表中，选择要查看的模型组。此时将打开一个新选项卡，其中显示组中的模型版本的列表。
5. 在模型版本列表中，右键单击要更新的模型版本，然后选择更新模型版本状态.



6. 在更新模型版本状态对话框，状态选择已拒绝，然后选择更新状态.



部署注册表中的模型

注册模型版本并批准其进行部署后，请将其部署到 SageMaker 终端节点以进行实时推理。

当您创建 mLOPS 项目并选择包含模型部署的 mLOPS 项目模板时，模型注册表中的已批准的模型版本将自动部署到生产环境中。有关使用的信息 SageMaker mLOP 项目，请参阅使用实现自动化 mLOP SageMaker 项目 (p. 2507).

您还可以启用Amazon账户来部署通过添加跨账户资源策略在其他账户中创建的模型版本。例如，组织中的一个团队可能负责培训模型，另一个团队负责部署和更新模型。

主题

- 从注册表中部署模型（SageMaker SDK）(p. 2272)
- 从注册表中部署模型（Boto3）(p. 2272)
- 从其他账户部署模型版本 (p. 2273)

从注册表中部署模型（SageMaker SDK）

使用部署模型版本亚马逊 SageMaker Python 开发工具包使用以下代码片段：

```
from sagemaker import ModelPackage
from time import gmtime, strftime

model_package_arn = 'arn:aws:sagemaker:us-east-2:12345678901:model-package/modeltest/1'
model = ModelPackage(role=role,
                     model_package_arn=model_package_arn,
                     sagemaker_session=sagemaker_session)
model.deploy(initial_instance_count=1, instance_type='ml.m5.xlarge')
```

从注册表中部署模型（Boto3）

使用部署模型版本Amazon SDK for Python (Boto3)，请完成以下步骤：

1. 通过调用create_model方法。作为模型版本的 Amazon 资源名称 (ARN) 作为Containers对于模型对象。

   以下代码段假设您已经创建了 SageMaker Boto3 客户端sm_client，而且您已经创建了一个模型版本，其中包含已存储在名为的变量中的 ARNmodel_version_arn.

   ```
   model_name = 'DEMO-modelregistry-model-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
   print("Model name : {}".format(model_name))
   container_list = [{'ModelPackageName': model_version_arn}]

   create_model_response = sm_client.create_model(
       ModelName = model_name,
       ExecutionRoleArn = role,
       Containers = container_list
   )
   print("Model arn : {}".format(create_model_response["ModelArn"]))
   ```

2. 通过调用创建终端节点配置create_endpoint_config. 终端节点配置指定用于终端节点的 Amazon EC2 实例的数量和类型。

   ```
   endpoint_config_name = 'DEMO-modelregistry-EndpointConfig-' + strftime("%Y-%m-%d-%H-%M-
   %S", gmtime())
   print(endpoint_config_name)
   create_endpoint_config_response = sm_client.create_endpoint_config(
       EndpointConfigName = endpoint_config_name,
       ProductionVariants=[{
           'InstanceType':'ml.m4.xlarge',
           'InitialVariantWeight':1,
   ```

```
        'InitialInstanceCount':1,
        'ModelName':model_name,
        'VariantName':'AllTraffic'}])
```

3. 通过调用创建终端节点`create_endpoint`.

```
endpoint_name = 'DEMO-modelregistry-endpoint-' + strftime("%Y-%m-%d-%H-%M-%S",
 gmtime())
print("EndpointName={}".format(endpoint_name))

create_endpoint_response = sm_client.create_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=endpoint_config_name)
print(create_endpoint_response['EndpointArn'])
```

从其他账户部署模型版本

你可以允许Amazon账户来部署通过添加跨账户资源策略在其他账户中创建的模型版本。例如，组织中的一个团队可能负责培训模型，另一个团队负责部署和更新模型。创建这些资源策略时，您将该策略应用于要向其授予访问权限的特定资源。有关跨账户资源策略的更多信息，请参阅Amazon，请参阅跨账户策略评估逻辑中的Amazon Identity and Access Management用户指南.

> Note
>
> 您必须使用 KMS 密钥加密输出数据配置跨账户模型部署期间的操作。

要在 SageMaker 中启用跨账户模型部署，您必须为包含要部署的模型版本的模型组、模型组推理映像所在的 Amazon ECR 存储库以及模型版本所在的 Amazon S3 存储桶提供跨账户资源策略存储。

要能够部署在其他账户中创建的模型，您必须具有访问权限的角色 SageMaker 操作，例如具有`AmazonSageMakerFullAccess`托管策略。有关 SageMaker 托管策略的信息，请参阅。AmazonAmazon SageMaker 的托管式策略 (p. 2754).

以下示例为所有这三种资源创建跨账户策略，并将这些策略应用于资源。

该示例假定您之前定义了以下变量：

- `bucket`-存储模型版本的 Amazon S3 存储桶。
- `kms_key_id`-用于加密训练输出的 KMS 密钥。
- `sm_client`-A SageMaker Boto3 客户端。
- `model_package_group_name`-您希望授予跨账户访问权限的模型组。
- `model_package_group_arn`-您希望授予跨账户访问权限的模型组 ARN。

```
import json

# The cross-account id to grant access to
cross_account_id = "123456789012"

# Create the policy for access to the ECR repository
ecr_repository_policy = {
    'Version': '2012-10-17',
    'Statement': [{
        'Sid': 'AddPerm',
        'Effect': 'Allow',
        'Principal': {
            'AWS': f'arn:aws:iam::{cross_account_id}:root'
```

```
        },
        'Action': ['ecr:*']
    }]
}

# Convert the ECR policy from JSON dict to string
ecr_repository_policy = json.dumps(ecr_repository_policy)

# Set the new ECR policy
ecr = boto3.client('ecr')
response = ecr.set_repository_policy(
    registryId = account,
    repositoryName = 'decision-trees-sample',
    policyText = ecr_repository_policy
)

# Create a policy for accessing the S3 bucket
bucket_policy = {
    'Version': '2012-10-17',
    'Statement': [{
        'Sid': 'AddPerm',
        'Effect': 'Allow',
        'Principal': {
            'AWS': f'arn:aws:iam::{cross_account_id}:root'
        },
        'Action': 's3:*',
        'Resource': f'arn:aws:s3:::{bucket}/*'
    }]
}

# Convert the policy from JSON dict to string
bucket_policy = json.dumps(bucket_policy)

# Set the new policy
s3 = boto3.client('s3')
response = s3.put_bucket_policy(
    Bucket = bucket,
    Policy = bucket_policy)

# Create the KMS grant for encryption in the source account to the
# model registry account model package group
client = boto3.client('kms')

response = client.create_grant(
    GranteePrincipal=cross_account_id,
    KeyId=kms_key_id
    Operations=[
        'Decrypt',
        'GenerateDataKey',
    ],
)

# 3. Create a policy for access to the model package group.
model_package_group_policy = {
    'Version': '2012-10-17',
    'Statement': [{
        'Sid': 'AddPermModelPackageGroup',
        'Effect': 'Allow',
        'Principal': {
            'AWS': f'arn:aws:iam::{cross_account_id}:root'
        },
        'Action': ['sagemaker:DescribeModelPackageGroup'],
        'Resource': f'arn:aws:sagemaker:{region}:{account}:model-package-group/
{model_package_group_name}'
    },{
        'Sid': 'AddPermModelPackageVersion',
```

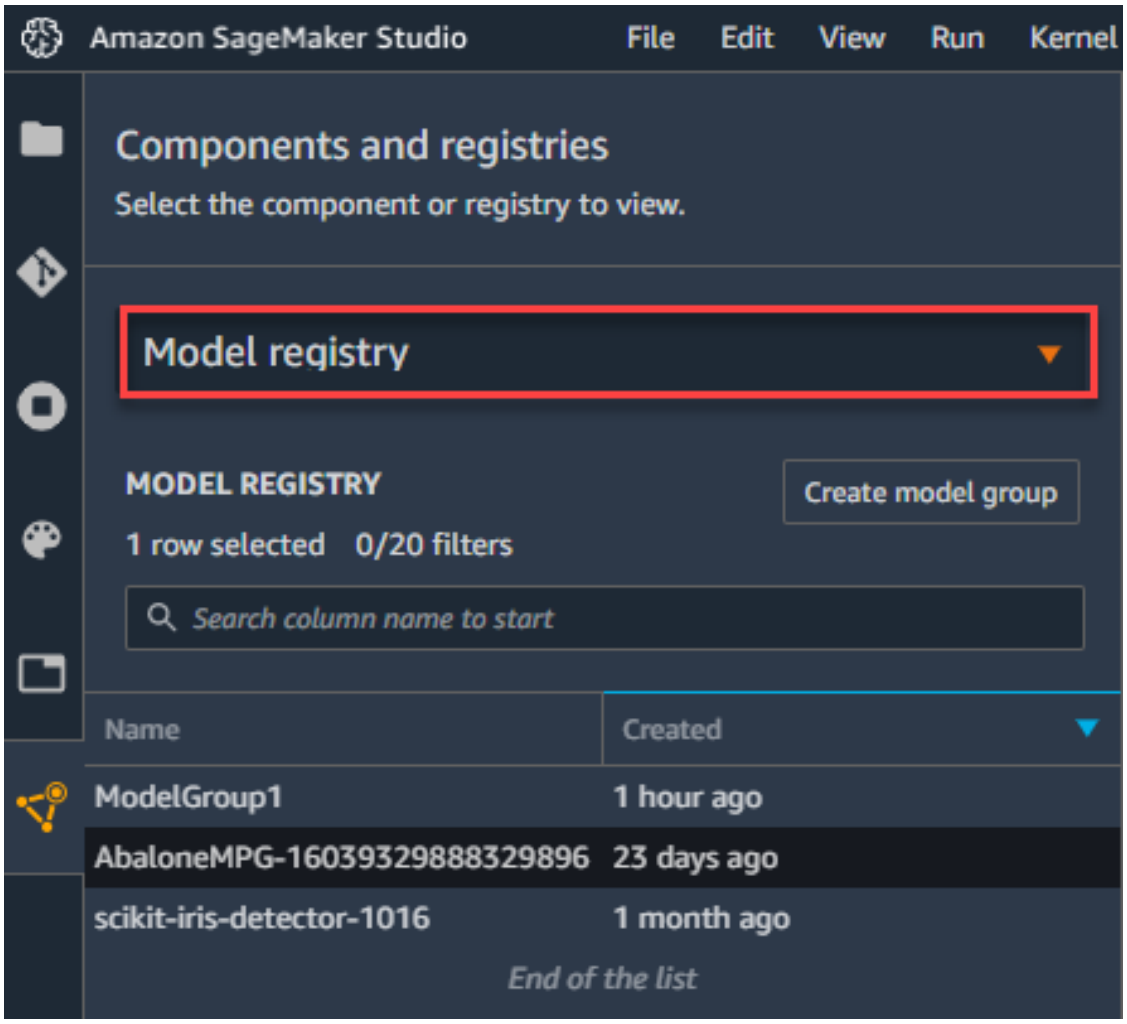
```
        'Effect': 'Allow',
        'Principal': {
            'AWS': f'arn:aws:iam::{cross_account_id}:root'
        },
        'Action': ["sagemaker:DescribeModelPackage",
                   "sagemaker:ListModelPackages",
                   "sagemaker:UpdateModelPackage",
                   "sagemaker:CreateModel"],
        'Resource': f'arn:aws:sagemaker:{region}:{account}:model-package/
{model_package_group_name}/*'
    }]
}

# Convert the policy from JSON dict to string
model_package_group_policy = json.dumps(model_package_group_policy)

# Set the policy to the model package group
response = sm_client.put_model_package_group_policy(
    ModelPackageGroupName = model_package_group_name,
    ResourcePolicy = model_package_group_policy)

print('ModelPackageGroupArn :
 {}'.format(create_model_package_group_response['ModelPackageGroupArn']))
print("First Versioned ModelPackageArn: " + model_package_arn)
print("Second Versioned ModelPackageArn: " + model_package_arn2)

print("Success! You are all set to proceed for cross-account deployment.")
```

查看模型的部署历史记录

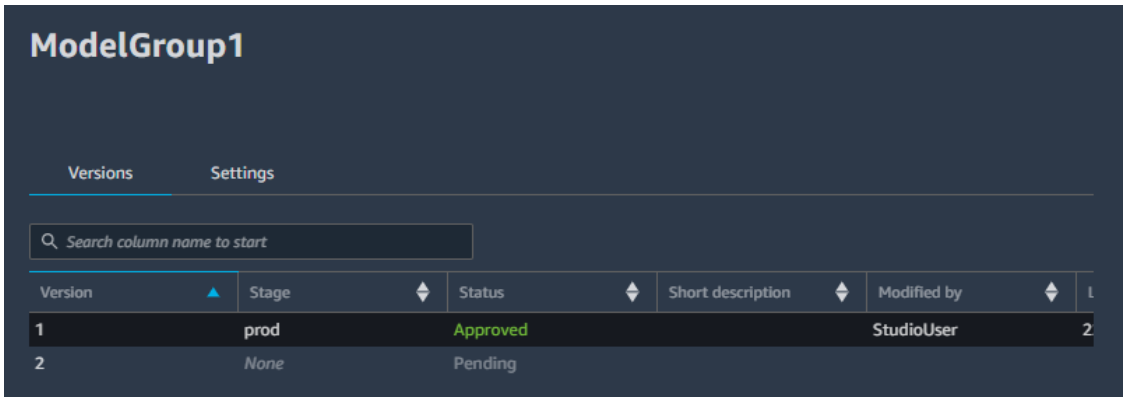
查看模型版本的部署亚马逊 SageMaker Studio 通过打开该模型版本的选项卡。

查看模型版本的部署历史记录

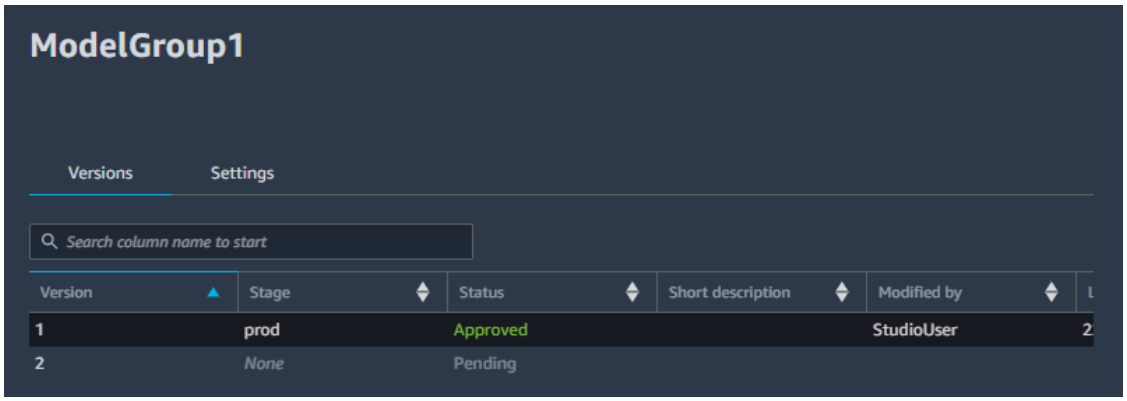
1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
2. 在左侧导航窗格中，选择组件和注册表图标 ( )。
3. 选择模型注册表.



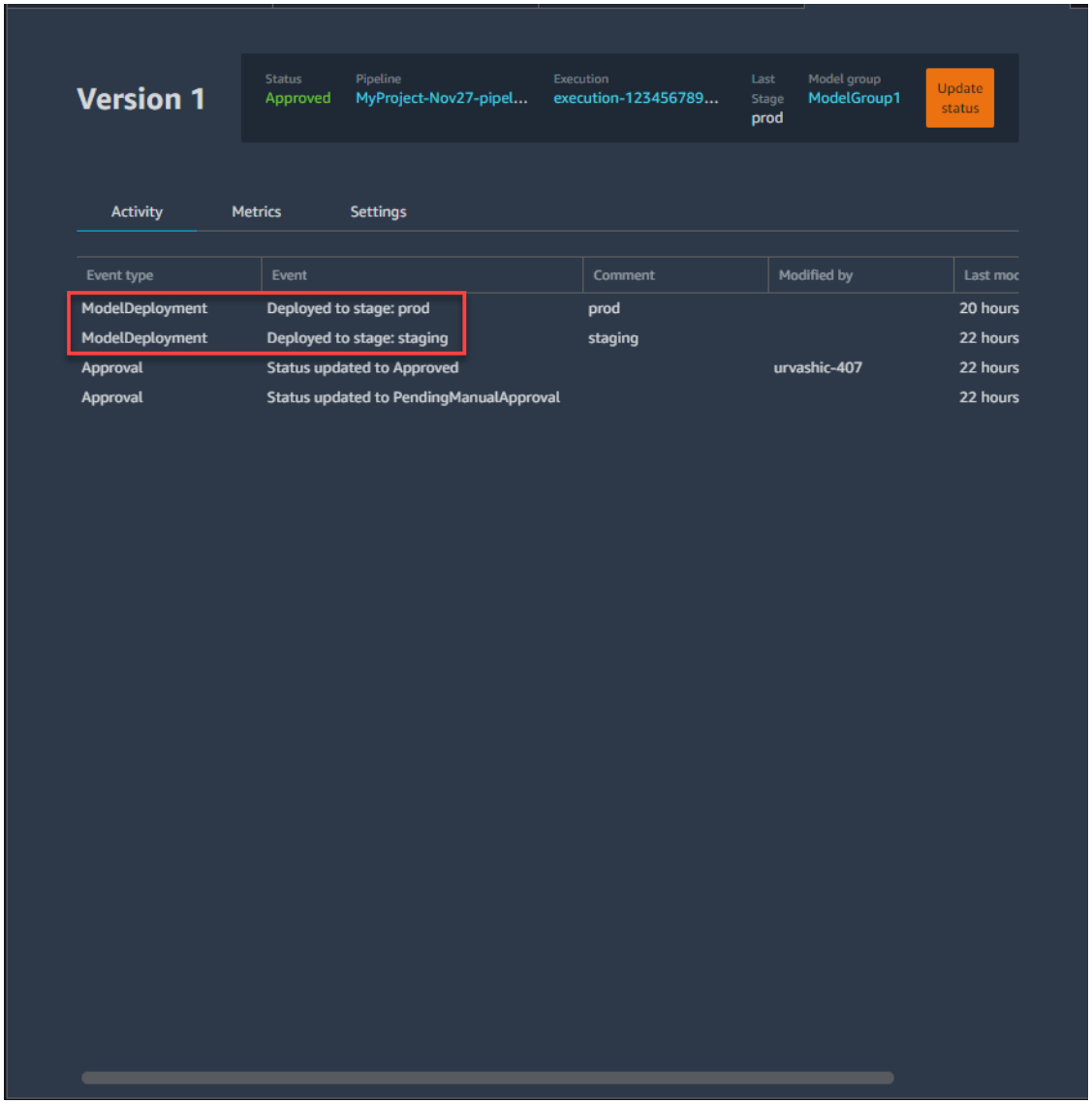
4. 从模型组列表中，选择要查看的模型组。
5. 此时将出现一个新选项卡，其中包含模型组中的模型版本列表。



6. 在模型版本列表中，双击要查看其详细信息的模型版本。



7. 在打开的模型版本选项卡上，选择活动. 模型版本的部署显示为活动列表中的事件，其中包含Event type的模型部署.

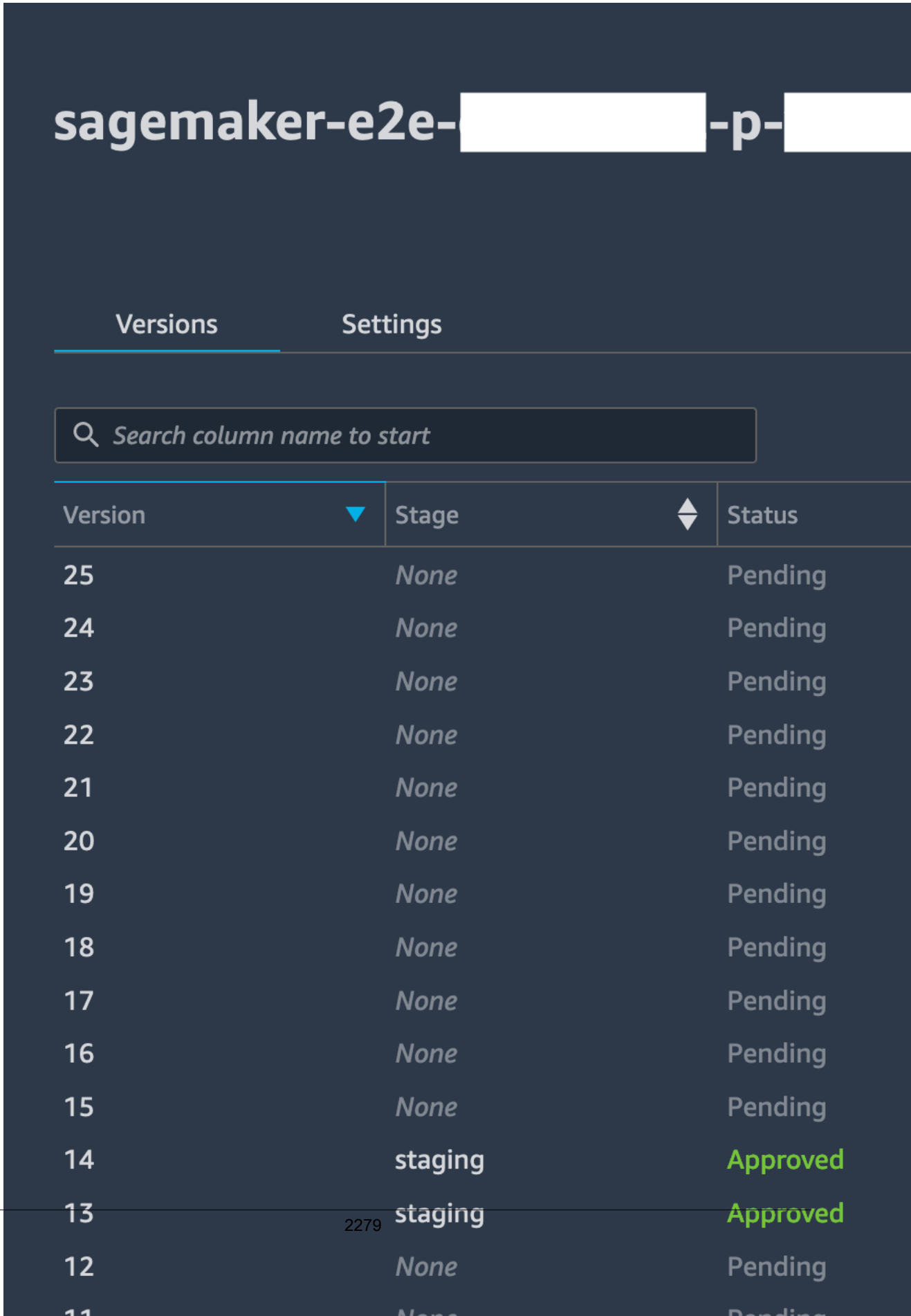


亚马逊 SageMaker 模型注册表

使用以下常见问题解答项目查找有关的常见问题的答案 SageMaker 模型注册表。

问：我应该如何将模型组织成模型组和模型包 SageMaker 模型注册表？

模型包是作为版本控制实体注册到模型注册表中的实际模型。请注意，在 SageMaker 中使用模型包有两种方法。一个是与SageMaker Marketplace— 这些模型包没有版本控制。另一个是与 SageMaker 模型注册表，其中模型包必须被版本控制。模型注册表会接收您重新训练的每个新模型，为其提供一个版本，然后将其分配给模型注册表内的模型组。下图显示了一个模型组的示例，其中包含 25 个连续版本控制模型。


sagemaker-e2e-[redacted]-p-[redacted]
Versions
Settings
Search column name to start
Version
Stage
Status
25 None Pending
24 None Pending
23 None Pending
22 None Pending
21 None Pending
20 None Pending
19 None Pending
18 None Pending
17 None Pending
16 None Pending
15 None Pending
14 staging Approved
13 staging Approved
12 None Pending

问：模型注册表与 Amazon Elastic Container Registry (Amazon ECR) 有何不同？

SageMaker 的模型注册表是机器学习模型的元数据存储。Amazon Elastic 容器注册表是一个存储所有容器的存储库。在模型注册表中，模型是版本化的，并将其注册为模型组中的模型包。每个模型包都包含一个指向与训练模型关联的模型文件的 Amazon S3 URI 和一个指向服务模型时使用的容器的 Amazon ECR URI。

使用在边缘部署模型 SageMaker Edge 管理器

亚马逊 SageMaker Edge Manager 为边缘设备提供模型管理，因此您可以在智能相机、机器人、个人计算机和移动设备等边缘设备队列上优化、保护、监控和维护机器学习模型。

为什么使用边缘管理器？

许多机器学习 (ML) 用例都需要在边缘设备队列上运行机器学习模型，这样您就可以实时获得预测，保护最终用户的隐私，并降低网络连接成本。随着专为机器学习设计的低功耗边缘硬件的可用性不断增加，现在可以在边缘设备上运行多个复杂的神经网络模型。

但是，在边缘设备上运行 ML 模型具有挑战性，因为设备与云实例不同，计算、内存和连接都有限。部署模型后，您需要持续监控模型，因为模型漂移可能会导致模型的质量随时衰减。监控设备队列中的模型很困难，因为您需要编写自定义代码才能从设备中收集数据样本并识别预测中的偏差。此外，模型通常被硬编码到应用程序中。要更新模型，必须重建和更新整个应用程序或设备固件，这可能会中断您的操作。

与 SageMaker 边缘管理器，您可以跨边缘设备群优化、运行、监控和更新机器学习模型。

如何工作？

从较高的层面上说，有五个主要组件 SageMaker Edge Manager 工作流程：使用编译模型 SageMaker Neo，打包新编译模型，将模型部署到设备上，在 SageMaker 推理引擎（边缘管理器代理），以及在设备上维护模型。

SageMaker 边缘管理器使用 SageMaker Neo 只需单击一下即可针对目标硬件优化模型，然后在部署之前对模型进行加密签名。使用 SageMaker Edge Manager，您可以对边缘设备中的模型输入和输出数据进行采样，然后将其发送到云进行监控和分析，还可以查看一个仪表板，该仪表板可跟踪和直观报告在 SageMaker 控制台。

SageMaker Edge Manager 将以前只能在云中提供的功能扩展到边缘，因此开发人员可以使用亚马逊不断提高模型质量 SageMaker 模型监视器用于漂移检测，然后使用重新标记数据 SageMaker 在 SageMaker 中重新训练模型。

如何使用 SageMaker Edge Manager？

如果您是首次接触的用户 SageMaker Edge Manager，建议执行以下操作：

1. 阅读开始使用部分-此部分引导您完成设置第一个边缘包装作业和创建第一个队组的过程。
2. 探索边缘管理器 Jupyter 笔记本示例-例如笔记本存储在亚马逊 sagemaker 示例 GitHub 中的存储库sagemaker_edge_managerfolder。

入门

本指南演示了如何完成注册、部署和管理设备队列的必要步骤，以及如何满足亚马逊的需求 SageMaker Edge Manager 先决条件。

主题

- 设置 (p. 2281)
- 训练、编译和 Package 模型 (p. 2283)
- 创建和注册队列并验证设备 (p. 2286)
- 下载并设置边缘管理器 (p. 2289)
- 运行代理 (p. 2292)

设置

在您开始使用之前 SageMaker Edge Manager 要管理设备队列上的模型，您必须首先为两者创建 IAM 角色 SageMaker 和Amazon IoT. 您还需要创建至少一个 Amazon S3 存储桶，您将在其中存储预训练的模型，即您的输出 SageMaker Neo 编译作业，以及来自边缘设备的输入数据。

1. 设置Amazonaccount.

   创建AmazonIAM 管理员用户和 IAM 管理员用户。有关如何设置您的说明Amazon账户，请参阅如何创建和激活新的Amazon账户？ 有关如何在您的中创建管理员用户的说明Amazon账户，请参阅创建您的第一个 IAM 管理员用户和组.
2. 为 Amazon SageMaker 创建 IAM 角色。

   SageMaker Edge Manager 需要对 Amazon S3 存储桶 URI 的访问权限。为此，请创建可以运行的 IAM 角色 SageMaker 并有访问 Amazon S3 的权限。使用这个角色， SageMaker 您可以在您的账户下运行并访问 Amazon S3 存储桶。

   您可以使用 IAM 控制台创建 IAM 角色，AmazonSDK for Python (Boto3)，或Amazon CLI. 以下是如何创建 IAM 角色并使用 IAM 控制台附加必要的策略的示例。

   a. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
   b. 在 IAM 控制台的导航窗格中，选择 Roles，然后选择 Create role。
   c. 对于选择受信任实体的类型，选择 Amazon 服务。
   d. 选择您希望允许其承担此角色的服务。在此情况下，请选择SageMaker. 接下来，选择 Next (下一步)：Permissions (下一步：权限)。

      - 这会自动创建 IAM 策略，该策略授予对相关服务（如 Amazon S3、Amazon ECR 和）的访问权限 CloudWatch 日志。

   e. 选择 Next:。标签。
   f. （可选）通过以键值对的形式附加标签来向角色添加元数据。有关将在 IAM 中使用标签的更多信息，请参阅 Tagging IAM resources（标记 IAM 资源）。

g. 选择 Next:。审核。
h. 输入Role name (角色名称).
i. 如果可能，键入角色名称或角色名称后缀。角色名称在您的 Amazon 账户内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 `PRODROLE` 和 `prodrole` 的角色。由于其他 Amazon 资源可能引用该角色，角色创建完毕后无法编辑角色名称。
j. (可选) 对于 Role description，键入新角色的描述。
k. 检查角色，然后选择 Create role。

   请注意 SageMaker 角色 ARN，您可以使用它创建编译作业 SageMaker Neo 和边缘管理器的打包工作。要使用控制台查找角色 ARN，请执行以下操作：

   i. 转到 IAM 控制台：https://console.aws.amazon.com/iam/
   ii. Select角色.
   iii. 通过在搜索字段中键入角色名称来搜索刚才创建的角色。
   iv. 选择角色。
   v. 角色 ARN 位于摘要页.

3. 为 创建 IAM 角色Amazon IoT.

   这些区域有：Amazon IoT您创建的 IAM 角色用于授权事物对象。您还可以使用 IAM 角色 ARN 创建设备队列并使用 SageMaker 客户端对象注册。

   在中配置 IAM 角色Amazon账户，以便凭证提供商代表设备队列中的设备承担。然后，附加策略以授权设备与其进行交互。Amazon IoT服务。

   为 创建角色Amazon IoT无论是以编程方式还是使用 IAM 控制台，类似于在为 SageMaker 创建角色时所做的操作。

   a. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
   b. 在 IAM 控制台的导航窗格中，选择 Roles，然后选择 Create role。
   c. 对于选择受信任实体的类型，选择 Amazon 服务。
   d. 选择您希望允许其承担此角色的服务。在此情况下，请选择IoT. SelectIoT作为使用案例.
   e. 选择 Next:。Permissions (下一步：权限)。
   f. 选择 Next:。标签。
   g. （可选）通过以键值对的形式附加标签来向角色添加元数据。有关将在 IAM 中使用标签的更多信息，请参阅 Tagging IAM resources（标记 IAM 资源）。
   h. 选择 Next:。审核。
   i. 输入Role name (角色名称). 角色名称必须以开头`SageMaker`.
   j. (可选) 对于 Role description，键入新角色的描述。
   k. 检查角色，然后选择 Create role。
   l. 创建角色后，选择角色在 IAM 控制台中。通过在搜索字段中返回的子位置类型。
   m. 选择你的角色。
   n. 下一步，选择附加策略.
   o. 搜索`AmazonSageMakerEdgeDeviceFleetPolicy`中的搜索字段中返回的子位置类型。Select `AmazonSageMakerEdgeDeviceFleetPolicy`.
   p. 选择 Attach policy（附上策略）。
   q. 将以下策略语句添加到信任关系中：

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
```

```
        "Effect": "Allow",
        "Principal": {"Service": "credentials.iot.amazonaws.com"},
        "Action": "sts:AssumeRole"
      },
      {
        "Effect": "Allow",
        "Principal": {"Service": "sagemaker.amazonaws.com"},
        "Action": "sts:AssumeRole"
      }
  ]
}
```

信任策略是JSON 策略文档您可以在其中定义您信任代入该角色的委托人。有关信任策略的更多信息，请参阅。角色术语和概念.

r. 请注意Amazon IoT角色 ARN。您将Amazon IoT角色 ARN 以创建和注册设备队列。要使用控制台查找 IAM 角色 ARN，请执行以下操作：

   i. 转到 IAM 控制台：https://console.aws.amazon.com/iam/
   ii. 选择 Roles（角色）。
   iii. 通过在搜索字段中返回的子位置类型。
   iv. 选择角色。
   v. ARN 在 Summary (摘要) 页面上。

4. 创建 Amazon S3 存储桶。

   SageMaker Neo 和 Edge Manager 从 Amazon S3 存储桶访问您的预编译模型和编译模型。边缘管理器还将来自设备队列的示例数据存储在 Amazon S3 中。

   a. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
   b. 选择 Create bucket (创建存储桶)。
   c. InBucket name中，输入存储桶的名称。
   d. 对于 Region（区域），选择要放置存储桶的Amazon区域。
   e. In阻止公有访问的存储桶设置中，选择要应用于存储桶的设置。
   f. 请选择 Create bucket (创建存储桶)。

   有关创建 Amazon S3 存储桶的更多信息，请参阅。Amazon S3 入门.

训练、编译和 Package 模型

在本部分中，您将创建 SageMaker 和Amazon IoT客户端对象，下载预训练的机器学习模型，将模型上传到 Amazon S3 存储桶，为目标设备编译模型 SageMaker Neo，然后打包模型，以便可以使用 Edge Manager 代理进行部署。

1. 导入库并创建客户端对象。

   本教程使用Amazon SDK for Python (Boto3)创建客户端以与 SageMaker、Amazon S3 和交互Amazon IoT.

   如以下示例所示，请导入 Boto3、指定 Region 并初始化所需的客户端对象：

```
import boto3
import json
import time

AWS_REGION = 'us-west-2'# Specify your Region
bucket = 'bucket-name'
```

```
sagemaker_client = boto3.client('sagemaker', region_name=AWS_REGION)
iot_client = boto3.client('iot', region_name=AWS_REGION)
```

定义变量并为其分配您为其创建的角色 ARN SageMaker 和Amazon IoT如字符串

```
# Replace with the role ARN you created for SageMaker
sagemaker_role_arn = "arn:aws:iam::<account>:role/*"

# Replace with the role ARN you created for Amazon IoT.
# Note: The name must start with 'SageMaker'
iot_role_arn = "arn:aws:iam::<account>:role/SageMaker*"
```

2. 训练机器学习模型。

   请参阅使用 Amazon SageMaker 训练模型有关如何使用 SageMaker 训练机器学习模型的更多信息。您可以选择将本地训练的模型直接上传到 Amazon S3 URI 存储桶中。

   如果您还没有模型，则可以在本教程的后续步骤中使用预训练的模型。例如，您可以保存 MobileNet V2 模型来自 TensorFlow 框架。 MobileNet V2 是针对移动应用程序进行了优化的图像分类模型。有关 的更多信息 MobileNet V2，请参阅MobileNet GitHub 自述文件.

   在你的 Jupyter 笔记本中键入以下内容以保存预训练过的 MobileNet V2 型号：

   ```
   # Save the MobileNet V2 model to local storage
      import tensorflow as tf
      model = tf.keras.applications.MobileNetV2()
      model.save("mobilenet_v2.h5")
   ```

   Note

   - 如果你没有 TensorFlow 已安装，您可以运行`pip install tensorflow=2.4`
   - 使用 TensorFlow 本教程的 2.4 或更低版本。

   模型将保存到`mobilenet_v2.h5`文件。在打包模型之前，您需要首先使用 SageMaker Neo。请参阅支持的框架、设备、系统和架构 (p. 2365)检查你的版本是否 TensorFlow (或其他选择框架) 目前受到支持 SageMaker Neo。

   SageMaker Neo 要求将模型存储为压缩的 TAR 文件。将其重新打包为压缩的 TAR 文件 (*.tar.gz)：

   ```
   # Package MobileNet V2 model into a TAR file
      import tarfile

      tarfile_name='mobilenet-v2.tar.gz'

      with tarfile.open(tarfile_name, mode='w:gz') as archive:
          archive.add('mobilenet-v2.h5')
   ```

3. 将模型上载到 Amazon S3。

   一旦您有机器学习模型，请将其存储在 Amazon S3 存储桶中。以下示例使用Amazon CLI将模型上传到您之前在名为的目录中创建的 Amazon S3 存储桶模型. 在你的 Jupyter 笔记本中键入以下内容：

   ```
   !aws s3 cp mobilenet-v2.tar.gz s3://{bucket}/models/
   ```

4. 使用编译模型 SageMaker Neo。

使用编译您的机器学习模型 SageMaker Neo 用于边缘设备。您需要了解存储训练模型的 Amazon S3 存储桶 URI、用于训练模型的机器学习框架、模型输入的形状以及目标设备。

对于 MobileNet V2 模型，请使用以下内容：

```
framework = 'tensorflow'
target_device = 'jetson_nano'
data_shape = '{"data":[1,3,224,224]}'
```

SageMaker Neo 需要基于您使用的深度学习框架的特定模型输入形状和模型格式。有关如何保存模型的更多信息，请参阅。SageMaker Neo 需要什么输入数据形状？ (p. 2327). 有关 Neo 支持的设备和框架的更多信息，请参阅。支持的框架、设备、系统和架构 (p. 2365).

使用`CreateCompilationJob`使用创建编译作业的 API SageMaker Neo。为编译作业提供名称，SageMaker 角色 ARN、存储模型的 Amazon S3 URI、模型的输入形状、框架的名称、所需的 Amazon S3 URI SageMaker 存储编译后的模型和边缘设备目标。

```
# Specify the path where your model is stored
model_directory = 'models'
s3_model_uri = 's3://{}/{}/{}'.format(bucket, model_directory, tarfile_name)

# Store compiled model in S3 within the 'compiled-models' directory
compilation_output_dir = 'compiled-models'
s3_output_location = 's3://{}/{}/'.format(bucket, compilation_output_dir)

# Give your compilation job a name
compilation_job_name = 'getting-started-demo'

sagemaker_client.create_compilation_job(CompilationJobName=compilation_job_name,
                                        RoleArn=sagemaker_role_arn,
                                        InputConfig={
                                            'S3Uri': s3_model_uri,
                                            'DataInputConfig': data_shape,
                                            'Framework' : framework.upper()},
                                        OutputConfig={
                                            'S3OutputLocation': s3_output_location,
                                            'TargetDevice': target_device},
                                        StoppingCondition={'MaxRuntimeInSeconds': 900})
```

5. Package 你的编译模型。

   打包工作需要 SageMaker NEO — 编译模型并进行任何必要的更改，以便使用推理引擎 Edge Manager 代理来部署模型。要打包模型，请使用`create_edge_packaging`API 或 SageMaker 控制台。

   您需要提供用于 Neo 编译作业的名称、打包作业的名称、角色 ARN（请参阅设置 (p. 2281)部分）、模型的名称、模型版本以及用于打包任务输出的 Amazon S3 存储桶 URI。请注意，Edge Manager 打包作业名称区分大小写。以下示例说明如何使用 API 创建打包作业。

   ```
   edge_packaging_name='edge-packaging-demo'
   model_name="sample-model"
   model_version="1.1"
   ```

   定义要将打包模型存储到的 Amazon S3 URI。

   ```
   # Output directory where you want to store the output of the packaging job
   packaging_output_dir = 'packaged_models'
   packaging_s3_output = 's3://{}/{}'.format(bucket, packaging_output_dir)
   ```

使用`CreateEdgePackagingJob`来打包你的新编译模型。提供边缘打包作业的名称以及为编译作业提供的名称（在本例中，该名称存储在变量中`compilation_job_name`)。还提供模型的名称、模型的版本（用于帮助您跟踪您使用的模型版本）以及所需的 S3 URI SageMaker 来存储打包的模型。

```
sagemaker_client.create_edge_packaging_job(
                    EdgePackagingJobName=edge_packaging_name,
                    CompilationJobName=compilation_job_name,
                    RoleArn=sagemaker_role_arn,
                    ModelName=model_name,
                    ModelVersion=model_version,
                    OutputConfig={
                        "S3OutputLocation": packaging_s3_output
                        }
                    )
```

创建和注册队列并验证设备

在本部分中，您将创建Amazon IoTthing 对象、创建设备队列、注册设备队列以便与云进行交互、创建 X.509 证书以验证您的设备Amazon IoT Core，将角色别名与Amazon IoT这是在您创建队组时生成的，请获取Amazon凭证提供商的账户特定终端节点，获取官方的 Amazon Root CA 文件，然后将 Amazon CA 文件上传到 Amazon S3。

1. CreateAmazon IoT东西。

   SageMaker 边缘管理器利用Amazon IoT Core用于促进边缘设备和终端之间的连接的服务Amazon云端。你可以利用现有的Amazon IoT将设备设置为与 Edge Manager 配合使用后的功能。

   将设备连接到Amazon IoT，您需要创建Amazon IoT 事物对象，创建并注册客户端证书AmazonIoT，然后为您的设备创建和配置 IAM 角色。

   首先，创建Amazon IoT事物对象Amazon IoT客户端 (`iot_client`) 你之前用 Boto3 创建的。以下示例演示如何创建两个事物对象：

   ```
   iot_thing_name = 'sample-device'
   iot_thing_type = 'getting-started-demo'

   iot_client.create_thing_type(
       thingTypeName=iot_thing_type
   )

   # Create an Amazon IoT thing objects
   iot_client.create_thing(
       thingName=iot_thing_name,
       thingTypeName=iot_thing_type
   )
   ```

2. 创建您的设备队列。

   使用创建设备队列 SageMaker 在上一步中定义的客户端对象。您也可以使用 SageMaker 控制台以创建设备队列。

   ```
   import time
   device_fleet_name="demo-device-fleet" + str(time.time()).split('.')[0]
   device_name="sagemaker-edge-demo-device" + str(time.time()).split('.')[0]
   ```

   指定您的 IoT 角色 ARN。这让您可以Amazon IoT向设备授予临时证书。

```
device_model_directory='device_output'
s3_device_fleet_output = 's3://{}/{}'.format(bucket, device_model_directory)

sagemaker_client.create_device_fleet(
    DeviceFleetName=device_fleet_name,
    RoleArn=iot_role_arn, # IoT Role ARN specified in previous step
    OutputConfig={
        'S3OutputLocation': s3_device_fleet_output
    }
)
```

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon IoT角色别名是在创建设备队列时创建的。此角色别名与Amazon IoT使用`iot_client`对象在后续步骤中。

3. 注册您的设备队列。

   要与云进行交互，您需要将设备注册到 SageMaker Edge 管理器。在此示例中，您将单个设备注册到您创建的队列中。要注册设备，您需要提供设备名称和Amazon IoT事物名称，如以下示例所示：

```
# Device name should be 36 characters
device_name = "sagemaker-edge-demo-device" + str(time.time()).split('.')[0]

sagemaker_client.register_devices(
    DeviceFleetName=device_fleet_name,
    Devices=[
        {
            "DeviceName": device_name,
            "IotThingName": iot_thing_name
        }
    ]
)
```

4. 创建 X.509 证书。

   创建之后Amazon IoT事物对象，您必须为事物对象创建 X.509 设备证书。此证书将您的设备验证为 Amazon IoT Core.

   使用以下内容来创建私有密钥、公有密钥和 X.509 证书文件，Amazon IoT客户端定义 (`iot_client`) 之前的。

```
# Creates a 2048-bit RSA key pair and issues an X.509 # certificate
# using the issued public key.
create_cert = iot_client.create_keys_and_certificate(
    setAsActive=True
)

# Get certificate from dictionary object and save in its own
with open('./device.pem.crt', 'w') as f:
    for line in create_cert['certificatePem'].split('\n'):
        f.write(line)
        f.write('\n')
# Get private key from dictionary object and save in its own
with open('./private.pem.key', 'w') as f:
    for line in create_cert['keyPair']['PrivateKey'].split('\n'):
        f.write(line)
        f.write('\n')
# Get a private key from dictionary object and save in its own
with open('./public.pem.key', 'w') as f:
    for line in create_cert['keyPair']['PublicKey'].split('\n'):
        f.write(line)
        f.write('\n')
```

5. 将角色别名与Amazon IoT.

   使用 SageMaker 创建设备队列时 (`sagemaker_client.create_device_fleet()`)，将为您生成角色别名。网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon IoT角色别名为已连接的设备提供了一种机制，可以对Amazon IoT使用 X.509 证书，然后获得短暂生命Amazon来自 IAM 角色的凭证，该角色与Amazon IoT角色别名。您可以使用角色别名更改设备的角色，而无需更新设备。使用`DescribeDeviceFleet`获取角色别名和 ARN。

   ```
   # Print Amazon Resource Name (ARN) and alias that has access
   # to Amazon Internet of Things (IoT).
   sagemaker_client.describe_device_fleet(DeviceFleetName=device_fleet_name)

   # Store iot role alias string in a variable
   # Grabs role ARN
   full_role_alias_name =
    sagemaker_client.describe_device_fleet(DeviceFleetName=device_fleet_name)
   ['IotRoleAlias']
   start_index = full_role_alias_name.find('SageMaker') # Find beginning of role name
   role_alias_name = full_role_alias_name[start_index:]
   ```

   使用`iot_client`以便将创建设备队列时生成的角色别名与Amazon IoT：

   ```
   role_alias = iot_client.describe_role_alias(
                       roleAlias=role_alias_name)
   ```

   有关 IAM 角色别名的更多信息，请参阅。角色别名允许访问未使用的服务.

   您创建并注册了证书Amazon IoT之前是为了成功验证您的设备。现在，您需要创建策略并将策略附加到证书，以便授权对安全令牌的请求。

   ```
   alias_policy = {
     "Version": "2012-10-17",
     "Statement": {
       "Effect": "Allow",
       "Action": "iot:AssumeRoleWithCertificate",
       "Resource": role_alias['roleAliasDescription']['roleAliasArn']
     }
   }

   policy_name = 'aliaspolicy-'+ str(time.time()).split('.')[0]
   aliaspolicy = iot_client.create_policy(policyName=policy_name,
                                          policyDocument=json.dumps(alias_policy))

   # Attach policy
   iot_client.attach_policy(policyName=policy_name,
                               target=create_cert['certificateArn'])
   ```

6. 获取您的Amazon凭证提供程序的账户特定的终端节点。

   边缘设备需要终端节点才能使用凭据。获取您的Amazon凭证提供程序的账户特定的终端节点。

   ```
   # Get the unique endpoint specific to your Amazon account that is making the call.
   iot_endpoint = iot_client.describe_endpoint(
       endpointType='iot:CredentialProvider'
   )

   endpoint="https://{}/role-aliases/{}/
   credentials".format(iot_endpoint['endpointAddress'],role_alias_name)
   ```

7. 获取官方的 Amazon root CA 文件并将其上传到 Amazon S3 存储桶。

在你的 Jupyter 笔记本中使用以下内容或Amazon CLI（如果你使用终端，请删除 '!' 魔术函数):

```
!wget https://www.amazontrust.com/repository/AmazonRootCA1.pem
```

使用终端节点向凭证提供程序发出 HTTPS 请求以返回安全令牌。以下示例命令使用 `curl`，但您可以使用任何 HTTP 客户端。

```
!curl --cert device.pem.crt --key private.pem.key --cacert AmazonRootCA1.pem $endpoint
```

如果证书已验证，请将密钥和证书上传到 Amazon S3 存储桶 URI：

```
!aws s3 cp private.pem.key s3://{bucket}/authorization-files/
!aws s3 cp device.pem.crt s3://{bucket}/authorization-files/
!aws s3 cp AmazonRootCA1.pem s3://{bucket}/authorization-files/
```

通过将密钥和证书移至其他目录来清理工作目录：

```
# Optional - Clean up working directory
!mkdir authorization-files
!mv private.pem.key device.pem.crt AmazonRootCA1.pem authorization-files/
```

下载并设置边缘管理器

边缘管理器代理是边缘设备的推理引擎。使用代理程序对加载到边缘设备上的模型进行预测。代理还收集模型指标并按特定时间间隔捕获数据。

在本节中，您将使用代理程序设置设备。为此，请首先将发布项目和签名根证书从发布存储桶本地复制到您的计算机。在解压缩发布工件后，请将其上传到 Amazon S3。接下来，定义并保存代理的配置文件。提供了一个模板供您复制和粘贴。最后，将发布工件、配置文件和凭据复制到设备。

1. 下载 SageMaker 边缘管理器代理。

   对于受支持的操作系统，该代理以二进制格式发布。此示例在使用 Linux 操作系统并具有 ARM64 架构的 Jetson Nano 上运行推理。有关支持的操作系统和架构设备使用的更多信息，请参阅。支持的设备、芯片架构和系统 (p. 2366).

   从获取最新版本的二进制文件 SageMaker Edge Manager 从 us-west-2 区域发布存储桶。

   ```
   !aws s3 ls s3://sagemaker-edge-release-store-us-west-2-linux-armv8/Releases/ | sort -r
   ```

   这将返回按版本排序的发布工件。

   ```
                              PRE 1.20210512.96da6cc/
                              PRE 1.20210305.a4bc999/
                              PRE 1.20201218.81f481f/
                              PRE 1.20201207.02d0e97/
   ```

   版本采用以下格式：`<MAJOR_VERSION>.<YYYY-MM-DD>.<SHA-7>`. 它由三个组成部分：

   - `<MAJOR_VERSION>`: 发布版本。发布版本当前设置为`1`.
   - `<YYYY-MM-DD>`: 神器发布的时间戳。
   - **<SHA-7>**: 构建发行版的存储库提交 ID。

将压缩的 TAR 文件复制到本地或直接复制到设备上。以下示例说明了如何在本文档发布时复制最新版本对象。

```
!aws s3 cp s3://sagemaker-edge-release-store-us-west-2-linux-x64/
Releases/1.20201218.81f481f/1.20201218.81f481f.tgz ./
```

获得工件后，解压缩压缩的 TAR 文件。以下操作解压缩 TAR 文件并将其存储在名为的目录中`agent_demo`：

```
!mkdir agent_demo
!tar -xvzf 1.20201218.81f481f.tgz -C ./agent_demo
```

将代理发行工件上传到 Amazon S3 存储桶。下面的代码示例复制了内容`agent_demo`然后将其上传到 Amazon S3 存储桶中的名为`agent_demo`：

```
!aws s3 cp --recursive ./agent_demo s3://{bucket}/agent_demo
```

您还需要发布存储桶中的签名根证书：

```
!aws s3 cp s3://sagemaker-edge-release-store-us-west-2-linux-x64/Certificates/us-
west-2/us-west-2.pem ./
```

将签名根证书上传到 Amazon S3 存储桶：

```
!aws s3 cp us-west-2.pem s3://{bucket}/authorization-files/
```

2. 定义 SageMaker Edge Manager 代理配置文件。

   首先，按如下方式定义代理配置文件：

```
sagemaker_edge_config = {
    "sagemaker_edge_core_device_name": "device_name",
    "sagemaker_edge_core_device_fleet_name": "device_fleet_name",
    "sagemaker_edge_core_capture_data_buffer_size": 30,
    "sagemaker_edge_core_capture_data_push_period_seconds": 4,
    "sagemaker_edge_core_folder_prefix": "demo_capture",
    "sagemaker_edge_core_region": "us-west-2",
    "sagemaker_edge_core_root_certs_path": "/agent_demo/certificates",
    "sagemaker_edge_provider_aws_ca_cert_file": "/agent_demo/iot-credentials/
AmazonRootCA1.pem",
    "sagemaker_edge_provider_aws_cert_file": "/agent_demo/iot-credentials/
device.pem.crt",
    "sagemaker_edge_provider_aws_cert_pk_file": "/agent_demo/iot-credentials/
private.pem.key",
    "sagemaker_edge_provider_aws_iot_cred_endpoint": "endpoint",
    "sagemaker_edge_provider_provider": "Aws",
    "sagemaker_edge_provider_s3_bucket_name": bucket,
    "sagemaker_edge_core_capture_data_destination": "Cloud"
}
```

   替换以下内容：

   - `"device_name"`使用你的设备名称（此字符串存储在名为的变量中的前一步中`device_name`）。
   - `"device_fleet_name"`使用设备队列的名称（此字符串存储在名为的变量中的前一步）`device_fleet_name`）

- "endpoint"与您的Amazon凭证提供程序的账户特定终端节点（此字符串存储在名为的变量中的前一步中）endpoint)。

接下来，将其保存为 JSON 文件：

```
edge_config_file = open("sagemaker_edge_config.json", "w")
json.dump(sagemaker_edge_config, edge_config_file, indent = 6)
edge_config_file.close()
```

将配置文件上传到 Amazon S3 存储桶：

```
!aws s3 cp sagemaker_edge_config.json s3://{bucket}/
```

3. 将发布工件、配置文件和凭据复制到设备。

在边缘设备本身上执行以下说明。

Note

你必须首先安装 Python，Amazon SDK for Python (Boto3)，以及Amazon CLI在边缘设备上。

在您的设备上打开终端。创建一个文件夹以存储发布工件、您的凭据和配置文件。

```
mkdir agent_demo
cd agent_demo
```

将存储在 Amazon S3 存储桶中的发布项目的内容复制到设备：

```
# Copy release artifacts
aws s3 cp s3://<bucket-name>/agent_demo/ ./ --recursive
```

（发布工件的内容存储在名为的目录中agent_demo在之前的步骤中）。Replace<bucket-name>和agent_demo分别提供 Amazon S3 存储桶的名称和发布项目的文件路径。

转到/bin目录并使二进制文件可执行：

```
cd bin

chmod +x sagemaker_edge_agent_binary
chmod +x sagemaker_edge_agent_client_example

cd agent_demo
```

创建一个目录来存储你的Amazon IoT凭证并将您的凭证从 Amazon S3 存储桶复制到边缘设备（使用变量中定义的相同）bucket：

```
mkdir iot-credentials
cd iot-credentials

aws s3 cp s3://<bucket-name>/authorization-files/AmazonRootCA1.pem ./
aws s3 cp s3://<bucket-name>/authorization-files/device.pem.crt ./
aws s3 cp s3://<bucket-name>/authorization-files/private.pem.key ./

cd ../
```

创建一个目录来存储模型签名根证书：

```
mkdir certificates

cd certificates

aws s3 cp s3://<bucket-name>/authorization-files/us-west-2.pem ./

cd agent_demo
```

将您的配置文件复制到设备：

```
#Download config file from S3
aws s3 cp s3://<bucket-name>/sagemaker_edge_config.json ./

cd agent_demo
```

您的agent_demo边缘设备上的目录应类似于以下内容：

```
###agent_demo
|    ### bin
|        ### sagemaker_edge_agent_binary
|        ### sagemaker_edge_agent_client_example
|    ### sagemaker_edge_config.json
|    ### certificates
|        ###us-west-2.pem
|    ### iot-credentials
|        ### AmazonRootCA1.pem
|        ### device.pem.crt
|        ### private.pem.key
|    ### docs
|        ### api
|        ### examples
|    ### ATTRIBUTIONS.txt
|    ### LICENSE.txt
|    ### RELEASE_NOTES.md
```

运行代理

在本节中，您将使用 gRPC 将代理作为二进制文件运行，并检查您的设备和队列是否正常工作并收集示例数据。

1. 启动代理程序。

   这些区域有： SageMaker Edge Manager 代理可以作为独立进程以可执行文件和可链接格式 (ELF) 可执行二进程的形式运行，也可以作为动态共享对象 (.dll) 链接。作为独立的可执行二进制文件运行是首选模式，并且在 Linux 上受支持。

   此示例使用 gRPC 来运行代理。gRPC 是一个开源的高性能远程过程调用 (RPC) 框架，可以在任何环境中运行。有关 GrPC 的更多信息，请参阅GRPC 文档.

   要使用 GrPC，请执行以下步骤：

   a. 在 .proto 文件中定义服务。
   b. 使用协议缓冲区编译器生成服务器和客户端代码。
   c. 使用 Python（或 gRPC 支持的其他语言）gRPC API 为您的服务编写服务器。
   d. 使用 Python（或 gRPC 支持的其他语言）gRPC API 为您的服务编写客户端。

您下载的发布工件包含一个准备运行代理的 gRPC 应用程序。该示例位于/bin发布工件的目录。这些区域有：sagemaker_edge_agent_binary二进制可执行文件在此目录中。

要使用此示例运行代理，请提供套接字文件 (.sock) 和 JSON .config 文件的路径：

```
./bin/sagemaker_edge_agent_binary -a /tmp/sagemaker_edge_agent_example.sock -c
 sagemaker_edge_config.json
```

2. 检查你的设备。

   检查您的设备是否已连接并对数据进行采样。通过手动或自动进行定期检查，您可以检查设备或机群是否正常工作。

   提供设备所属队组的名称以及唯一的设备标识符。在本地计算机上，运行以下命令：

```
sagemaker_client.describe_device(
    DeviceName=device_name,
    DeviceFleetName=device_fleet_name
)
```

   对于给定模型，您可以看到名称、模型版本、最新采样时间以及上次推断的时间。

```
{
  "DeviceName": "sample-device",
  "DeviceFleetName": "demo-device-fleet",
  "IoTThingName": "sample-thing-name-1",
  "RegistrationTime": 1600977370,
  "LatestHeartbeat": 1600977370,
  "Models":[
    {
        "ModelName": "mobilenet_v2.tar.gz",
        "ModelVersion": "1.1",
        "LatestSampleTime": 1600977370,
        "LatestInference": 1600977370
    }
  ]
}
```

   提供的时间戳LastetHeartbeat表示从设备收到的最后一个信号。LatestSampleTime和LatestInference分别描述最后一个数据样本和推理的时间戳。

3. 检查你的车队。

   检查你的车队是否正在使用GetDeviceFleetReport. 提供设备所属的队列的名称。

```
sagemaer_client.get_device_fleet_report(
    DeviceFleetName=device_fleet_name
)
```

   对于给定模型，您可以看到名称、模型版本、最新采样时间、上次推断的时间以及存储数据样本的 Amazon S3 存储桶 URI。

```
# Sample output
{
 "DeviceFleetName": "sample-device-fleet",
 "DeviceFleetArn": "arn:aws:sagemaker:us-west-2:9999999999:device-fleet/sample-fleet-
name",
 "OutputConfig": {
```

```
                "S3OutputLocation": "s3://fleet-bucket/package_output",
  },
 "AgentVersions":[{"Version": "1.1", "AgentCount": 2}]}
 "DeviceStats": {"Connected": 2, "Registered": 2},
 "Models":[{
            "ModelName": "sample-model",
            "ModelVersion": "1.1",
            "OfflineDeviceCount": 0,
            "ConnectedDeviceCount": 2,
            "ActiveDeviceCount": 2,
            "SamplingDeviceCount": 100
            }]
}
```

设置设备和队列

队列是可用于收集和分析数据的逻辑分组设备的集合。您可以使用 SageMaker Edge Manager 可在智能相机、智能扬声器、机器人和其他边缘设备上运行机器学习模型。

创建一个队列并以编程方式使用Amazon SDK for Python (Boto3)或通过 SageMaker 控制台。

主题

- 创建机群 (p. 2294)
- 注册设备 (p. 2297)
- 检查状态 (p. 2299)

创建机群

您可以使用编程方式创建队列Amazon SDK for Python (Boto3)或通过 SageMaker 控制台https://console.aws.amazon.com/sagemaker.

创建队列 (Boto3)

使用`CreateDeviceFleet`创建队列的 API。为队列指定名称，Amazon IoT的角色 ARN`RoleArn`字段，以及您希望设备存储采样数据的 Amazon S3 URI。

您可以选择性地添加队组、标签和队组的描述Amazon KMS密钥 ID。

```
import boto3

# Create SageMaker client so you can interact and manage SageMaker resources
sagemaker_client = boto3.client("sagemaker", region_name="aws-region")

sagemaker_client.create_device_fleet(
    DeviceFleetName="sample-fleet-name",
    RoleArn="arn:aws:iam::999999999:role/rolename", # IoT Role ARN
    Description="fleet description",
    OutputConfig={
        S3OutputLocation="s3://bucket/",
        KMSKeyId: "1234abcd-12ab-34cd-56ef-1234567890ab",
    },
        Tags=[
        {
            "Key": "string",
            "Value" : "string"
         }
     ],
)
```

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon IoT将在您创建设备队列时为您创建角色别名。这些区域有：Amazon IoT角色别名为已连接的设备提供了一种机制，可以对Amazon IoT使用 X.509 证书然后获得短暂生命Amazon来自 IAM 角色的凭证，该角色与Amazon IoT角色别名。

使用`DescribeDeviceFleet`获取角色别名和 ARN。

```
# Print Amazon Resource Name (ARN) and alias that has access
# to Amazon Internet of Things (IoT).
sagemaker_client.describe_device_fleet(DeviceFleetName=device_fleet_name)['IotRoleAlias']
```

使用`DescribeDeviceFleet`API 来获取您创建的队列的描述。

```
sagemaker_client.describe_device_fleet(
    DeviceFleetName="sample-fleet-name"
)
```

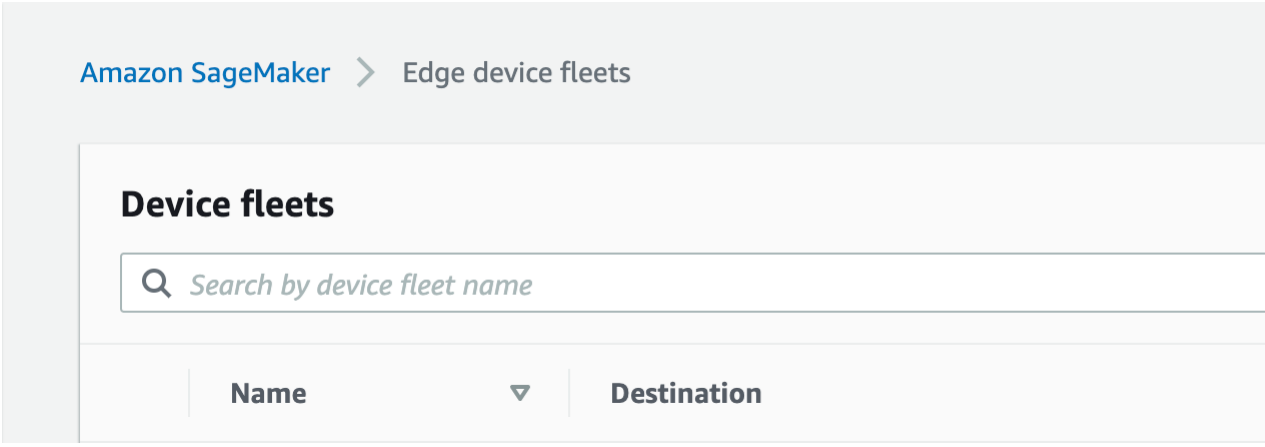
默认情况下，它返回队列的名称、设备队列 ARN、Amazon S3 存储桶 URI、IAM 角色、在中创建的角色别名Amazon IoT、队列创建时间的时间戳以及上次修改队列的时间戳。

```
{ "DeviceFleetName": "sample-fleet-name",
  "DeviceFleetArn": "arn:aws:sagemaker:us-west-2:9999999999:device-fleet/sample-fleet-
name",
  "IAMRole": "arn:aws:iam::999999999:role/rolename",
  "Description": "this is a sample fleet",
  "IoTRoleAlias": "arn:aws:iot:us-west-2:9999999999:rolealias/SagemakerEdge-sample-fleet-
name"
  "OutputConfig": {
              "S3OutputLocation": "s3://bucket/folder",
              "KMSKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
   },
   "CreationTime": "1600977370",
   "LastModifiedTime": "1600977370"}
```

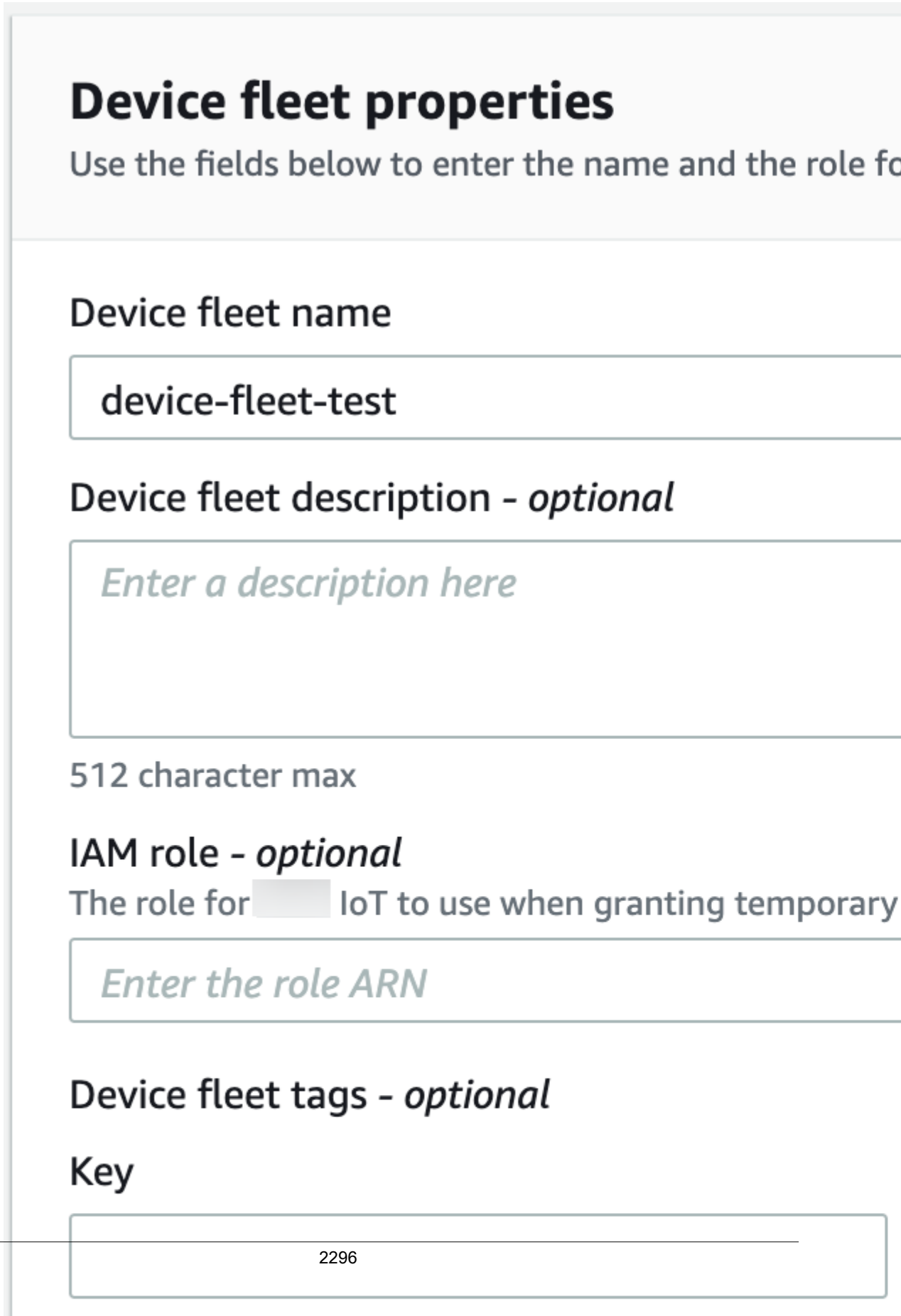
创建队组 (控制台)

您可以使用亚马逊创建边缘管理器打包作业 SageMaker 控制台https://console.aws.amazon.com/sagemaker.

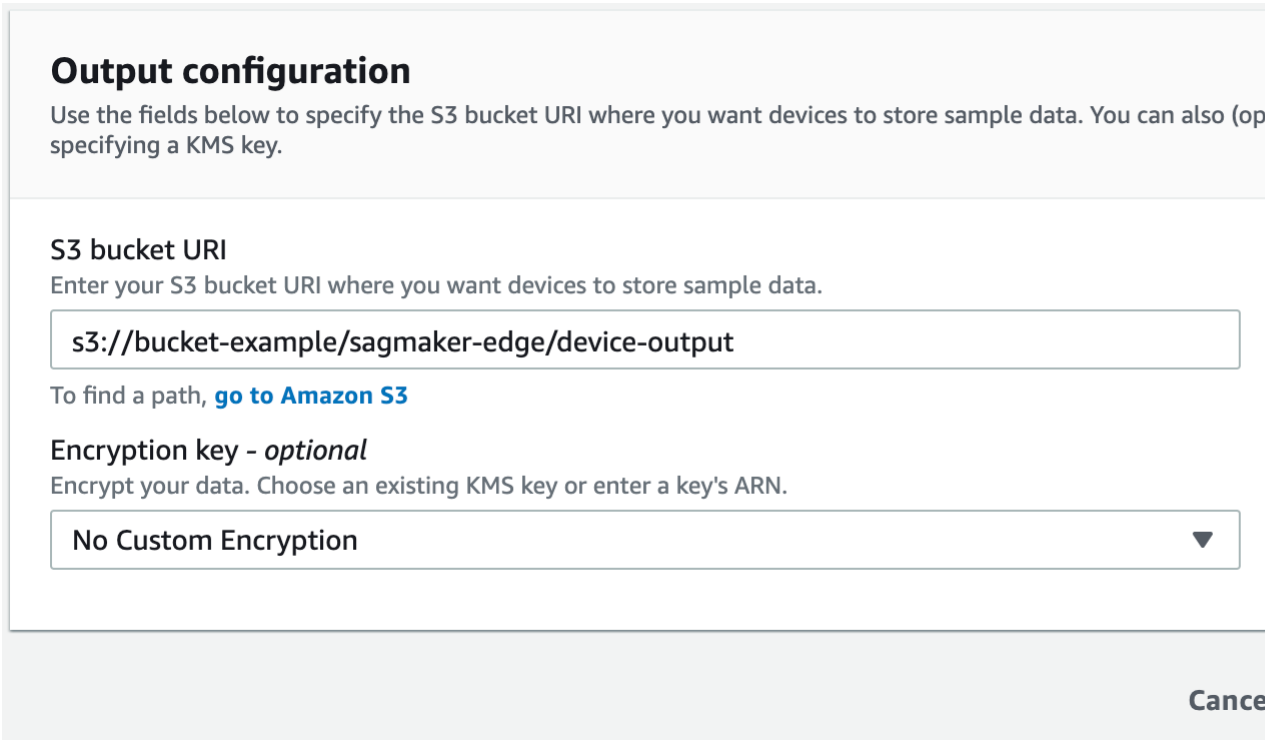
1. 在 SageMaker 控制台，选择Edge 管理器然后选择Edge 设备队组.
2. 选择创建设备队组.



3. 在中输入设备队列的名称设备队组名称字段中返回的子位置类型。选择 Next（下一步）。


Device fleet properties
Use the fields below to enter the name and the role fo
Device fleet name
device-fleet-test
Device fleet description - optional
Enter a description here
512 character max
IAM role - optional
The role for IoT to use when granting temporary
Enter the role ARN
Device fleet tags - optional
Key

4. 在存储库的输出配置页面中，指定要存储设备队列中的示例数据的 Amazon S3 存储桶 URI。您也可以选择通过选择现有的加密密钥来添加加密密钥Amazon KMS键从下拉列表中或输入密钥的 ARN。选择 Submit（提交）。



5. 选择要重定向到设备队列详细信息的设备队列的名称。此页面显示设备队列的名称、ARN、描述（如果您提供了）、队列的创建日期、上次修改队列时间、Amazon S3 存储桶 URI、Amazon KMS密钥 ID（如果提供），Amazon IoT别名（如果提供）和 IAM 角色。如果添加了标签，它们将显示在设备队组标签部分。

注册设备

Important

需要注册设备才能使用 SageMaker Edge 管理器。

您可以使用编程方式创建队列Amazon SDK for Python (Boto3)或通过 SageMaker 控制台https://console.aws.amazon.com/sagemaker.

注册设备 (Boto3)

要注册设备，请先创建并注册Amazon IoT事物对象并配置 IAM 角色。 SageMaker Edge Manager 利用Amazon IoT Core便于边缘设备和云之间的连接的服务。你可以利用现有的Amazon IoT将设备设置为与 Edge Manager 配合使用后的功能。

将设备连接到Amazon IoT你需要创建Amazon IoTthing 对象，创建并注册客户端证书Amazon IoT，然后为您的设备创建和配置 IAM 角色。

请参阅入门指南有关深入的例子或了解Amazon实践教程中的 IoT Core 服务.

使用`RegisterDevices`用于注册设备的 API。提供您希望设备成为其中一部分的队列的名称以及设备的名称。您可以选择性地将描述添加到设备、标签和Amazon IoT与设备关联的事物名称。

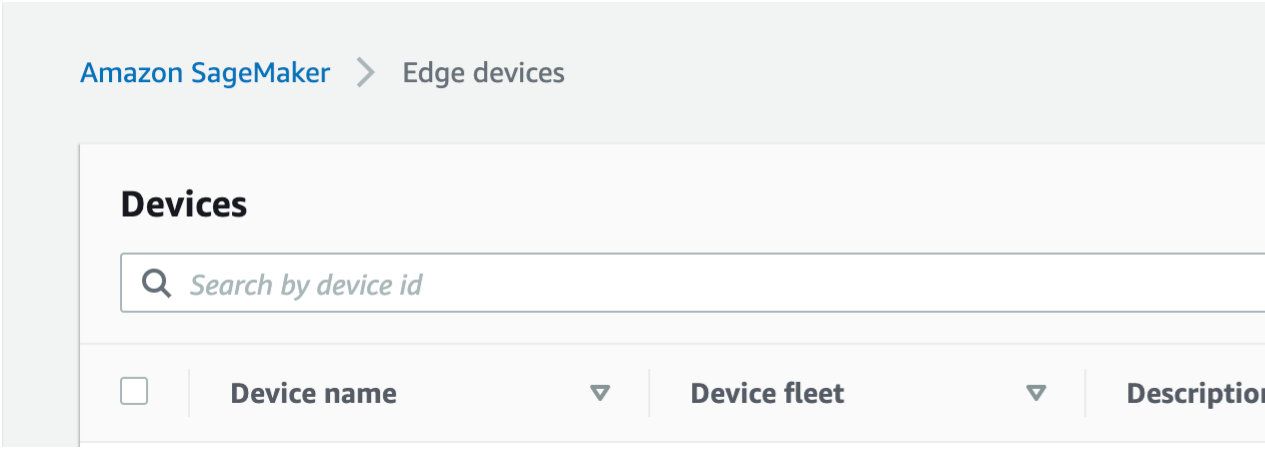
```
sagemaker_client.register_devices(
```

```
    DeviceFleetName="sample-fleet-name",
    Devices=[
        {
            "DeviceName": "sample-device-1",
            "IotThingName": "sample-thing-name-1",
            "Description": "Device #1"
        }
     ],
     Tags=[
        {
            "Key": "string",
            "Value" : "string"
         }
     ],
)
```

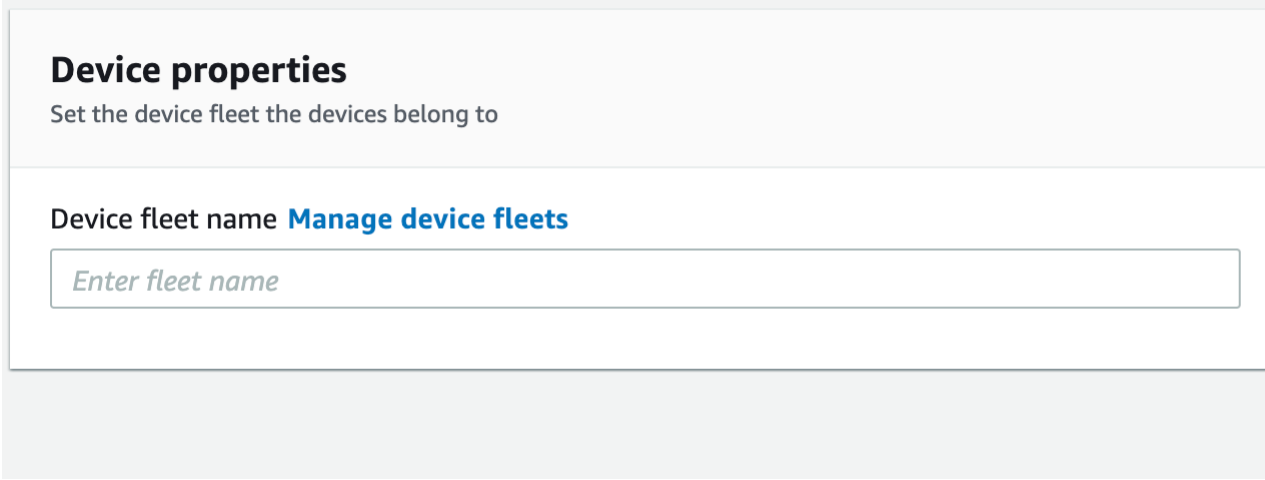
注册设备（控制台）

你可以使用 SageMaker 控制台https://console.aws.amazon.com/sagemaker.

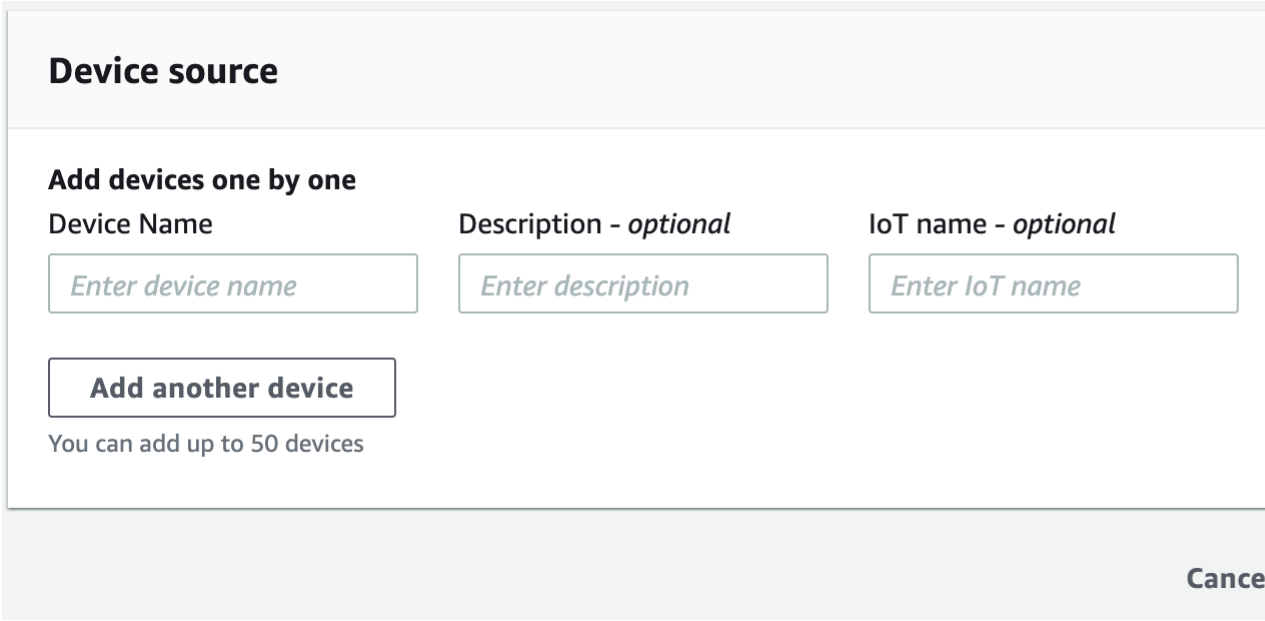
1. 在 SageMaker 控制台，选择Edge 推理然后选择Edge 设备.
2. 选择注册设备.



3. 在设备属性部分中，输入设备所属队列的名称设备队组名称字段中返回的子位置类型。选择 Next（下一步）。



4. 在设备源部分中，逐个添加你的设备。您必须包含设备名称针对机队中的每个设备。您可以选择提供描述（在说明字段）和物联网 (IoT) 对象名称（在IoT 名称字段）。选择提交添加所有设备后。



这些区域有：Devices页面显示已添加的设备的名称、设备所属的队列、注册时间、最后的心跳以及描述和Amazon IoT姓名，如果你提供了一个。

选择设备以查看设备的详细信息，包括设备名称、队列、ARN、描述、IoT 事物名称、设备注册时间和最后一次心跳。

检查状态

检查您的设备或机群是否已连接并对数据进行采样。通过手动或自动进行定期检查，您可以检查设备或机群是否正常工作。

在以下网址使用 Amazon S3 控制台：https://console.aws.amazon.com/s3/以交互方式选择机队进行状态检查。您也可以使用 Amazon SDK for Python (Boto3)。以下介绍了与 Boto3 的不同 API，您可以用来检查设备或队列的状态。请使用最符合您的使用案例的 API。

- 检查单个设备。

  要查看单个设备的状态，请使用DescribeDeviceAPI。如果已将模型部署到设备，则会提供包含一个或多个型号的列表。

```
sagemaker_client.describe_device(
    DeviceName="sample-device-1",
    DeviceFleetName="sample-fleet-name"
)
```

正在运行DescribeDevice返回：

```
{ "DeviceName": "sample-device".
  "Description": "this is a sample device",
  "DeviceFleetName": "sample-device-fleet",
  "IoTThingName": "SampleThing",
  "RegistrationTime": 1600977370,
  "LatestHeartbeat": 1600977370,
  "Models":[
        {
```

```
        "ModelName": "sample-model",
        "ModelVersion": "1.1",
        "LatestSampleTime": 1600977370,
        "LatestInference": 1600977370
       }
   ]
}
```

- 检查设备队列。

  要查看队列的状态，请使用`GetDeviceFleetReport`API。提供设备队列的名称以获取队列摘要。

```
sagemaker_client.get_device_fleet_report(
    DeviceFleetName="sample-fleet-name"
)
```

- 检查心跳。

  队列中的每个设备都会定期生成信号或 “心跳”。心跳可用于检查设备是否正在与边缘管理器进行通信。如果最后一个心跳的时间戳没有更新，则设备可能会出现故障。

  检查设备使用的最后一个心跳`DescribeDevice`API。指定设备的名称和边缘设备所属的队列。

```
sagemaker_client.describe_device(
    DeviceName="sample-device-1",
    DeviceFleetName="sample-fleet-name"
)
```

Package 模型

SageMaker Edge Manager 打包工作占用亚马逊 SageMaker NEO — 编译模型并进行任何必要的更改，以便使用推理引擎 Edge Manager 代理来部署模型。

主题

- 先决条件 (p. 2300)
- Package 模型（Amazon SageMaker 控制台）(p. 2301)
- Package 模型（Boto3）(p. 2305)

先决条件

要打包模型，您必须执行以下操作：

1. 使用 SageMaker Neo 编译机器学习模型。

   如果您尚未执行此操作，请使用编译模型。 SageMaker Neo。有关如何编译模型的更多信息，请参阅。使用 Neo 编译并部署模型. 如果您是首次接触的用户 SageMaker Neo，通过Neo Edge 设备入门.
2. 获取你的编译作业的名字。

   提供编译模型时使用的编译作业名称的名称 SageMaker Neo。打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/然后选择编译作业查找已提交给您的汇编列表Amazonaccount. 提交的编译作业的名称在名称column.
3. 获取您的 IAM ARN。

   您需要 IAM 角色的 Amazon 资源名称 (ARN)，您可以使用该名称 (ARN) 来下载和上传模型并联系 SageMaker Neo。

使用以下方法之一获取 IAM ARN：

- 以编程方式 SageMaker Python 开发工具包

```
import sagemaker

# Initialize SageMaker Session object so you can interact with Amazon resources
sess = sagemaker.Session()

# Get the role ARN
role = sagemaker.get_execution_role()

print(role)
>> arn:aws:iam::<your-aws-account-id>:role/<your-role-name>
```

有关使用的更多信息 SageMaker Python SDK，请参阅SageMaker Python 开发工具包.

- 使用Amazon Identity and Access Management(IAM) 控制台

通过以下网址导航到 IAM 控制台：https://console.aws.amazon.com/iam/。在 IAM 中资源部分，选择角色查看中的角色列表Amazonaccount. 选择或创建具有`AmazonSageMakerFullAccess`、`AWSIoTFullAccess`, 和`AmazonS3FullAccess`.

有关 IAM 的更多信息，请参阅。什么是 IAM？

4. 有 S3 存储桶 URI。

您需要至少有一个亚马逊 Simple Storage Service (Amazon S3) 存储桶 URI 来存储新编译模型、边缘管理器打包作业的输出以及设备队列中的示例数据。

使用以下方法之一创建 Amazon S3 存储桶：

- 以编程方式 SageMaker Python 开发工具包

您可以在会话期间使用默认的 Amazon S3 存储桶。默认存储桶是基于以下格式创建的：`sagemaker-{region}-{aws-account-id}`. 使用 SageMaker Python SDK，使用以下内容：

```
import sagemaker

session=sagemaker.create_session()

bucket=session.default_bucket()
```

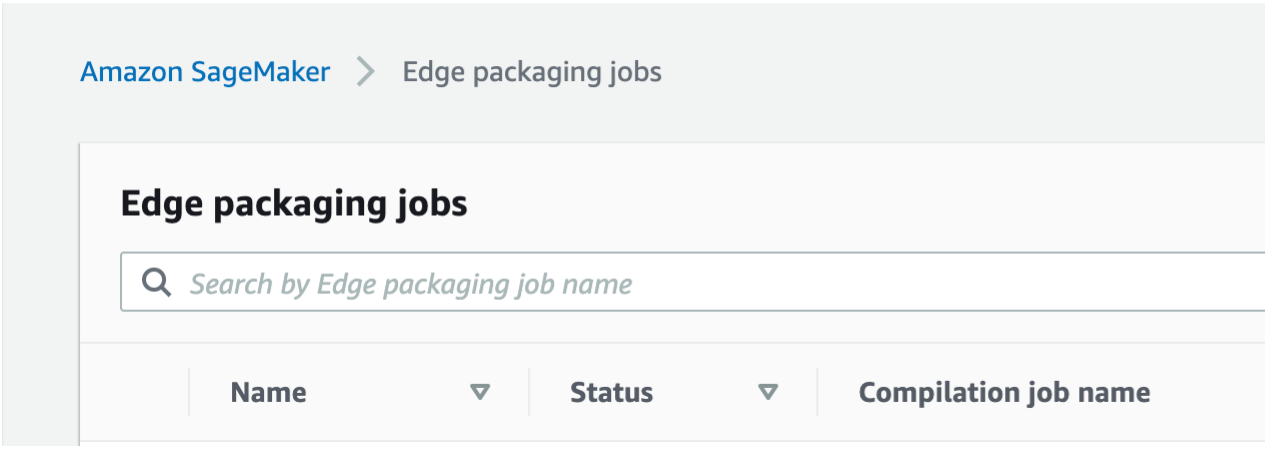
- 使用 Amazon S3 控制台

在以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/请参阅如何创建 S3 存储桶？为了 step-by-step 说明。

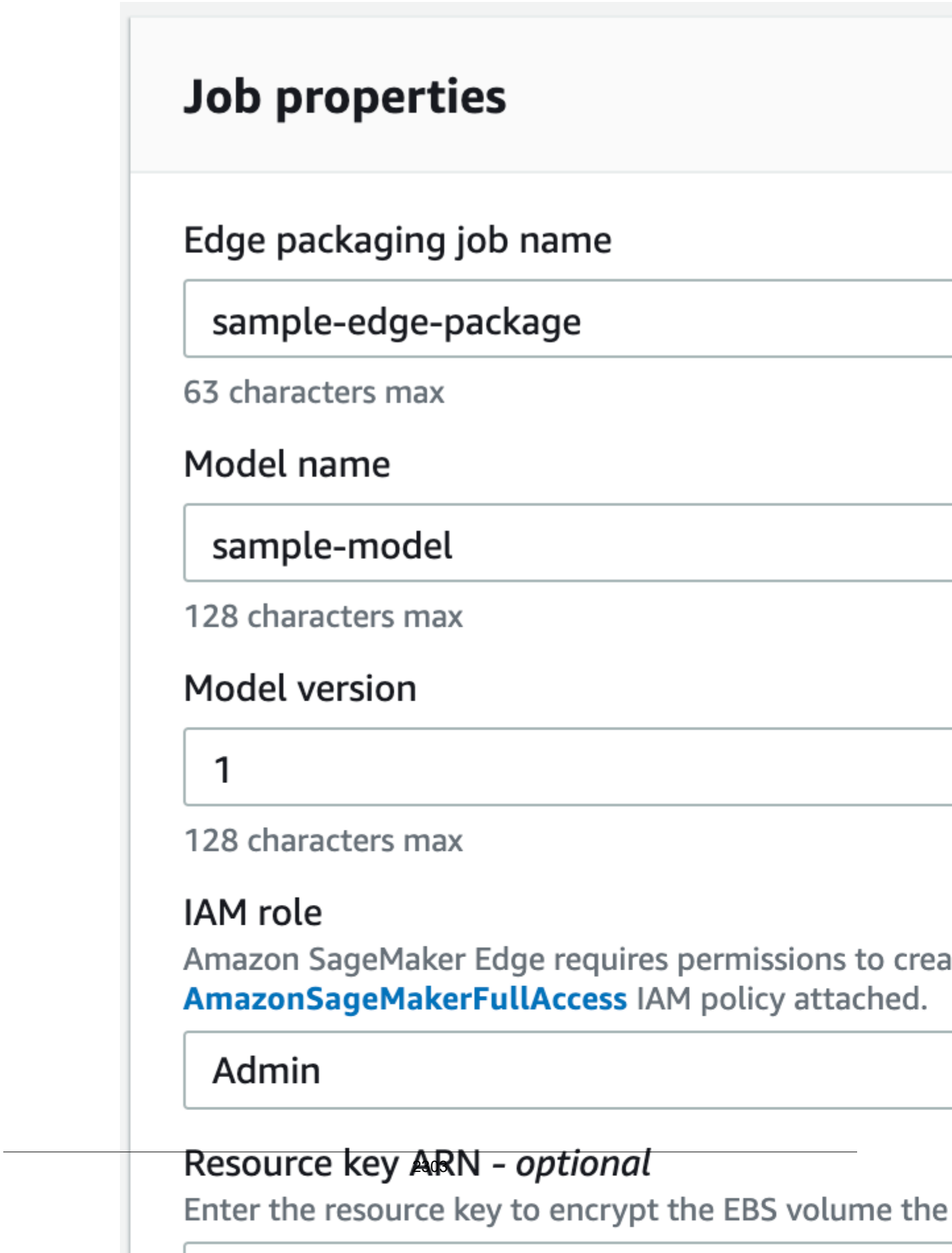
Package 模型（Amazon SageMaker 控制台）

您可以创建 SageMaker 边缘管理器使用 SageMaker 控制台https://console.aws.amazon.com/sagemaker/. 在继续之前，请确保您已满足先决条件 (p. 2300).

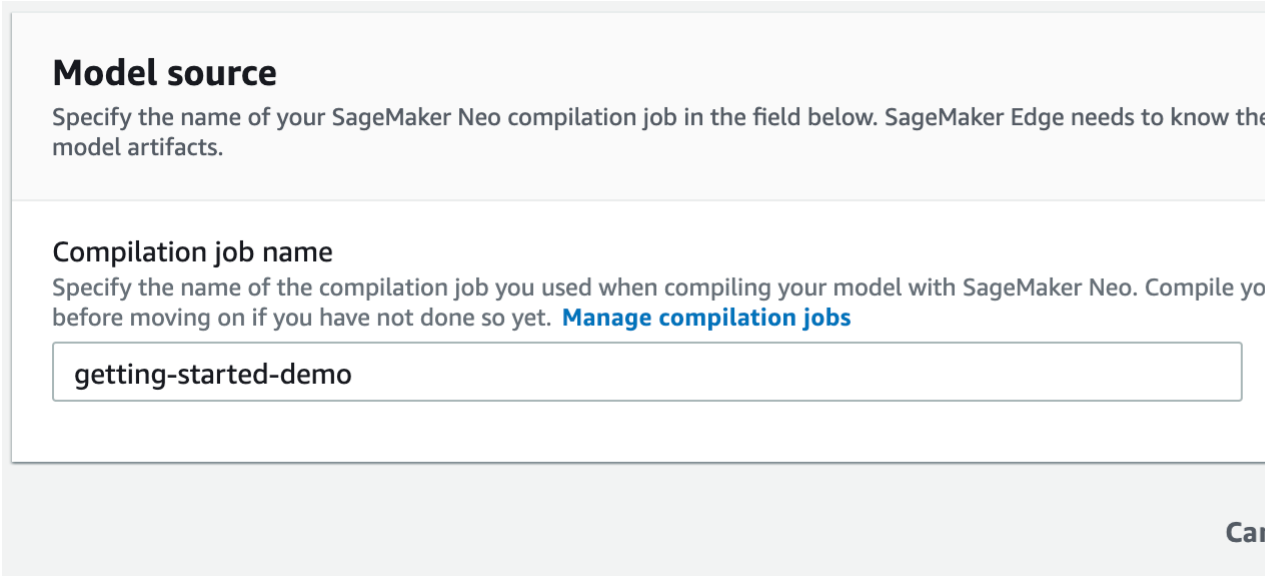
1. 在 SageMaker 控制台，选择Edge 推理然后选择边缘打包作业，如下图所示。



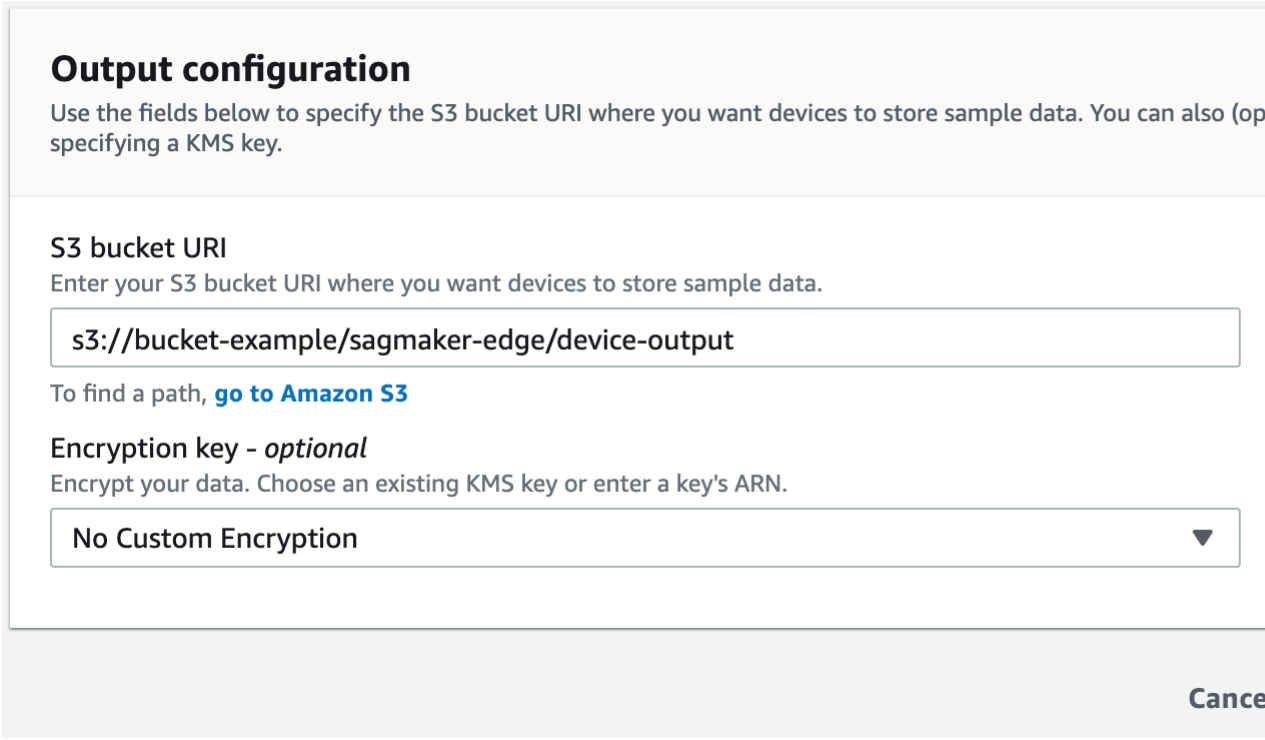
2. 在存储库的任务属性在页面中，输入打包任务的名称Edge 打包作业名称. 请注意，Edge Manager 打包作业名称区分大小写。命名你的模型并给它一个版本：在下面输入模型名称和模型版本，。
3. 接下来，选择IAM 角色. 你可以选择一个角色或者让Amazon为您创建一个角色。（可选）您可以指定资源密钥 ARN和作业标签.
4. 选择 Next（下一步）。


Job properties
Edge packaging job name
sample-edge-package
63 characters max
Model name
sample-model
128 characters max
Model version
1
128 characters max
IAM role
Amazon SageMaker Edge requires permissions to crea
AmazonSageMakerFullAccess IAM policy attached.
Admin
Resource key ARN - optional
Enter the resource key to encrypt the EBS volume the

5. 指定编译模型时使用的编译作业的名称 SageMaker Neo 在编译作业名称字段中返回的子位置类型。选择 Next（下一步）。



6. 在存储库的输出配置在页面上，输入要存储打包作业输出的 Amazon S3 存储桶 URI。



这些区域有：状态上的列边缘包装作业页面应为正在进行中. 打包作业完成后，状态将更新为已完成.

选择打包作业会引导您进入该作业的设置。这些区域有：Job 设置此部分显示作业名称、ARN、状态、创建时间、上次修改时间、打包作业的持续时间和角色 ARN。

这些区域有：输入配置部分显示模型对象的位置、数据输入配置和模型的机器学习框架。

这些区域有：输出配置部分显示打包作业的输出位置、为其编译模型的目标设备以及您创建的任何标记。

7. 选择要重定向到设备队列详细信息的设备队列的名称。此页面显示设备队列的名称、ARN、描述（如果您提供了）、队列的创建日期、上次修改队列时间、Amazon S3 存储桶 URI、Amazon KMS密钥 ID（如果提供），Amazon IoT别名（如果提供）和 IAM 角色。如果添加了标签，它们将显示在设备队组标签部分。

Package 模型（Boto3）

您可以创建 SageMaker 边缘管理器打包作业Amazon SDK for Python (Boto3). 在继续之前，请确保您已满足先决条件 (p. 2300).

要请求边缘打包作业，请使用`CreateEdgePackagingJob`. 你需要为边缘打包作业提供一个名称，你的名字 SageMaker Neo 编译作业、您的角色亚马逊资源名称 (ARN)、模型的名称、模型的版本以及要存储打包作业输出的 Amazon S3 存储桶 URI。请注意，边缘管理器打包作业名称和 SageMaker Neo 编译任务名称区分大小写。

```
# Import Amazon SDK for Python (Boto3)
import boto3

# Create Edge client so you can submit a packaging job
sagemaker_client = boto3.client("sagemaker", region_name='aws-region')

sagemaker_client.create_edge_packaging_job(
    EdgePackagingJobName="edge-packaging-name",
    CompilationJobName="neo-compilation-name",
    RoleArn="arn:aws:iam::99999999999:role/rolename",
    ModelName="sample-model-name",
    ModelVersion="model-version",
    OutputConfig={
        "S3OutputLocation": "s3://your-bucket/",
    }
)
```

您可以使用以下方法检查边缘打包作业的状态：`DescribeEdgePackagingJob`并提供区分大小写的边缘打包作业名称：

```
response = sagemaker_client.describe_edge_packaging_job(
                                    EdgePackagingJobName="edge-packaging-name")
```

这将返回一个可用于轮询打包作业状态的字典：

```
# Optional - Poll every 30 sec to check completion status
import time

while True:
    response = sagemaker_client.describe_edge_packaging_job(
                                        EdgePackagingJobName="edge-packaging-name")

    if response['EdgePackagingJobStatus'] == 'Completed':
        break
    elif response['EdgePackagingJobStatus'] == 'Failed':
        raise RuntimeError('Packaging job failed')
    print('Packaging model...')
    time.sleep(30)
print('Done!')
```

有关打包作业的列表，请使用`ListEdgePackagingJobs`. 您可以使用此 API 搜索特定的打包作业。提供部分名称以筛选打包作业名称`NameContains`，部分名称`ModelNameContains`以筛选模型名称中包含您提供的名称的作业。还可以指定要使用哪个列进行排序`SortBy`，以及要按照哪个方向进行排序`SortOrder`（或者`Ascending`要么`Descending`)。

```
sagemaker_client.list_edge_packaging_jobs(
    "NameContains": "sample",
    "ModelNameContains": "sample",
    "SortBy": "column-name",
    "SortOrder": "Descending"
)
```

要停止打包作业，请使用`StopEdgePackagingJob`并提供边缘打包作业的名称。

```
sagemaker_client.stop_edge_packaging_job(
        EdgePackagingJobName="edge-packaging-name"
)
```

有关 Edge Manager API 的完整列表，请参阅Boto3 文档.

Edge Manager Agent

边缘管理器代理是边缘设备的推理引擎。使用代理程序对加载到边缘设备上的模型进行预测。代理还收集模型指标并按特定时间间隔捕获数据。示例数据存储在 Amazon S3 存储桶中。

有两种方法可以在边缘设备上安装 Edge Manager 代理并将其部署到边缘设备上：

1. 从 Amazon S3 发布存储桶中将代理作为二进制文件下载。有关更多信息，请参阅 手动下载并设置 Edge Manager 代理 (p. 2306)。
2. 使用Amazon IoT GreengrassV2 控制台或Amazon CLI部署到`aws.greengrass.SageMakerEdgeManager`. 请参阅 CreateAmazon IoT GreengrassV2 组件 (p. 2312)。

手动下载并设置 Edge Manager 代理

根据您的操作系统、体系结构和Amazon区域。代理会定期更新，因此您可以根据发布日期和版本选择代理。拥有代理后，请创建 JSON 配置文件。指定设备 IoT 事物名称、队列名称、设备凭据和其他键值对。请参阅正在运行 SageMaker Edge Manager Agent (p. 2308)获取必须在配置文件中指定的密钥的完整列表。您可以将代理作为可执行二进制文件运行，也可以将其作为动态共享对象链接。

代理程序的工作方式

代理程序在设备的 CPU 上运行。代理在编译作业期间指定的目标设备的框架和硬件上运行推理。例如，如果您为 Jetson Nano 编译了模型，则代理程序在提供的深度学习运行时(DLR)。

对于受支持的操作系统，该代理以二进制格式发布。检查您的操作系统是否受支持并符合下表中的最低操作系统要求：

Linux

版本：Ubuntu 18.04

支持的二进制格式：x86-64 位（ELF 二进制文件）和 armv8 64 位（ELF 二进制文件）

Windows

版本：Windows 10 版本 1909

支持的二进制格式：x86-32 位（DLL）和 x86-64 位（DLL）

安装边缘管理器代理

要使用 Edge Manager 代理，您必须首先获取发布工件和根证书。发布工件存储在 Amazon S3 存储桶中us-west-2区域。要下载工件，请指定操作系统 (<OS>) 和VERSION.

根据你的操作系统，替换<OS>具有以下操作之一：

32 位 Windows	Windows 64 位	Linux x86-64	Linux ARMv8
Windows x86	Windows x64	Linux-x64	linux-armv8

这些区域有：VERSION分为三个组成部分：<MAJOR_VERSION>.<YYYY-MM-DD>-<SHA-7>，其中：

- MAJOR_VERSION: 发布版本。发布版本当前设置为1.
- <YYYY-MM-DD>: 文物发布的时间戳。
- SHA-7：构建发行版的存储库提交 ID。

您必须提供MAJOR_VERSION还有时间戳<YYYY-MM-DD>格式的日期和时间。我们建议您使用最新的神器发布时间戳。使用以下命令获取最新的时间戳。

在命令行中运行以下命令以获取最新的时间戳。Replace<OS>使用操作系统：

```
aws s3 ls s3://sagemaker-edge-release-store-us-west-2-<OS>/Releases/ | sort -r
```

例如，如果你有 Windows 32 位操作系统，请运行：

```
aws s3 ls s3://sagemaker-edge-release-store-us-west-2-windows-x86/Releases/ | sort -r
```

返回：

```
2020-12-01 23:33:36 0

                   PRE 1.20201218.81f481f/
                   PRE 1.20201207.02d0e97/
```

此示例中的返回输出显示了两个发布工件。第一个发布工件文件指出，发布版本有一个主要发布版本1，一个时间戳20201218（格<YYYY-MM-DD>式）和 a81f481fSHA-7 提交 ID。

> **Note**
>
> 前面的命令假定你已经配置了Amazon Command Line Interface. 有关如何配置设置的更多信息，请参阅Amazon CLI用来交互Amazon，请参阅配置AmazonCLI.

根据您的操作系统，使用以下命令安装工件：

Windows 32-bit

```
aws s3 cp s3://sagemaker-edge-release-store-us-west-2-windows-x86/
Releases/<VERSION>/<VERSION>.zip .
aws s3 cp s3://sagemaker-edge-release-store-us-west-2-windows-x86/Releases/<VERSION>/
sha256_hex.shasum .
```

Windows 64-bit

```
aws s3 cp s3://sagemaker-edge-release-store-us-west-2-windows-x64/
Releases/<VERSION>/<VERSION>.zip .
```

```
aws s3 cp s3://sagemaker-edge-release-store-us-west-2-windows-x64/Releases/<VERSION>/
sha256_hex.shasum .
```

Linux x86-64

```
aws s3 cp s3://sagemaker-edge-release-store-us-west-2-linux-x64/
Releases/<VERSION>/<VERSION>.tgz .
aws s3 cp s3://sagemaker-edge-release-store-us-west-2-linux-x64/Releases/<VERSION>/
sha256_hex.shasum .
```

Linux ARMv8

```
aws s3 cp s3://sagemaker-edge-release-store-us-west-2-linux-armv8/
Releases/<VERSION>/<VERSION>.tgz .
aws s3 cp s3://sagemaker-edge-release-store-us-west-2-linux-armv8/Releases/<VERSION>/
sha256_hex.shasum .
```

您还必须下载根证书。此证书验证由签名的模型工件Amazon然后再将它们加载到边缘设备上。

Replace<OS>从受支持的操作系统列表中与您的平台对应，然后替换<REGION>与您的Amazon区域。

```
aws s3 cp s3://sagemaker-edge-release-store-us-west-2-<OS>/
Certificates/<REGION>/<REGION>.pem .
```

正在运行 SageMaker Edge Manager Agent

您可以运行 SageMaker Edge Manager 代理可以作为一个独立的进程，采用可执行文件和可链接格式 (ELF) 可执行二进程，也可以将其作为动态共享对象 (.dll) 链接。作为独立的可执行二进制文件运行是首选模式，并且在 Linux 上受支持。Windows 支持作为共享对象 (.dll) 运行。

在 Linux 上，我们建议您通过作为初始化一部分的服务来运行二进制文件（`init`) 系统。如果要直接运行二进制文件，则可以在终端节点执行此操作，如以下示例所示。如果您有现代操作系统，则在运行代理之前不需要其他安装，因为所有要求都是静态内置在可执行文件中。这使您可以灵活地在终端、作为服务或在容器内运行代理。

要运行代理，首先创建 JSON 配置文件。指定以下键/值对：

- `sagemaker_edge_core_device_name`：设备名称。此设备名称需要与设备队列一起注册 SageMaker Edge Manager 控制台。
- `sagemaker_edge_core_device_fleet_name`：设备所属的队列的名称。
- `sagemaker_edge_core_region`：Amazon与设备、队列和 Amazon S3 存储桶关联的区域。这对应于注册设备的区域和 Amazon S3 存储桶的创建地区（预计它们将是相同的）。模型本身可以用来编译 SageMaker Neo 在另一个区域，这种配置与模型编译区域无关。
- `sagemaker_edge_core_root_certs_path`：根证书的绝对文件夹路径。这用于使用相关的设备验证Amazonaccount.
- `sagemaker_edge_provider_aws_ca_cert_file`：获取亚马逊根 CA 证书（AmazonrootCA1.peM）的绝对路径。这用于使用相关的设备验证Amazonaccount.`AmazonCA`是由所有的证书Amazon.
- `sagemaker_edge_provider_aws_cert_file`: 通往的绝对路径Amazon IoT签署根证书 (`*.pem.crt`)。
- `sagemaker_edge_provider_aws_cert_pk_file`: 通往的绝对路径Amazon IoT私有密钥。 (`*.pem.key`)。
- `sagemaker_edge_provider_aws_iot_cred_endpoint`: 该Amazon IoT凭据终端节点 (*###*.iot。*##*.amazonaws.com)。此终端节点用于凭据验证。请参阅将设备连接到Amazon IoT有关更多信息。
- `sagemaker_edge_provider_provider`：这表示正在使用的提供程序接口的实现。提供商界面与终端网络服务进行通信，以进行上传、检测信号和注册验证。默认情况下，该选项设置为 `"Aws"`。我们允许自

定义实现提供商界面。可以设置为None对于没有提供商或Custom用于提供了相关共享对象路径的自定义实现。

- sagemaker_edge_provider_provider_path：提供提供程序实现共享对象的绝对路径。（.so 或 .dll 文件）。这些区域有："Aws"提供商 dll/so 文件随代理版本一起提供。此字段为必填字段。
- sagemaker_edge_provider_s3_bucket_name：Amazon S3 存储桶的名称（不是 Amazon S3 存储桶 URI）。存储桶必须有sagemaker名称中的字符串。
- sagemaker_edge_log_verbose（布尔值值。）: 可选。设置调试日志。选择 True 或 False。
- sagemaker_edge_telemetry_libsystemd_path：仅适用于 Linux，systemd实现代理崩溃计数器指标。设置 libsystemd 的绝对路径以启用崩溃计数器指标。可以通过运行来找到默认的 libsystemd 路径"whereis libsystemd"在设备终端中。
- sagemaker_edge_core_capture_data_destination：上传捕获数据的目的地。选择 Cloud 或 Disk。默认设置为"Disk". 设置为"Disk"将在首选位置将输入/输出张量和辅助数据写入本地文件系统。当向写入时"Cloud"使用中提供的 Amazon S3 存储桶名称sagemaker_edge_provider_s3_bucket_name配置。
- sagemaker_edge_core_capture_data_disk_path：在本地文件系统中设置绝对路径，捕获数据文件将写入该路径"Disk"目标是目标。在以下情况下不使用此字段"Cloud"指定为目的地。
- sagemaker_edge_core_folder_prefix：指定时，Amazon S3 中的父前缀存储捕获的数据"Cloud"作为捕获数据目标 (sagemaker_edge_core_capture_data_disk_path). 捕获的数据存储在下面的子文件夹中sagemaker_edge_core_capture_data_disk_path如果"Disk"被设置为数据目标。
- sagemaker_edge_core_capture_data_buffer_size（整数值）：捕获数据循环缓冲区大小。它表示缓冲区中存储的最大请求数。
- sagemaker_edge_core_capture_data_batch_size（整数值）：捕获数据批处理大小。它表示从缓冲区处理的一批请求的大小。此值必须低于sagemaker_edge_core_capture_data_buffer_size. 对于批量大小，建议最多为缓冲区的一半大小。
- sagemaker_edge_core_capture_data_push_period_seconds（整数值）：以秒为单位的捕获数据推送周期。当缓冲区中存在批量大小请求或此时段达到时间段（以先到者为准）时，将处理缓冲区中的一批请求。此配置设置了该时间段。
- sagemaker_edge_core_capture_data_base64_embed_limit：上传捕获数据的限制（以字节为单位）。整数值。

您的配置文件应类似于以下内容（指定了您的具体值）。在此示例中，默认值Amazon（提供方"Aws"）已使用且未指定定期上传。

```
{
    "sagemaker_edge_core_device_name": "device-name",
    "sagemaker_edge_core_device_fleet_name": "fleet-name",
    "sagemaker_edge_core_region": "region",
    "sagemaker_edge_core_root_certs_path": "<Absolute path to root certificates>",
    "sagemaker_edge_provider_provider": "Aws",
    "sagemaker_edge_provider_provider_path" : "/path/to/libprovider-aws.so",
    "sagemaker_edge_provider_aws_ca_cert_file": "<Absolute path to Amazon Root CA
 certificate>/AmazonRootCA1.pem",
    "sagemaker_edge_provider_aws_cert_file": "<Absolute path to Amazon IoT signing root
 certificate>/device.pem.crt",
    "sagemaker_edge_provider_aws_cert_pk_file": "<Absolute path to Amazon IoT private
 key.>/private.pem.key",
    "sagemaker_edge_provider_aws_iot_cred_endpoint": "https://<Amazon IoT Endpoint
 Address>",
    "sagemaker_edge_core_capture_data_destination": "Cloud",
    "sagemaker_edge_provider_s3_bucket_name": "sagemaker-bucket-name",
    "sagemaker_edge_core_folder_prefix": "Amazon S3 folder prefix",
    "sagemaker_edge_core_capture_data_buffer_size": 30,
    "sagemaker_edge_core_capture_data_batch_size": 10,
    "sagemaker_edge_core_capture_data_push_period_seconds": 4000,
```

```
    "sagemaker_edge_core_capture_data_base64_embed_limit": 2,
    "sagemaker_edge_log_verbose": false
}
```

发布工件中包含一个名为的二进制可执行文件`sagemaker_edge_agent_binary`中的`/bin`目录。要运行二进制文件，请使用`-a`标记以在您选择的目录中创建套接字文件描述符 (.sock)，并指定使用`-c`旗帜。

```
./sagemaker_edge_agent_binary -a <ADDRESS_TO_SOCKET> -c <PATH_TO_CONFIG_FILE>
```

例如：

```
./sagemaker_edge_agent_binary -a /tmp/sagemaker_edge_agent_example.sock -c
 sagemaker_edge_config.json
```

在此示例中，名为的套接字文件描述符`sagemaker_edge_agent_example.sock`是在`/tmp`目录并指向与名为的代理位于同一工作目录中的配置文件`sagemaker_edge_config.json`.

使用部署模型 Package 和边缘管理器代理AmazonIoT Greengrass

SageMaker 边缘管理器集成Amazon IoT Greengrass版本 2 用于简化访问、维护和将 Edge Manager 代理和模型部署到设备的过程。没有Amazon IoT GreengrassV2，将设备和队列设置为使用 SageMaker 边缘管理器要求您从 Amazon S3 发布存储桶手动复制边缘管理器代理。您可以使用代理对加载到边缘设备的模型进行预测。与Amazon IoT GreengrassV2 和 SageMaker 边缘管理器集成，你现在可以使用Amazon IoT GreengrassV2 组件。组件是预先构建的软件模块，可以将边缘设备连接到Amazon服务或第三方服务Amazon IoT Greengrass.

你必须安装Amazon IoT Greengrass如果你想使用核心软件到你的设备上Amazon IoT GreengrassV2 来部署SageMaker 边缘管理器代理和你的模型。有关设备要求以及如何设置设备的更多信息，请参阅设置Amazon IoT Greengrass核心设备中的Amazon IoT Greengrass文档中）。

您可以为 Edge Manager 应用程序使用预构建的组件、自动生成的组件和自定义组件。公有 SageMaker 边缘管理器组件由 SageMaker 维护。当您将机器学习模型与`CreateEdgePackagingJob`API 并指定`"GreengrassV2Component"`对于边缘管理器 API 字段`PresetDeploymentType`. 第三个组件是负责在设备上进行预处理和推断的推理应用程序。您必须创建此组件。或者请参阅??? (p. 2313)中的 SageMaker 边缘管理器文档或创建自定义Amazon IoT Greengrass组件中的Amazon IoT Greengrass有关如何创建自定义组件的更多信息。

先决条件

SageMaker 边缘管理器使用Amazon IoT GreengrassV2，使用组件简化 Edge Manager 代理、机器学习模型和推理应用程序到设备的部署。为了更轻松地维护Amazon边缘管理器允许您重用现有的 IAM 角色Amazon IoT角色别名。如果您还没有，边缘管理器将生成角色别名作为边缘管理器打包作业的一部分。您不再需要关联从 SageMaker 边缘管理器打包作业Amazon IoT角色。

在开始之前，您必须：

1. 安装 Amazon IoT Greengrass Core 软件。有关详细信息，请参阅安装Amazon IoT GreengrassCore 软件.
2. 设置Amazon IoT GreengrassV2。有关更多信息，请参阅 。安装Amazon IoT Greengrass具有手动资源配置的核心软件.

   Note

   - 确保Amazon IoT事物名称全部为小写字母，除了（可选）破折号外不包含字符 (#)。
   - IAM 角色必须以`SageMaker*`
3. 将以下权限和内联策略附加到期间创建的 IAM 角色Amazon IoT GreengrassV2 设置。
   - 导航到 IAM 控制台https://console.aws.amazon.com/iam/.

- 通过在中键入角色名称来搜索您创建的角色搜索字段中返回的子位置类型。
- 选择你的角色。
- 下一步，选择附加策略.
- 搜索AmazonSageMakerEdgeDeviceFleetPolicy.
- SelectAmazonSageMakerFullAccess（这是一个可选步骤，使您可以更轻松地在模型编译和打包中重复使用此 IAM 角色）。
- 选择添加内联策略。

```
{
    "Version":"2012-10-17",
    "Statement":[
      {
        "Sid":"GreengrassComponentAccess",
        "Effect":"Allow",
        "Action":[
            "greengrass:CreateComponentVersion",
            "greengrass:DescribeComponent"
        ],
        "Resource":"*"
       }
    ]
}
```

- 单击附加策略.
- Select信任关系.
- 单击编辑信任关系.
- 将内容替换为以下内容。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

4. 创建 Edge Manager 设备队列。有关如何创建队列的信息，请参阅。设置设备和队列 (p. 2294).
5. 使用与您的同名注册设备Amazon IoT在Amazon IoT GreengrassV2 设置。
6. 创建至少一个自定义私有Amazon IoT Greengrass组件。此组件是在设备上运行推理的应用程序。请参阅创建 Hello World 自定义组件 (p. 2313)。

   Note

   - 这些区域有： SageMaker Edge Manager 和Amazon IoT Greengrass集成只适用于Amazon IoT Greengrassv2。

- 你的两种Amazon IoT事物名称和 Edge Manager 设备名称必须相同。
- SageMaker 边缘管理器不在本地加载Amazon IoT证书然后致电Amazon IoT直接凭据提供程序终端节点。相反， SageMaker 边缘管理器使用Amazon IoT Greengrassv2 TokenExchangeService 它从 TES 终端节点获取临时凭证。

CreateAmazon IoT GreengrassV2 组件

Amazon IoT Greengrass使用组件，一个软件模块，部署到并在Amazon IoT Greengrass核心设备。你将需要（至少）三个组件：

1. 公共边缘管理器代理Amazon IoT Greengrass组件。
2. 将机器学习模型打包时自动生成的模型组件Amazon SDK for Python (Boto3)API 或使用 SageMaker 控制台。
3. 推理应用程序的私有自定义组件。

Edge Manager AgentAmazon IoT Greengrass组件 (1) 部署边缘管理器代理二进制文件。

当您使用Amazon SDK for Python (Boto3)API 或 SageMaker 控制台。有关如何生成模型组件的信息，请参阅。自动生成的组件 (p. 2312).

私有自定义组件 (3) 是用于实现 Edge Manager Agent 客户端应用程序以及对推理结果进行任何预处理和后处理的应用程序。有关如何创建自定义组件的详细信息，请参阅。自动生成的组件 (p. 2312)要么创建自定义Amazon IoT Greengrass组件.

自动生成的组件

使用生成模型组件`CreateEdgePackagingJob`API 并指定`GreengrassV2Component`(对于) SageMaker 边缘管理器打包作业 API 字段`PresetDeploymentType`. 当你打电话给`CreateEdgePackagingJob`API、边缘管理器将您的 SageMaker Neo 编译模型带到 Amazon S3 中并创建模型组件。模型组件将自动存储在您的账户中。您可以通过导航到Amazon IoT控制台https://console.aws.amazon.com/iot/. SelectGreengrass然后选择核心实例设备。该页面有一个列表Amazon IoT Greengrass与您的账户关联的核心设备。如果未在中指定模型组件名称`PresetDeploymentConfig`，生成的默认名称包括`"SagemakerEdgeManager"`和你的名称 SageMaker 边缘管理器打包作业。以下示例演示如何指定 Edge Manager 以创建Amazon IoT GreengrassV2 组件与`CreateEdgePackagingJob`API。

```
import sagemaker
import boto3

# Create a SageMaker client object to make it easier to interact with other AWS services.
sagemaker_client = boto3.client('sagemaker', region=<YOUR_REGION>)

# Replace with your IAM Role ARN
sagemaker_role_arn = "arn:aws:iam::<account>:role/*"

# Replace string with the name of your already created S3 bucket.
bucket = 'edge-manager-demo-bucket'

# Specify a name for your edge packaging job.
edge_packaging_name = "edge_packag_job_demo"

# Replace the following string with the name you used for the SageMaker Neo compilation
 job.
compilation_job_name = "getting-started-demo"

# The name of the model and the model version.
model_name = "sample-model"
model_version = "1.1"
```

```
# Output directory in S3 where you want to store the packaged model.
packaging_output_dir = 'packaged_models'
packaging_s3_output = 's3://{}/{}'.format(bucket, packaging_output_dir)

# The name you want your Greengrass component to have.
component_name = "SagemakerEdgeManager" + edge_packaging_name

sagemaker_client.create_edge_packaging_job(
                    EdgePackagingJobName=edge_packaging_name,
                    CompilationJobName=compilation_job_name,
                    RoleArn=sagemaker_role_arn,
                    ModelName=model_name,
                    ModelVersion=model_version,
                    OutputConfig={
                        "S3OutputLocation": packaging_s3_output,
                        "PresetDeploymentType":"GreengrassV2Component",
                        "PresetDeploymentConfig":"{\"ComponentName\":\"sample-component-
name\", \"ComponentVersion\":\"1.0.2\"}"
                        }
                    )
```

您也可以使用创建自动生成的组件。 SageMaker 控制台。按照中的步骤 1-6 操作Package 模型（Amazon SageMaker 控制台）(p. 2301)

输入要存储打包作业输出和可选加密密钥的 Amazon S3 存储桶 URI。

完成以下操作以创建模型组件：

1. 选择预设部署.
2. 指定组件名称组件名称字段中返回的子位置类型。
3. （可选）提供组件、组件版本、平台操作系统或平台体系结构的说明组件描述、组件版本、平台 OS, 和平台架构，。
4. 选择 Submit（提交）。

创建 Hello World 自定义组件

自定义应用程序组件用于在边缘设备上执行推理。该组件负责将模型加载到 SageMaker 边缘管理器，调用 Edge Manager 代理进行推理，然后在组件关闭时卸载模型。在创建组件之前，请确保代理和应用程序可以与 SageMaker Edge Manager。要执行此操作，请配置gRPC. 这些区域有： SageMaker Edge Manager 代理使用 Protobuf 缓冲区和 gRPC 服务器中定义的方法来与边缘设备和云上的客户端应用程序建立通信。

要使用 GrPC，您必须：

1. 使用从 Amazon S3 发行存储桶下载 Edge Manager 代理时提供的 .proto 文件创建 gRPC 存根。
2. 用你喜欢的语言编写客户端代码。

不需要在 .proto 文件 (1) 中定义服务。下载时，服务 .proto 文件包含在压缩的 TAR 文件中 SageMaker Edge Manager 代理从 Amazon S3 发行存储桶中发布二进制文件。

在主机上安装 gRPC 和其他必要的工具并创建 gRPC 存根`agent_pb2_grpc.py`和`agent_pb2.py`在 Python 中。确保你有`agent.proto`在本地目录中。

```
%%bash
pip install grpcio
pip install grpcio-tools
python3 -m grpc_tools.protoc --proto_path=. --python_out=. --grpc_python_out=. agent.proto
```

前面的代码根据 .proto 服务定义 (2) 生成 gRPC 客户端和服务器接口。换句话说，它在 Python 中创建 GrPC 模型。API 文件夹包含用于与代理进行通信的 Protobuf 规范。

接下来，使用 gRPC API 为您的服务编写客户端和服务器 (2)。以下示例脚本，`edge_manager_python_example.py`，使用 Python 加载、列出和卸载`yolov3`模型到边缘设备。

```
import grpc
from PIL import Image
import agent_pb2
import agent_pb2_grpc
import os


model_path = '<PATH-TO-SagemakerEdgeManager-COMPONENT>'

agent_socket = 'unix:///tmp/aws.greengrass.SageMakerEdgeManager.sock'

agent_channel = grpc.insecure_channel(agent_socket, options=(('grpc.enable_http_proxy',
 0),))

agent_client = agent_pb2_grpc.AgentStub(agent_channel)


def list_models():
    return agent_client.ListModels(agent_pb2.ListModelsRequest())


def list_model_tensors(models):
    return {
        model.name: {
            'inputs': model.input_tensor_metadatas,
            'outputs': model.output_tensor_metadatas
        }
        for model in list_models().models
    }


def load_model(model_name, model_path):
    load_request = agent_pb2.LoadModelRequest()
    load_request.url = model_path
    load_request.name = model_name
    return agent_client.LoadModel(load_request)


def unload_model(name):
    unload_request = agent_pb2.UnLoadModelRequest()
    unload_request.name = name
    return agent_client.UnLoadModel(unload_request)


def predict_image(model_name, image_path):
    image_tensor = agent_pb2.Tensor()
    image_tensor.byte_data = Image.open(image_path).tobytes()
    image_tensor_metadata = list_model_tensors(list_models())[model_name]['inputs'][0]
    image_tensor.tensor_metadata.name = image_tensor_metadata.name
    image_tensor.tensor_metadata.data_type = image_tensor_metadata.data_type
    for shape in image_tensor_metadata.shape:
        image_tensor.tensor_metadata.shape.append(shape)
    predict_request = agent_pb2.PredictRequest()
    predict_request.name = model_name
    predict_request.tensors.append(image_tensor)
    predict_response = agent_client.Predict(predict_request)
    return predict_response

def main():
```

```
    try:
        unload_model('your-model')
    except:
        pass

    print('LoadModel...', end='')
    try:
        load_model('your-model', model_path)
        print('done.')
    except Exception as e:
        print()
        print(e)
        print('Model already loaded!')

    print('ListModels...', end='')
    try:
        print(list_models())
        print('done.')

    except Exception as e:
        print()
        print(e)
        print('List model failed!')

    print('Unload model...', end='')
    try:
        unload_model('your-model')
        print('done.')
    except Exception as e:
        print()
        print(e)
        print('unload model failed!')

if __name__ == '__main__':
    main()
```

确保`model_path`指向的名字Amazon IoT Greengrass如果您使用相同的客户端代码示例，则包含模型的组件。

您可以创建Amazon IoT GreengrassV2 Hello World 组件一旦你生成了 gRPC 存根并准备好了你的 Hello World 代码。为此，请执行以下操作：

- 上传您的`edge_manager_python_example.py`、`agent_pb2_grpc.py`, 和`agent_pb2.py`将其记下 Amazon S3 存储桶，然后记下其 Amazon S3 路径。
- 在中创建私有组件Amazon IoT GreengrassV2 控制台并定义组件的配方。在以下配方中为您的 Hello World 应用程序和 gRPC 存根指定 Amazon S3 URI。

```
---
RecipeFormatVersion: 2020-01-25
ComponentName: com.sagemaker.edgePythonExample
ComponentVersion: 1.0.0
ComponentDescription: Sagemaker Edge Manager Python example
ComponentPublisher: Amazon Web Services, Inc.
ComponentDependencies:
  aws.greengrass.SageMakerEdgeManager:
    VersionRequirement: '>=1.0.0'
    DependencyType: HARD
Manifests:
  - Platform:
      os: linux
      architecture: "/amd64|x86/"
    Lifecycle:
      install: |-
```

```
      apt-get install python3-pip
      pip3 install grpcio
      pip3 install grpcio-tools
      pip3 install protobuf
      pip3 install Pillow
    run:
      script: |-
        python3 {{artifacts:path}}/edge_manager_python_example.py
  Artifacts:
    - URI: <code-s3-path>
    - URI: <pb2-s3-path>
    - URI: <pb2-grpc-s3-path>
```

有关创建 Hello World 食谱的详细信息，请参阅创建您的第一个组件中的Amazon IoT Greengrass文档中）。

将组件部署到设备

使用Amazon IoT控制台或使用Amazon CLI.

部署组件（控制台）

部署您的Amazon IoT Greengrass组件使用Amazon IoT控制台。

1. 在Amazon IoT Greengrass控制台https://console.aws.amazon.com/iot/选择导航菜单部署.
2. 在存储库的组件页面上，公共组件选项卡上，选择aws.greengrass.SageMakerEdgeManager.
3. 在存储库的aws.greengrass.SageMakerEdgeManager页面上，选择部署.
4. 从Add to deployment，选择以下选项之一：
   a. 要将此组件合并到目标设备上的现有部署，请选择添加到现有部署，然后选择要修改的部署。
   b. 要在目标设备上创建新部署，请选择创建新部署. 如果您的设备上有现有部署，则选择此步骤将替换现有部署。
5. 在存储库的指定目标页面，请执行以下操作：
   a. UDER部署信息中，输入或修改部署的友好名称。
   b. UDER部署目标，选择部署的目标，然后选择下一步. 如果要修改现有部署，则无法更改部署目标。
6. 在存储库的选择组件页面，下我的组件，选择：
   - COM。*<CUSTOM-COMPONENT-NAME>*
   - aws.greengrass.SageMakerEdgeManager
   - SageMakeredgemManager。*<YOUR-PACKAGING-JOB>*
7. 在存储库的配置组件页面上，选择com.GreenGrass.SageMakeredgemManager，并执行以下操作。
   a. 选择配置组件.
   b. UDER配置更新，在要合并的配置中，输入以下配置。

      ```
      {
          "DeviceFleetName": "device-fleet-name",
          "BucketName": "DOC-EXAMPLE-BUCKET"
      }
      ```

      Replace*device-fleet-name*将替换为您创建的边缘设备队列的名称，然后替换*DOC-EXAMPLE-BUCKET*将替换为与您的设备队列关联的 Amazon S3 存储桶的名称。
   c. 选择确认，然后选择下一步.
8. 在存储库的配置高级设置页面上，保留默认配置设置，然后选择下一步.
9. 在 Review（检查）页上，选择 Deploy（部署）。

部署组件 (Amazon CLI)

1. 创建 `deployment.json`文件来定义您的部署配置 SageMaker 边缘管理器组件。此文件应类似以下示例：

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.SageMakerEdgeManager": {
      "componentVersion": 1.0.0,
      "configurationUpdate": {
        "merge": {
          "DeviceFleetName": "device-fleet-name",
          "BucketName": "DOC-EXAMPLE-BUCKET"
        }
      }
    },
    "com.greengrass.SageMakerEdgeManager.ImageClassification": {
      "componentVersion": 1.0.0,
      "configurationUpdate": {
      }
    },
    "com.greengrass.SageMakerEdgeManager.ImageClassification.Model": {
      "componentVersion": 1.0.0,
      "configurationUpdate": {
      }
    },
  }
}
```

   - 在`targetArn`字段中，替换*`targetArn`*将部署作为目标事物或事物组的 Amazon 资源名称 (ARN)，格式如下：
     - 事物：`arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - 事物组：`arn:aws:iot:`*`region`*`:`*`account-id:`*`thinggroup/`*`thingGroupName`*
   - 在`merge`字段中，替换*`device-fleet-name`*将替换为您创建的边缘设备队列的名称，然后替换*`DOC-EXAMPLE-BUCKET`*将显示与设备队列关联的 Amazon S3 存储桶的名称。
   - 将每个组件的组件版本替换为最新的可用版本。
2. 运行以下命令以在设备上部署组件：

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

完成部署可能需要几分钟时间。在下一步中，检查组件日志以验证部署是否成功完成并查看推理结果。

有关将组件部署到单个设备或设备组的更多信息，请参阅部署Amazon IoTGreengrass组件到设备。

管理模型

Edge Manager 代理可以一次加载多个模型，并推断边缘设备上加载的模型。代理程序可加载的型号数取决于设备上的可用内存。代理验证模型签名并将边缘打包作业生成的所有工件加载到内存中。此步骤要求前面步骤中描述的所有必需证书与其余二进制安装一起安装。如果无法验证模型的签名，则加载模型将失败，并显示适当的返回代码和原因。

SageMaker Edge Manager 代理提供了在边缘设备上实现控制平面和数据平面 API 的模型管理 API 的列表。除了本文档之外，我们建议浏览示例客户端实现，其中显示了下面描述的 API 的规范使用情况。

这些区域有：`proto`文件可作为发布工件的一部分（在发布 tarball 中）。在本文档中，我们列出并描述本中列出的 API 的使用情况`proto`文件。

Note

还有 one-to-one Windows 版本上的这些 API 的映射以及 C# 中应用程序实现的示例代码将与 Windows 的发布工件共享。以下说明用于将 Agent 作为独立进程运行，适用于 Linux 的发布工件。

根据操作系统提取存档。其中VERSION分为三个组成部分：<MAJOR_VERSION>.<YYYY-MM-DD>-<SHA-7>. 请参阅安装边缘管理器代理 (p. 2307)有关如何获取发布版本的信息（<MAJOR_VERSION>）、发布工件的时间戳 (<YYYY-MM-DD>) 和存储库提交 ID (SHA-7)

Linux

可以使用以下命令提取 zip 存档：

```
tar -xvzf <VERSION>.tgz
```

Windows

zip 存档可以使用 UI 或命令提取：

```
unzip <VERSION>.tgz
```

发布工件层次结构（在提取tar/zip存档）如下所示。代理数proto文件可在api/.

```
0.20201205.7ee4b0b
### bin
#         ### sagemaker_edge_agent_binary
#         ### sagemaker_edge_agent_client_example
### docs
### api
#         ### agent.proto
### attributions
#         ### agent.txt
#         ### core.txt
### examples
### ipc_example
### CMakeLists.txt
### sagemaker_edge_client.cc
### sagemaker_edge_client_example.cc
### sagemaker_edge_client.hh
### sagemaker_edge.proto
### README.md
### shm.cc
### shm.hh
### street_small.bmp
```

主题

- 加载模型 (p. 2319)
- 卸载模型 (p. 2320)
- 列出模型 (p. 2320)
- 描述模型 (p. 2321)
- 捕获数据 (p. 2321)
- 获取捕获状态 (p. 2323)
- 预测 (p. 2323)

加载模型

Edge Manager 代理支持加载多个模型。此 API 验证模型签名，并将`EdgePackagingJob`operation. 此步骤要求安装所有必需的证书以及代理二进制安装的其余部分。如果无法验证模型的签名，则此步骤将失败，并在日志中显示适当的返回代码和错误消息。

```
// perform load for a model
// Note:
// 1. currently only local filesystem paths are supported for loading models.
// 2. multiple models can be loaded at the same time, as limited by available device memory
// 3. users are required to unload any loaded model to load another model.
// Status Codes:
// 1. OK - load is successful
// 2. UNKNOWN - unknown error has occurred
// 3. INTERNAL - an internal error has occurred
// 4. NOT_FOUND - model doesn't exist at the url
// 5. ALREADY_EXISTS - model with the same name is already loaded
// 6. RESOURCE_EXHAUSTED - memory is not available to load the model
// 7. FAILED_PRECONDITION - model is not compiled for the machine.
//
rpc LoadModel(LoadModelRequest) returns (LoadModelResponse);
```

Input

```
//
// request for LoadModel rpc call
//
message LoadModelRequest {
  string url = 1;
  string name = 2;  // Model name needs to match regex "^[a-zA-Z0-9](-*[a-zA-Z0-9])*$"
}
```

Output

```
//
//
// response for LoadModel rpc call
//
message LoadModelResponse {
  Model model = 1;
}

//
// Model represents the metadata of a model
//  url - url representing the path of the model
//  name - name of model
//  input_tensor_metadatas - TensorMetadata array for the input tensors
//  output_tensor_metadatas - TensorMetadata array for the output tensors
//
// Note:
//  1. input and output tensor metadata could empty for dynamic models.
//
message Model {
  string url = 1;
  string name = 2;
  repeated TensorMetadata input_tensor_metadatas = 3;
  repeated TensorMetadata output_tensor_metadatas = 4;
}
```

卸载模型

卸载之前加载的模型。它是通过在期间提供的模型别名进行识别的`loadModel`. 如果找不到别名或未加载模型，则返回错误。

```
//
// perform unload for a model
// Status Codes:
// 1. OK - unload is successful
// 2. UNKNOWN - unknown error has occurred
// 3. INTERNAL - an internal error has occurred
// 4. NOT_FOUND - model doesn't exist
//
rpc UnLoadModel(UnLoadModelRequest) returns (UnLoadModelResponse);
```

Input

```
//
// request for UnLoadModel rpc call
//
message UnLoadModelRequest {
 string name = 1; // Model name needs to match regex "^[a-zA-Z0-9](-*[a-zA-Z0-9])*$"
}
```

Output

```
//
// response for UnLoadModel rpc call
//
message UnLoadModelResponse {}
```

列出模型

列出所有加载的模型及其别名。

```
//
// lists the loaded models
// Status Codes:
// 1. OK - unload is successful
// 2. UNKNOWN - unknown error has occurred
// 3. INTERNAL - an internal error has occurred
//
rpc ListModels(ListModelsRequest) returns (ListModelsResponse);
```

Input

```
//
// request for ListModels rpc call
//
message ListModelsRequest {}
```

Output

```
//
// response for ListModels rpc call
//
```

```
message ListModelsResponse {
 repeated Model models = 1;
}
```

描述模型

描述在代理上加载的模型。

```
//
// Status Codes:
// 1. OK - load is successful
// 2. UNKNOWN - unknown error has occurred
// 3. INTERNAL - an internal error has occurred
// 4. NOT_FOUND - model doesn't exist at the url
//
rpc DescribeModel(DescribeModelRequest) returns (DescribeModelResponse);
```

Input

```
//
// request for DescribeModel rpc call
//
message DescribeModelRequest {
  string name = 1;
}
```

Output

```
//
// response for DescribeModel rpc call
//
message DescribeModelResponse {
  Model model = 1;
}
```

捕获数据

允许客户端应用程序捕获 Amazon S3 存储桶中的输入和输出张量，也可以选择捕获辅助。预计客户端应用程序将在每次调用此 API 时传递唯一的捕获 ID。以后可以用来查询捕获的状态。

```
//
// allows users to capture input and output tensors along with auxiliary data.
// Status Codes:
// 1. OK - data capture successfully initiated
// 2. UNKNOWN - unknown error has occurred
// 3. INTERNAL - an internal error has occurred
// 5. ALREADY_EXISTS - capture initiated for the given capture_id
// 6. RESOURCE_EXHAUSTED - buffer is full cannot accept any more requests.
// 7. OUT_OF_RANGE - timestamp is in the future.
// 8. INVALID_ARGUMENT - capture_id is not of expected format.
//
rpc CaptureData(CaptureDataRequest) returns (CaptureDataResponse);
```

Input

```
enum Encoding {
```

```
 CSV = 0;
 JSON = 1;
 NONE = 2;
 BASE64 = 3;
}

//
// AuxilaryData represents a payload of extra data to be capture along with inputs and
 outputs of inference
// encoding - supports the encoding of the data
// data - represents the data of shared memory, this could be passed in two ways:
// a. send across the raw bytes of the multi-dimensional tensor array
// b. send a SharedMemoryHandle which contains the posix shared memory segment id and
// offset in bytes to location of multi-dimensional tensor array.
//
message AuxilaryData {
 string name = 1;
 Encoding encoding = 2;
 oneof data {
 bytes byte_data = 3;
 SharedMemoryHandle shared_memory_handle = 4;
 }
}

//
// Tensor represents a tensor, encoded as contiguous multi-dimensional array.
// tensor_metadata - represents metadata of the shared memory segment
// data_or_handle - represents the data of shared memory, this could be passed in two
 ways:
// a. send across the raw bytes of the multi-dimensional tensor array
// b. send a SharedMemoryHandle which contains the posix shared memory segment
// id and offset in bytes to location of multi-dimensional tensor array.
//
message Tensor {
 TensorMetadata tensor_metadata = 1; //optional in the predict request
 oneof data {
 bytes byte_data = 4;
 // will only be used for input tensors
 SharedMemoryHandle shared_memory_handle = 5;
 }
}

//
// request for CaptureData rpc call
//
message CaptureDataRequest {
 string model_name = 1;
 string capture_id = 2; //uuid string
 Timestamp inference_timestamp = 3;
 repeated Tensor input_tensors = 4;
 repeated Tensor output_tensors = 5;
 repeated AuxilaryData inputs = 6;
 repeated AuxilaryData outputs = 7;
}
```

Output

```
//
// response for CaptureData rpc call
//
message CaptureDataResponse {}
```

获取捕获状态

根据加载的型号，输入和输出张量可能很大（对于许多边缘设备而言）。捕获到云可能非常耗时。因此，CaptureData()将作为异步操作实现。捕获 ID 是客户端在捕获数据调用期间提供的唯一标识符，此 ID 可用于查询异步调用的状态。

```
//
// allows users to query status of capture data operation
// Status Codes:
// 1. OK - data capture successfully initiated
// 2. UNKNOWN - unknown error has occurred
// 3. INTERNAL - an internal error has occurred
// 4. NOT_FOUND - given capture id doesn't exist.
//
rpc GetCaptureDataStatus(GetCaptureDataStatusRequest) returns
 (GetCaptureDataStatusResponse);
```

Input

```
//
// request for GetCaptureDataStatus rpc call
//
message GetCaptureDataStatusRequest {
  string capture_id = 1;
}
```

Output

```
enum CaptureDataStatus {
  FAILURE = 0;
  SUCCESS = 1;
  IN_PROGRESS = 2;
  NOT_FOUND = 3;
}

//
// response for GetCaptureDataStatus rpc call
//
message GetCaptureDataStatusResponse {
  CaptureDataStatus status = 1;
}
```

预测

这些区域有：predictAPI 对之前加载的模型执行推理。它接受直接输入神经网络的张量形式的请求。输出是模型的输出张量（或标量）。这是一个阻止性调用。

```
//
// perform inference on a model.
//
// Note:
// 1. users can chose to send the tensor data in the protobuf message or
// through a shared memory segment on a per tensor basis, the Predict
// method with handle the decode transparently.
// 2. serializing large tensors into the protobuf message can be quite expensive,
// based on our measurements it is recommended to use shared memory of
// tenors larger than 256KB.
// 3. SMEdge IPC server will not use shared memory for returning output tensors,
// i.e., the output tensor data will always send in byte form encoded
```

```
// in the tensors of PredictResponse.
// 4. currently SMEdge IPC server cannot handle concurrent predict calls, all
// these call will be serialized under the hood. this shall be addressed
// in a later release.
// Status Codes:
// 1. OK - prediction is successful
// 2. UNKNOWN - unknown error has occurred
// 3. INTERNAL - an internal error has occurred
// 4. NOT_FOUND - when model not found
// 5. INVALID_ARGUMENT - when tenors types mismatch
//
rpc Predict(PredictRequest) returns (PredictResponse);
```

Input

```
// request for Predict rpc call
//
message PredictRequest {
string name = 1;
repeated Tensor tensors = 2;
}

//
// Tensor represents a tensor, encoded as contiguous multi-dimensional array.
//    tensor_metadata - represents metadata of the shared memory segment
//    data_or_handle - represents the data of shared memory, this could be passed in
 two ways:
//                        a. send across the raw bytes of the multi-dimensional tensor
 array
//                        b. send a SharedMemoryHandle which contains the posix shared
 memory segment
//                            id and offset in bytes to location of multi-dimensional
 tensor array.
//
message Tensor {
  TensorMetadata tensor_metadata = 1; //optional in the predict request
  oneof data {
    bytes byte_data = 4;
    // will only be used for input tensors
    SharedMemoryHandle shared_memory_handle = 5;
  }
}

//
// Tensor represents a tensor, encoded as contiguous multi-dimensional array.
//    tensor_metadata - represents metadata of the shared memory segment
//    data_or_handle - represents the data of shared memory, this could be passed in
 two ways:
//                        a. send across the raw bytes of the multi-dimensional tensor
 array
//                        b. send a SharedMemoryHandle which contains the posix shared
 memory segment
//                            id and offset in bytes to location of multi-dimensional
 tensor array.
//
message Tensor {
  TensorMetadata tensor_metadata = 1; //optional in the predict request
  oneof data {
    bytes byte_data = 4;
    // will only be used for input tensors
    SharedMemoryHandle shared_memory_handle = 5;
  }
}
```

```
//
// TensorMetadata represents the metadata for a tensor
//    name - name of the tensor
//    data_type  - data type of the tensor
//    shape - array of dimensions of the tensor
//
message TensorMetadata {
  string name = 1;
  DataType data_type = 2;
  repeated int32 shape = 3;
}

//
// SharedMemoryHandle represents a posix shared memory segment
//    offset - offset in bytes from the start of the shared memory segment.
//    segment_id - shared memory segment id corresponding to the posix shared memory
 segment.
//    size - size in bytes of shared memory segment to use from the offset position.
//
message SharedMemoryHandle {
  uint64 size = 1;
  uint64 offset = 2;
  uint64 segment_id = 3;
}
```

Output

Note

这些区域有：`PredictResponse`只有退货`Tensors`而不是`SharedMemoryHandle`.

```
// response for Predict rpc call
//
message PredictResponse {
   repeated Tensor tensors = 1;
}
```

使用 Neo 优化模型性能

Neo 是 Amazon SageMaker 的一项功能，使机器学习模型训练一次即可在云和边缘站点中的任何位置运行。

如果您是 SageMaker Neo 的第一个用户，我们建议您查看Edge 设备入门以获取有关如何编译和部署到边缘设备的分步说明。

什么是 SageMaker Neo？

通常，优化机器学习模型以在多个平台上进行推理是困难的，因为开发人员需要针对每个平台的特定硬件和软件配置手动优化模型。如果您想要针对指定工作负载获得最佳性能，则需要了解硬件架构、指令集、内存访问模式和输入数据形状以及其他因素。对于传统软件开发，编译器和分析器简化了流程。对于机器学习，大多数工具特定于框架或硬件。这需要您手动完成不可靠的非生产性试错流程。

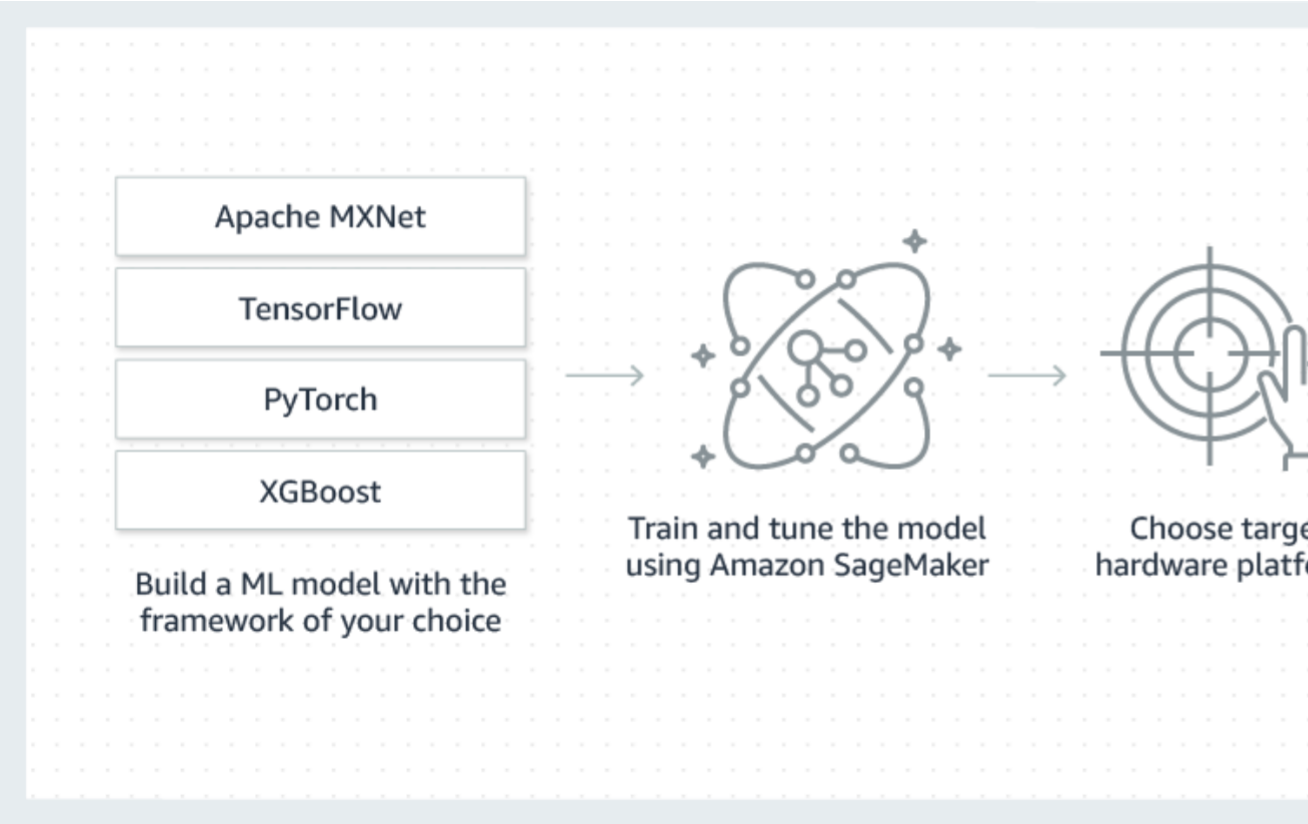
Neo 基于 Ambarella、ARM、Intel、Nvidia、NXP、Qualcomm、Texas Instruments 和 Xilinx 自动优化 Gluon、Keras、MXNet、PyTorch、TensorFlow、TensorFlow-Lite 和 ONNX 模型。Neo 已通过跨框架的 Model Zoo 中可用的计算机视觉模型进行了测试。SageMaker Neo 支持两个主要平台的编译和部署：云实例（包括 Inferentia）和边缘设备。

有关可部署到的支持框架和云实例类型的更多信息，请参阅支持的实例类型和框架构 (p. 2340)适用于云实例。

有关受支持的框架、边缘设备、操作系统、芯片架构和 SageMaker Neo 测试的边缘设备的常见机器学习模型的更多信息，请参阅支持的框架、设备、系统和架构 (p. 2365)对边缘设备的。

工作方式

Neo 包含一个编译器和一个运行时。首先，Neo 编译 API 读取从不同框架导出的模型。它将框架特定的功能和操作转换为与框架无关的中间表示形式。接着，它会执行一系列优化。然后，它为优化操作生成二进制代码，将代码写入共享对象库，然后将模型定义和参数保存到单独的文件中。Neo 还为加载和执行所编译模型的各个平台提供运行时。



您可以通过 SageMaker 控制台创建 Neo 编译作业，Amazon Command Line Interface(Amazon CLI)、Python 笔记本或 SageMaker SDK。有关如何编译模型的信息，请参阅使用 Neo 编译模型 (p. 2327). 使用几个 CLI 命令、一次 API 调用或者几次单击，您可以为所选平台转换模型。您可以将模型部署到 SageMaker 终端节点或Amazon IoT Greengrass快速设备。

Neo 可以使用 FP32 或量化为 INT8 或 FP16 位宽度的参数来优化模型。

Neo 示例笔记本

有关使用 SageMaker Neo 训练、编译、优化和部署机器学习模型以进行推理的示例笔记本，请参阅：

- 使用 MXNet 模块进行 MNIST 训练、编译和部署
- 使用 Tensorflow 模块进行 MNIST 训练、编译和部署
- 使用 SageMaker Neo 部署预训练的 PyTorch 视觉模型
- 使用图像分类示例进行模型优化
- 使用 XGBoost 示例进行模型优化

有关如何在 SageMaker 中运行这些示例笔记本的说明，请参阅示例笔记本 (p. 158). 如果您需要有关创建笔记本实例来运行这些示例的说明，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147). 要导航到笔记本实例中的相关示例，请选择Amazon SageMaker 示例选项卡来查看所有 SageMaker 示例的列表。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

使用 Neo 编译模型

本节说明了如何创建、描述、停止和列出编译作业。Amazon SageMaker Neo 中提供了以下选项，用于管理机器学习模型的编译作业：Neo CLI、Amazon SageMaker 控制台或 Amazon SageMaker 开发工具包。

主题

- 准备模型进行编译 (p. 2327)
- 编译模型 (Amazon Command Line Interface) (p. 2333)
- 编译模型 (Amazon SageMaker 控制台) (p. 2335)
- 编译模型 (Amazon SageMaker 开发工具包) (p. 2339)

准备模型进行编译

SageMaker Neo 要求机器学习模型满足特定的输入数据形状。编译所需的输入形状取决于您使用的深度学习框架。模型输入形状格式正确后，请根据以下要求保存模型。保存模型后，压缩模型伪影。

主题

- SageMaker Neo 需要什么输入数据形状？ (p. 2327)
- 为 SageMaker Neo 保存模型 (p. 2328)

SageMaker Neo 需要什么输入数据形状？

在编译模型之前，请确保模型格式正确。Neo 需要已训练模型的 JSON 格式或列表格式的预期数据输入的名称和形状。预期投入特定于框架。

以下是 SageMaker Neo 预期的输入形状：

Keras

使用已训练模型的字典格式，指定预期数据输入的名称和形状 (NCHW 格式)。请注意，虽然 Keras 模型构件应以 NHWC (最后一个频道) 格式上传，但 DataainPutConfig 应以 NCHW (通道优先) 格式指定。所需的字典格式如下：

- 对于一个输入：`{'input_1':[1,3,224,224]}`
- 对于两个输入：`{'input_1': [1,3,224,224], 'input_2':[1,3,224,224]}`

MXNet /ONNX

使用已训练模型的字典格式，指定预期数据输入的名称和形状 (NCHW 格式)。所需的字典格式如下：

- 对于一个输入：`{'data':[1,3,1024,1024]}`
- 对于两个输入：`{'var1': [1,1,28,28], 'var2':[1,1,28,28]}`

PyTorch

使用已训练模型的字典格式，指定预期数据输入的名称和形状 (NCHW 格式)。或者，您可以使用列表格式指定形状。所需的字典格式如下：

- 对于字典格式的一个输入：`{'input0':[1,3,224,224]}`
- 对于列表格式的一个输入：`[[1,3,224,224]]`
- 对于字典格式的两个输入：`{'input0':[1,3,224,224], 'input1':[1,3,224,224]}`
- 对于列表格式的两个输入：`[[1,3,224,224], [1,3,224,224]]`

TensorFlow

使用已训练模型的字典格式，指定预期数据输入的名称和形状（NHWC 格式）。所需的字典格式如下：

- 对于一个输入：`{'input':[1,1024,1024,3]}`
- 对于两个输入：`{'data1': [1,28,28,1], 'data2':[1,28,28,1]}`

tflite

使用已训练模型的字典格式，指定预期数据输入的名称和形状（NHWC 格式）。所需的字典格式如下：

- 对于一个输入：`{'input':[1,224,224,3]}`

> **Note**
>
> SageMaker Neo 仅为边缘设备目标支持 TensorFlow Lite。有关受支持的 SageMaker Neo 边缘设备目标的列表，请参阅 SageMaker Neo设备 (p. 2366)页. 有关支持的 SageMaker Neo 云实例目标列表，请参阅 SageMaker Neo支持的实例类型和框架构 (p. 2340)页.

XGBoost

不需要输入数据名称和形状。

为 SageMaker Neo 保存模型

以下代码示例显示如何保存模型以使其与 Neo 兼容。必须将模型打包为压缩 tar 文件（`*.tar.gz`）。

Keras

Keras 模型需要一个模型定义文件（`.h5`）。

为了使其与 SageMaker Neo 兼容，有两种方法可以保存 Keras 模型：

1. 导出到`.h5`使用格式`model.save("<model-name>", save_format="h5")`.
2. 冻结`SavedModel`出口后。

以下是说明如何导出`tf.keras`模型作为冻结图表（选项二）：

```
import os
import tensorflow as tf
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras import backend

tf.keras.backend.set_learning_phase(0)
model = tf.keras.applications.ResNet50(weights='imagenet', include_top=False,
 input_shape=(224, 224, 3), pooling='avg')
model.summary()

# Save as a SavedModel
export_dir = 'saved_model/'
```

```
model.save(export_dir, save_format='tf')

# Freeze saved model
input_node_names = [inp.name.split(":")[0] for inp in model.inputs]
output_node_names = [output.name.split(":")[0] for output in model.outputs]
print("Input names: ", input_node_names)
with tf.Session() as sess:
    loaded = tf.saved_model.load(sess, export_dir=export_dir, tags=["serve"])
    frozen_graph = tf.graph_util.convert_variables_to_constants(sess,
                                                                 sess.graph.as_graph_def(),
                                                                 output_node_names)
    tf.io.write_graph(graph_or_graph_def=frozen_graph, logdir=".", name="frozen_graph.pb",
 as_text=False)

import tarfile
tar = tarfile.open("frozen_graph.tar.gz", "w:gz")
tar.add("frozen_graph.pb")
tar.close()
```

Warning

不要使用SavedModel课堂使用model.save(<path>, save_format='tf'). 这种格式适合训练，但不适合推断。

MXNet

MxNet 模型必须保存为单个符号文件*-symbol.json和一个参数*.params files.

Gluon Models

使用HybridSequential类。这将以符号编程的风格运行代码（而不是命令式编程）。

```
from mxnet import nd, sym
from mxnet.gluon import nn

def get_net():
    net = nn.HybridSequential()  # Here we use the class HybridSequential.
    net.add(nn.Dense(256, activation='relu'),
            nn.Dense(128, activation='relu'),
            nn.Dense(2))
    net.initialize()
    return net

# Define an input to compute a forward calculation.
x = nd.random.normal(shape=(1, 512))
net = get_net()

# During the forward calculation, the neural network will automatically infer
# the shape of the weight parameters of all the layers based on the shape of
# the input.
net(x)

# hybridize model
net.hybridize()
net(x)

# export model
net.export('<model_name>') # this will create model-symbol.json and model-0000.params
 files

import tarfile
tar = tarfile.open("<model_name>.tar.gz", "w:gz")
for name in ["<model_name>-0000.params", "<model_name>-symbol.json"]:
```

```
    tar.add(name)
tar.close()
```

有关混合模型的更多信息，请参阅MXNet 杂交文档.

Gluon Model Zoo (GluonCV)

GluonCV 模型动物园模型是预先混合的。所以你可以直接导出它们。

```
import numpy as np
import mxnet as mx
import gluoncv as gcv
from gluoncv.utils import export_block
import tarfile

net = gcv.model_zoo.get_model('<model_name>', pretrained=True) # For example, choose
 <model_name> as resnet18_v1
export_block('<model_name>', net, preprocess=True, layout='HWC')

tar = tarfile.open("<model_name>.tar.gz", "w:gz")

for name in ["<model_name>-0000.params", "<model_name>-symbol.json"]:
    tar.add(name)
tar.close()
```

Non Gluon Models

保存到磁盘时所有非 Gluon 型号都使用`*-symbol`和`*.params`文件。因此，Neo 的格式已经是正确的。

```
# Pass the following 3 parameters: sym, args, aux
mx.model.save_checkpoint('<model_name>',0,sym,args,aux) # this will create
 <model_name>-symbol.json and <model_name>-0000.params files

import tarfile
tar = tarfile.open("<model_name>.tar.gz", "w:gz")

for name in ["<model_name>-0000.params", "<model_name>-symbol.json"]:
    tar.add(name)
tar.close()
```

PyTorch

PyTorch 模型必须保存为定义文件 (`.pt`要么`.pth`) 输入数据类型为`float32`.

要保存模型，请使用`torch.jit.trace`随后是`torch.save`. 这会将对象保存到磁盘文件中，默认情况下使用 python pickle (`pickle_module=pickle`) 来保存对象和一些元数据。接下来，将保存的模型转换为压缩的 tar 文件。

```
import torchvision
import torch

model = torchvision.models.resnet18(pretrained=True)
model.eval()
inp = torch.rand(1, 3, 224, 224)
model_trace = torch.jit.trace(model, inp)

# Save your model. The following code saves it with the .pth file extension
model_trace.save('model.pth')
```

```
# Save as a compressed tar file
import tarfile
with tarfile.open('model.tar.gz', 'w:gz') as f:
    f.add('model.pth')
f.close()
```

TensorFlow

TensorFlow 需要一个.pb或者一个.pbtxt文件和包含变量的变量目录。对于冷冻模型，只有一个.pb要么.pbtxt需要文件。

Pre-Trained Model

以下代码示例显示如何使用 tar Linux 命令来压缩模型。在终端或 Jupyter 笔记本中运行以下命令（如果您使用的是 Jupyter 笔记本，请插入!语句开头的魔法命令）：

```
# Download SSD_Mobilenet trained model
!wget http://download.tensorflow.org/models/object_detection/
ssd_mobilenet_v2_coco_2018_03_29.tar.gz

# unzip the compressed tar file
!tar xvf ssd_mobilenet_v2_coco_2018_03_29.tar.gz

# Compress the tar file and save it in a directory called 'model.tar.gz'
!tar czvf model.tar.gz ssd_mobilenet_v2_coco_2018_03_29/frozen_inference_graph.pb
```

本示例中使用的命令标志可完成以下操作：

- c：创建存档
- z: 用 gzip 压缩存档
- v：显示存档进度
- f：指定档案的文件名

内置估算器

内置估计器要么由特定于框架的容器或算法特定的容器进行。在使用内置的训练模型时，内置算法和框架特定估计器的 Esttimator 对象会以正确的格式保存模型.fit方法。

例如，您可以使用sagemaker.TensorFlow要定义 TensorFlow 估计器：

```
from sagemaker.tensorflow import TensorFlow

estimator = TensorFlow(entry_point='mnist.py',
                       role=role,  #param role can be arn of a sagemaker execution role
                       framework_version='1.15.3',
                       py_version='py3',
                       training_steps=1000,
                       evaluation_steps=100,
                       instance_count=2,
                       instance_type='ml.c4.xlarge')
```

然后使用训练模型.fit内置方法：

```
estimator.fit(inputs)
```

在最后使用内置来编译模型之前compile_model方法：

```
# Specify output path of the compiled model
output_path = '/'.join(estimator.output_path.split('/')[:-1])

# Compile model
optimized_estimator = estimator.compile_model(target_instance_family='ml_c5',
                            input_shape={'data':[1, 784]},  # Batch size 1, 3 channels,
 224x224 Images.
                            output_path=output_path,
                            framework='tensorflow', framework_version='1.15.3')
```

您也可以使用`sagemaker.estimator.Estimator`用于初始化一个估计器对象以进行训练和编译内置算法的类`compile_model`SageMaker Python 开发工具包中的方法：

```
import sagemaker
from sagemaker.image_uris import retrieve
sagemaker_session = sagemaker.Session()
aws_region = sagemaker_session.boto_region_name

# Specify built-in algorithm training image
training_image = retrieve(framework='image-classification',
                          region=aws_region, image_scope='training')

training_image = retrieve(framework='image-classification', region=aws_region,
 image_scope='training')

# Create estimator object for training
estimator = sagemaker.estimator.Estimator(image_uri=training_image,
                                          role=role,  #param role can be arn of a sagemaker
 execution role
                                          instance_count=1,
                                          instance_type='ml.p3.8xlarge',
                                          volume_size = 50,
                                          max_run = 360000,
                                          input_mode= 'File',
                                          output_path=s3_training_output_location,
                                          base_job_name='image-classification-training'
                                          )

# Setup the input data_channels to be used later for training.

train_data = sagemaker.inputs.TrainingInput(s3_training_data_location,
                                            content_type='application/x-recordio',
                                            s3_data_type='S3Prefix')
validation_data = sagemaker.inputs.TrainingInput(s3_validation_data_location,
                                                 content_type='application/x-recordio',
                                                 s3_data_type='S3Prefix')
data_channels = {'train': train_data, 'validation': validation_data}


# Train model
estimator.fit(inputs=data_channels, logs=True)

# Compile model with Neo

optimized_estimator = estimator.compile_model(target_instance_family='ml_c5',
                                          input_shape={'data':[1, 3, 224, 224],
 'softmax_label':[1]},
                                          output_path=s3_compilation_output_location,
                                          framework='mxnet',
                                          framework_version='1.7')
```

有关使用 SageMaker Python 开发工具包编译模型的更多信息，请参阅编译模型（Amazon SageMaker 开发工具包）(p. 2339).

编译模型 (Amazon Command Line Interface)

本部分介绍如何使用管理机器学习模型的 Amazon SageMaker Neo 编译作业。Amazon Command Line Interface(CLI)。您可以创建、描述、停止和列出编译作业。

1. 创建编译作业

   使用创建编译作业API 操作，您可以指定数据输入格式、存储模型的 S3 存储桶、写入编译后模型的 S3 存储桶以及目标硬件设备或平台。

   下表演示了如何配置：`CreateCompilationJob`API 基于您的目标是设备还是平台。

   Device Example

```
{
    "CompilationJobName": "neo-compilation-job-demo",
    "RoleArn": "arn:aws:iam::<your-account>:role/service-role/AmazonSageMaker-
ExecutionRole-yyyymmddThhmmss",
    "InputConfig": {
        "S3Uri": "s3://<your-bucket>/sagemaker/neo-compilation-job-demo-data/
train",
        "DataInputConfig":  "{'data': [1,3,1024,1024]}",
        "Framework": "MXNET"
    },
    "OutputConfig": {
        "S3OutputLocation": "s3://<your-bucket>/sagemaker/neo-compilation-job-demo-
data/compile",
        # A target device specification example for a ml_c5 instance family
        "TargetDevice": "ml_c5"
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 300
    }
}
```

   您可以选择指定与`FrameworkVersion`字段如果您使用 PyTorch 框架来训练模型并且目标设备是`ml_*` 目标。

```
{
    "CompilationJobName": "neo-compilation-job-demo",
    "RoleArn": "arn:aws:iam::<your-account>:role/service-role/AmazonSageMaker-
ExecutionRole-yyyymmddThhmmss",
    "InputConfig": {
        "S3Uri": "s3://<your-bucket>/sagemaker/neo-compilation-job-demo-data/
train",
        "DataInputConfig":  "{'data': [1,3,1024,1024]}",
        "Framework": "PYTORCH",
         # The FrameworkVersion field is only supported when compiling for PyTorch
 framework and ml_* targets,
         # excluding ml_inf. Supported values are 1.4 or 1.5 or 1.6 . Default is
 1.6
        "FrameworkVersion": "1.6"
    },
    "OutputConfig": {
        "S3OutputLocation": "s3://<your-bucket>/sagemaker/neo-compilation-job-demo-
data/compile",
        # A target device specification example for a ml_c5 instance family
        "TargetDevice": "ml_c5",
        # When compiling for ml_* instances using PyTorch framework, use the
 "CompilerOptions" field in
        # OutputConfig to provide the correct data type ("dtype") of the model's
 input. Default assumed is "float32"
```

```
        "CompilerOptions": "{'dtype': 'long'}"
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 300
    }
}
```

Note

此 API 字段仅适用于 PyTorch。

Platform Example

```
{
    "CompilationJobName": "neo-test-compilation-job",
    "RoleArn": "arn:aws:iam::<your-account>:role/service-role/AmazonSageMaker-
ExecutionRole-yyyymmddThhmmss",
    "InputConfig": {
        "S3Uri": "s3://<your-bucket>/sagemaker/neo-compilation-job-demo-data/
train",
        "DataInputConfig":  "{'data': [1,3,1024,1024]}",
        "Framework": "MXNET"
    },
    "OutputConfig": {
        "S3OutputLocation": "s3://<your-bucket>/sagemaker/neo-compilation-job-demo-
data/compile",
        # A target platform configuration example for a p3.2xlarge instance
        "TargetPlatform": {
            "Os": "LINUX",
            "Arch": "X86_64",
            "Accelerator": "NVIDIA"
        },
        "CompilerOptions": "{'cuda-ver': '10.0', 'trt-ver': '6.0.1', 'gpu-code':
 'sm_70'}"
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 300
    }
}
```

Note

对于OutputConfigAPI 操作，TargetDevice和TargetPlatformAPI 操作是互斥的。您必须选择两个选项中的一个。

要查找 JSON 字符串示例DataInputConfig根据框架，请参阅Neo 需要什么输入数据形状.

有关设置配置的更多信息，请参阅InputConfig、OutputConfig, 和目标平台SageMaker API 参考中的 API 操作。

2. 在配置 JSON 文件之后，可以运行以下命令来创建编译作业：

```
aws sagemaker create-compilation-job \
--cli-input-json file://job.json \
--region us-west-2

# You should get CompilationJobArn
```

3. 通过运行以下命令来描述编译作业：

```
aws sagemaker describe-compilation-job \
```

```
--compilation-job-name $JOB_NM \
--region us-west-2
```

4. 通过运行以下命令来停止编译作业：

```
aws sagemaker stop-compilation-job \
--compilation-job-name $JOB_NM \
--region us-west-2

# There is no output for compilation-job operation
```

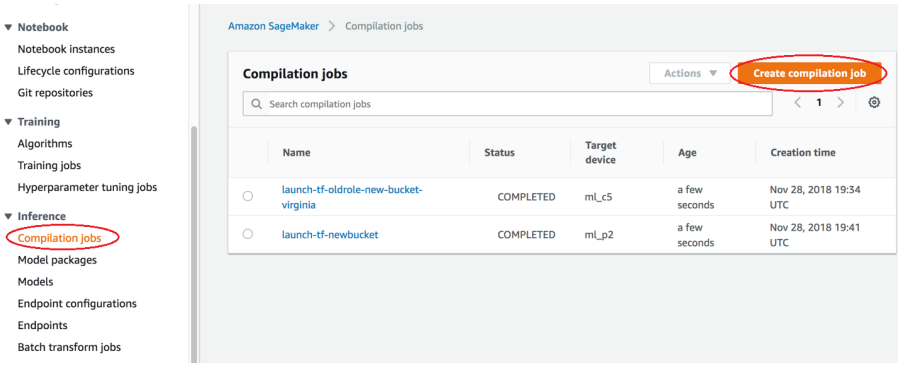
5. 通过运行以下命令列出编译作业：

```
aws sagemaker list-compilation-jobs \
--region us-west-2
```

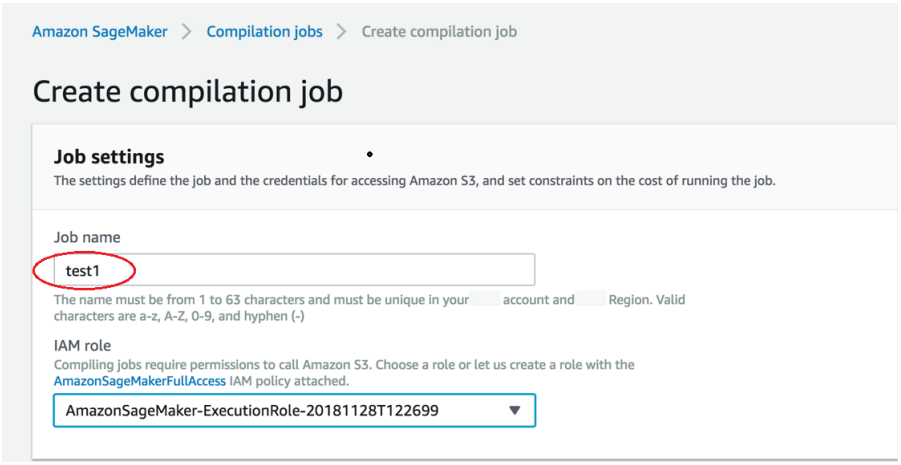
编译模型（Amazon SageMaker 控制台）

您可在 Amazon SageMaker 控制台中创建 Amazon SageMaker Neo 编译作业。

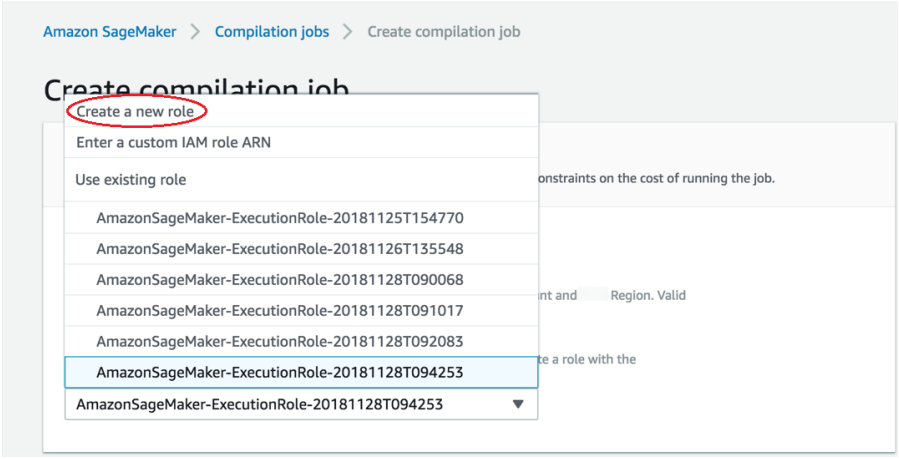
1. 在Amazon SageMaker控制台，选择编译作业，然后选择创建编译作业.



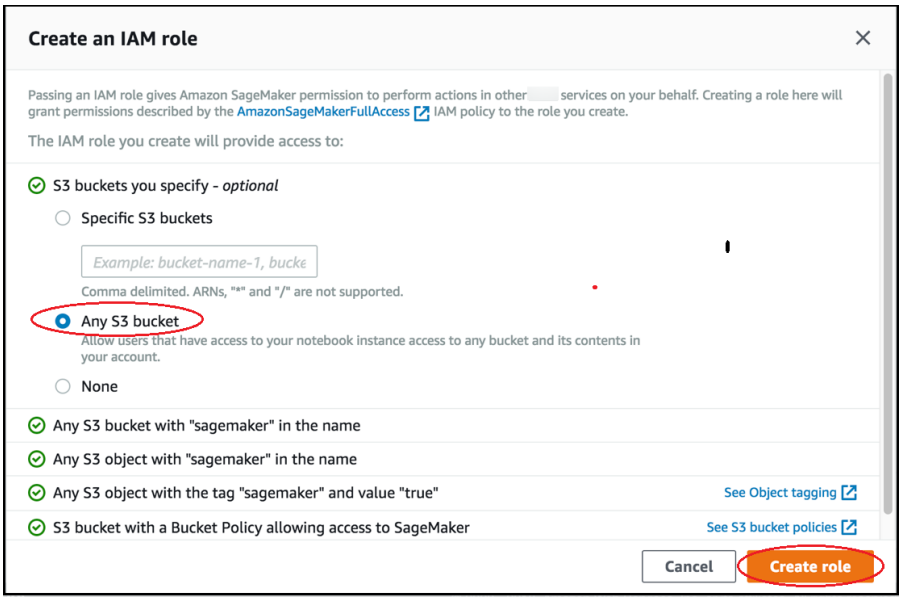
2. 在存储库的创建编译作业页面，下作业名称，输入一个名称。然后选择 IAM role (IAM 角色)。



3. 如果您没有 IAM 角色，请选择 Create a new role (创建新角色)。



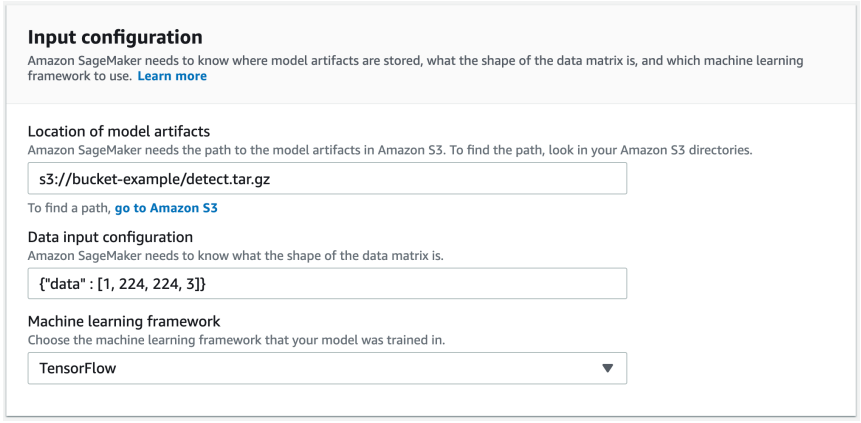
4. 在 Create an IAM role (创建 IAM 角色) 页面上，选择 Any S3 bucket (任意 S3 存储桶)，然后选择 Create role (创建角色)。



5. Non PyTorch Frameworks

   在输入配置部分中，输入包含模型项目的 Amazon S3 存储桶 URI 的完整路径模型工件的位置输入字段。模型工件必须采用压缩的 tarball 文件格式 (`.tar.gz`)。

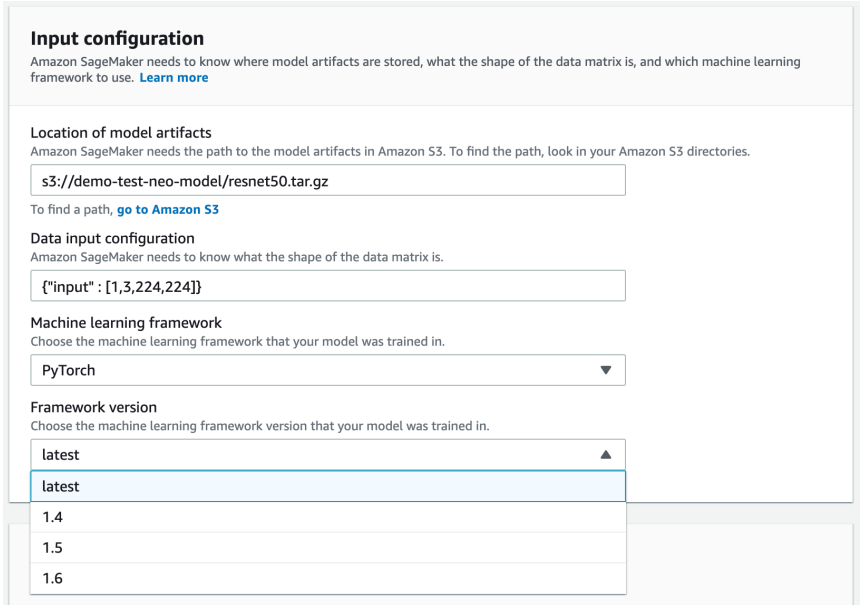
   对于数据输入配置字段中，输入用于指定输入数据形状的 JSON 字符串。适用于机器学习框架选择您选择的框架。



要根据框架查找输入数据形状的 JSON 字符串示例，请参阅Neo 需要什么输入数据形状.

PyTorch Framework

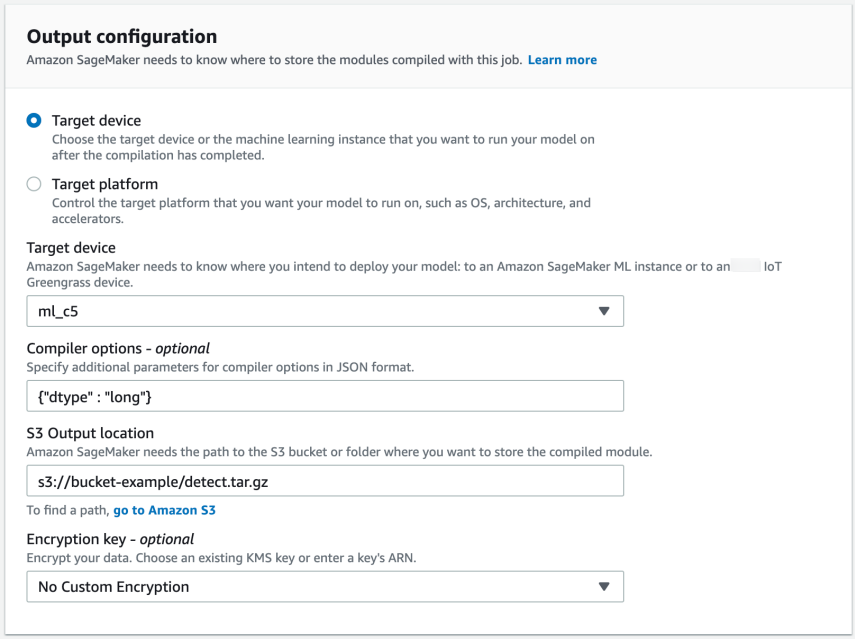
编译 PyTorch 模型也适用类似的说明。但是，如果你使用 PyTorch 训练并且正在尝试编译模型`ml_*`（除外）`ml_inf`) 目标，你可以选择指定你使用的 PyTorch 的版本。



要根据框架查找输入数据形状的 JSON 字符串示例，请参阅Neo 需要什么输入数据形状.

Note

在编译时`ml_*`使用 PyTorch 框架的实例，使用编译器选项字段中的输出配置提供正确的数据类型 (`dtype`) 模型的输入。默认值设置为`"float32"`.

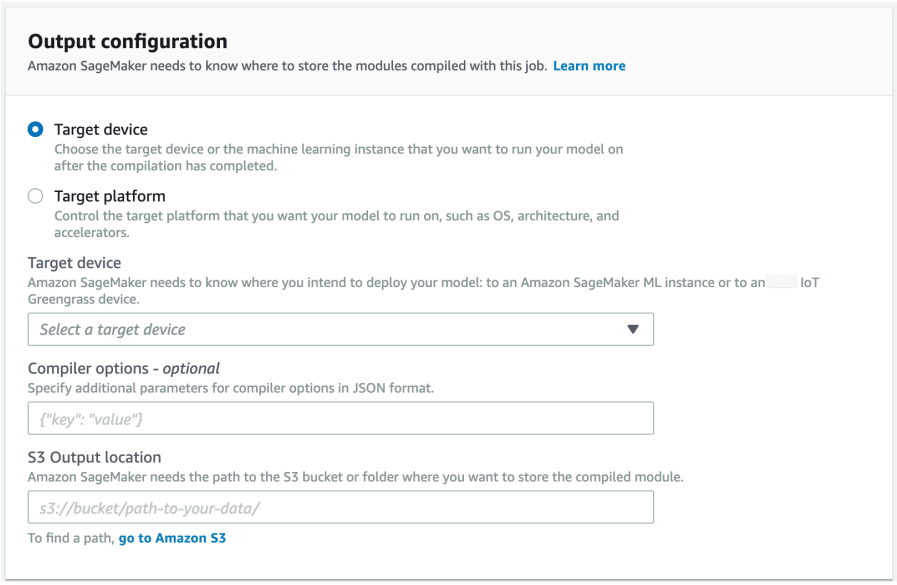


Warning

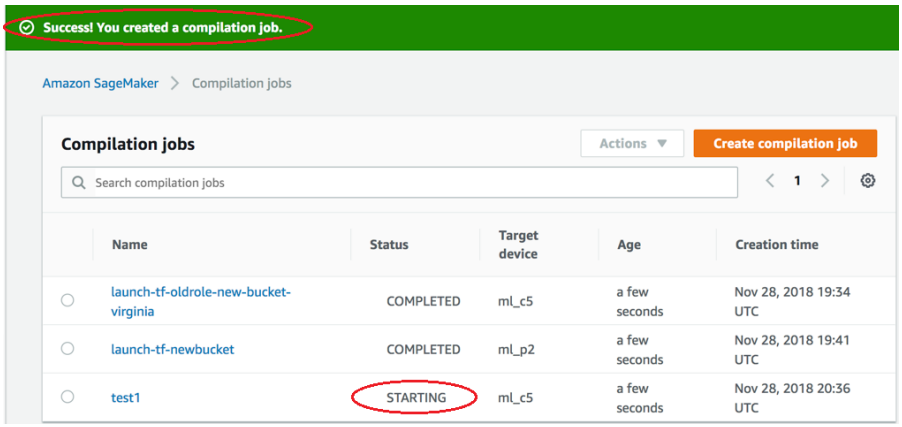
如果您指定的是导致的 Amazon S3 存储桶 URI 路径.pth文件中，您将在开始编译后收到以下错误：`ClientError: InputConfiguration: Unable to untar input model.Please confirm the model is a tar.gz file`

6. 转至输出配置部分。选择要在其中部署模型的位置。你可以将模型部署到目标设备或者目标平台. 目标设备包括云和边缘设备。目标平台是指您希望模型在其上运行的特定操作系统、架构和加速器。

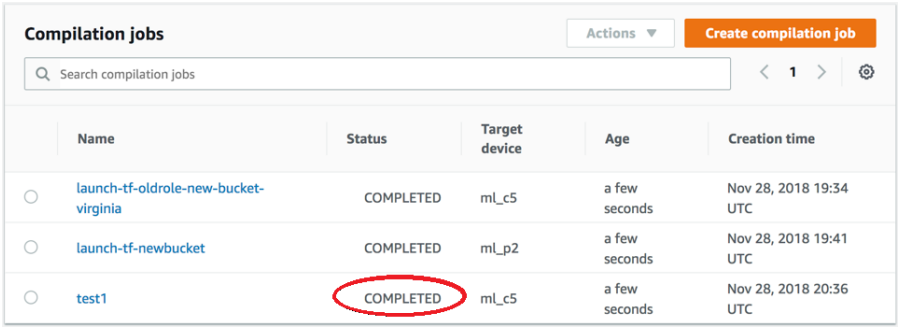
   适用于S3 输出位置，输入存储模型的 S3 存储桶的路径。您可以选择在下面添加 JSON 格式的编译器选项编译器选项部分。



7. 在启动时检查编译作业的状态。作业的此状态可在顶部找到。编译 Job页面，如以下屏幕截图所示。您也可以在状态column.



8. 在完成时检查编译作业的状态。您可以在状态列如以下屏幕截图所示。



编译模型（Amazon SageMaker 开发工具包）

您可以使用`compile_model`中的 API适用于 Python 的 Amazon SageMaker 开发工以编译训练有素的模型并针对特定目标硬件进行优化。应该在模型训练期间使用的估计器对象上调用 API。

> **Note**
>
> 你必须设置`MMS_DEFAULT_RESPONSE_TIMEOUT`环境变量`500`使用 MxNet 或 PyTorch 编译模型时。TensorFlow 不需要环境变量。

以下是说明如何使用编译模型的示例。`trained_model_estimator`对象：

```
# Replace the value of expected_trained_model_input below and
# specify the name & shape of the expected inputs for your trained model
# in json dictionary form
expected_trained_model_input = {'data':[1, 784]}

# Replace the example target_instance_family below to your preferred target_instance_family
compiled_model = trained_model_estimator.compile_model(target_instance_family='ml_c5',
        input_shape=expected_trained_model_input,
        output_path='insert s3 output path',
        env={'MMS_DEFAULT_RESPONSE_TIMEOUT': '500'})
```

代码会编译模型，将优化后模型保存在`output_path`，然后创建可部署到终端节点的 SageMaker 模型。有关使用适用于 Python 的开发工具包的示例笔记本，请参阅Neo 模型编译示例笔记本部分。

云实例

Amazon SageMaker Neo 为 TensorFlow、PyTorch、MXNet 等常用的机器学习框架提供编译支持。您可以将编译后的模型部署到云实例，AmazonInferentia 的实例。有关支持的框架和实例类型的完整列表，请参阅支持的实例类型和框架.

您可以通过以下三种方式之一编译模型：Amazon CLI、SageMaker 控制台或适用于 Python 的 SageMaker 开发工具包。请参阅，使用 Neo 编译模型有关. 编译后，您的模型工件将存储在编译作业期间指定的 Amazon S3 存储桶 URI 中。您可以将编译后的模型部署到云实例，Amazon使用适用于 Python 的 SageMaker SDK 的推断实例，Amazon SDK for Python (Boto3)、Amazon CLI，或者Amazon控制台。

如果你使用部署模型Amazon CLI、控制台或 Boto3，您必须为主容器选择 Docker 映像 Amazon ECR URI。请参阅Neo 推理容器映像获取亚马逊 ECR URI 的列表。

主题

- 支持的实例类型和框架构 (p. 2340)
- 部署模型 (p. 2342)
- 请求从已部署服务推理 (p. 2359)
- 推理容器映像 (p. 2362)

支持的实例类型和框架构

亚马逊SageMakerNeo 支持流行的深度学习框架进行编译和部署。您可以将模型部署到云实例，Amazon推理实例类型，或者 Amazon Elastic Inference 加速器。

下面介绍了框架SageMakerNeo 支持以及您可以编译和部署到的目标云实例。有关如何将编译后的模型部署到云或 Inferentia 实例的信息，请参阅使用云实例部署模型. 有关如何使用 Elastic Inference 加速器部署编译后的模型的信息，请参阅将 EI 在 Amazon SageMaker 托管终端节点上 (p. 2395).

云实例

SageMakerNeo 支持以下适用于 CPU 和 GPU 云实例的深度学习框架：

框架	框架版本	模型版本	模型	模型格式（包装在 *.tar.gz 中）	工具箱
MXNet	1.8.0	支持 1.8.0 或更低版本	图像分类、目标检测、语义分割、姿势估计、活动识别	一个符号文件 (.json) 和一个参数文件 (.params)	GluonCV v0.8.0
ONNX	1.7.0	支持 1.7.0 或更低版本	图像分类、SVM	一个模型文件 (.onnx)	
Keras	2.2.4	支持 2.2.4 或更低版本	图像分类	一个模型定义文件 (.h5)	
PyTorch	1.4、1.5、1.6、1.7 或 1.8	支持 1.4、1.5、1.6、1.7 和 1.8	图像分类	一个模型定义文件（.pt 或 .pth），输入 dtype 为 float32	
TensorFlow	1.15.3 或 2.4	支持 1.15.3 和 2.4	图像分类	对于保存的模型，需要一个 .pb 或一个 .pbtxt 文件，	

框架	框架版本	模型版本	模型	模型格式（包装在 *.tar.gz 中）	工具箱
				以及包含变量的变量目录。 对于冻结模型，只需要一个 .pb 或一个 .pbtxt 文件	
XGBoost	1.3.3	支持 1.3.3 或更低版本	决策树	其中树的节点数小于 2^31 个 XGBoost 模型文件 (.model)	

Note

“模型版本” 是用于训练和导出模型的框架的版本。

实例类型

你可以部署SageMaker已编译模型到下面列出的云实例之一：

实例	计算类型				
`ml_c4`	标准				
`ml_c5`	标准				
`ml_m4`	标准				
`ml_m5`	标准				
`ml_p2`	加速计算				
`ml_p3`	加速计算				
`ml_g4dn`	加速计算				

有关每种实例类型的可用 vCPU、内存和每小时价格的信息，请参阅亚马逊SageMaker定价.

Note

在编译时`ml_*`使用实例PyTorch框架，使用编译器选项字段输出配置提供正确的数据类型 (`dtype`) 模型的输入。
默认值设置为`"float32"`.

Amazon推理

SageMakerInferentia 支持以下深度学习框架：

框架	框架版本	模型版本	模型	模型格式（包装在 *.tar.gz 中）	工具箱
MXNet	1.5.1	支持 1.5.1 或更低版本	图像分类、目标检测、语义分	一个符号文件 (.json) 和	GluonCV v0.8.0

框架	框架版本	模型版本	模型	模型格式（包装在 *.tar.gz 中）	工具箱
			割、姿势估计、活动识别	一个参数文件 (.params)	
PyTorch	1.7.1	支持 1.7.1 或更低版本	图像分类	一个模型定义文件（.pt 或 .pth），输入 dtype 为 float32	
TensorFlow	1.15.0	支持 1.15.0 或更低版本	图像分类	对于保存的模型，需要一个 .pb 或一个 .pbtxt 文件，以及包含变量的变量目录。 对于冻结模型，只需要一个 .pb 或一个 .pbtxt 文件	

Note

“模型版本” 是用于训练和导出模型的框架的版本。

你可以部署SageMakerNEO 编译模型Amazon基于推断的 Amazon EC2 Inf1 实例。AmazonInferentia 是亚马逊首款旨在加速深度学习的定制硅芯片。目前，您可以使用`ml_inf1`实例来部署编译后的模型。

Amazon Elastic Inference

SageMakerElastic Inference 支持以下深度学习框架：

框架	框架版本	模型版本	模型	模型格式（包装在 *.tar.gz 中）
TensorFlow	2.3.2	支持 2.3	图像分类、目标检测、语义分割、姿势估计、活动识别	对于保存的模型，需要一个 .pb 或一个 .pbtxt 文件，以及包含变量的变量目录。 对于冻结模型，只需要一个 .pb 或一个 .pbtxt 文件。

你可以部署SageMaker新编译模型转换为 Elastic Inference 加速器。有关更多信息，请参阅将 EI 在 Amazon SageMaker 托管终端节点上 (p. 2395)。

部署模型

部署亚马逊 SageMaker Neo 编译的模型转换为 HTTPS 终端节点，您必须使用 Amazon 为模型配置和创建终端节点。 SageMaker 托管服务。目前，开发者可以使用亚马逊 SageMaker 将模块部署到 ml.c5、ml.c4、ml.c4、ml.m5、ml.c4、ml.p3、ml.p3、ml.p3、ml.p2 和 ml.inf1 实例上。

对于 Inf1 实例，需要专门为 ml.inf1 实例编译模型。无法保证为其他实例类型编译的模型能够与 ml.inf1 实例配合使用。

适用于Elastic Inference 加速器，需要专门为 ml_eia2 设备编译模型。有关如何将编译后的模型部署到 Elastic Inference 加速器的信息，请参阅将 EI 在 Amazon SageMaker 托管终端节点上 (p. 2395).

部署编译的模型时，您需要为用于编译的目标使用相同的实例。这会创建一个 SageMaker 您可以用来执行推理的终端节点。您可以使用以下任一方法部署 NEO 编译模型：亚马逊 SageMaker 适用于 Python 的开发工具包、适用于 Python (Boto3) 的 SDK、Amazon Command Line Interface，以及 SageMaker 控制台https://console.aws.amazon.com/ SageMaker.

> **Note**
>
> 用于使用部署模型Amazon CLI、控制台或 Boto3，请参阅Neo 推理容器映像为主容器选择推理图像 URI。

主题

- 先决条件 (p. 2343)
- 使用部署编译模型 SageMaker 开发工具包 (p. 2349)
- 使用 Boto3 部署编译模型 (p. 2351)
- 使用Amazon CLI (p. 2353)
- 使用控制台部署已编译的模型 (p. 2355)

先决条件

> **Note**
>
> 如果您使用编译模型，请按照本节中的说明操作Amazon SDK for Python (Boto3)、Amazon CLI，或者 SageMaker 控制台。

创建 SageMaker 新编译模型，你需要以下内容：

1. Docker 镜像亚马逊 ECR URI。您可以从中选择满足您的需求的选择此列表.
2. 入口点脚本文件：

   a. 适用于 PyTorch 和 MXNet 模型：

      如果你使用 SageMaker 训练模型，训练脚本必须实现下面描述的功能。在推理期间，训练脚本作为入口点脚本。在中详细介绍的示例中MNIST 使用 MXNet 模块进行 MNIST 训练、编译和部署 SageMaker 新，训练脚本（`mnist.py`) 实现所需的功能。

      如果你没有使用 SageMaker 训练模型，你需要提供入口点脚本 (`inference.py`) 推理时可以使用的文件。基于框架（MXNet 或 PyTORCHH）推理脚本位置必须符合 SageMaker Python 开发工具包MxNet 的模型目录结构要么PyTorch 的模型目录结构.

      当使用 Neo 推理优化容器镜像时PyTorch和MXNet在 CPU 和 GPU 实例类型上，推理脚本必须实现以下功能：

      - `model_fn`：加载模型。(可选)
      - `input_fn`：将传入的请求有效负载转换为 numpy 数组。
      - `predict_fn`：执行预测。
      - `output_fn`：将预测输出转换为响应有效负载。
      - 或者，您可以定义`transform_fn`组合`input_fn`、`predict_fn`, 和`output_fn`.

      示例如下：`inference.py`名为的目录中的脚本`code`(`code/inference.py`) 对于PyTorch 和 MxNet（Gluon 和模块）。这些示例首先加载模型，然后在 GPU 上的图像数据上提供模型：

MXNet Module

```
import numpy as np
import json
import mxnet as mx
import neomx  # noqa: F401
from collections import namedtuple

Batch = namedtuple('Batch', ['data'])

# Change the context to mx.cpu() if deploying to a CPU endpoint
ctx = mx.gpu()

def model_fn(model_dir):
    # The compiled model artifacts are saved with the prefix 'compiled'
    sym, arg_params, aux_params = mx.model.load_checkpoint('compiled', 0)
    mod = mx.mod.Module(symbol=sym, context=ctx, label_names=None)
    exe = mod.bind(for_training=False,
                   data_shapes=[('data', (1,3,224,224))],
                   label_shapes=mod._label_shapes)
    mod.set_params(arg_params, aux_params, allow_missing=True)

    # Run warm-up inference on empty data during model load (required for GPU)
    data = mx.nd.empty((1,3,224,224), ctx=ctx)
    mod.forward(Batch([data]))
    return mod


def transform_fn(mod, image, input_content_type, output_content_type):
    # pre-processing
    decoded = mx.image.imdecode(image)
    resized = mx.image.resize_short(decoded, 224)
    cropped, crop_info = mx.image.center_crop(resized, (224, 224))
    normalized = mx.image.color_normalize(cropped.astype(np.float32) / 255,
                                  mean=mx.nd.array([0.485, 0.456, 0.406]),
                                  std=mx.nd.array([0.229, 0.224, 0.225]))
    transposed = normalized.transpose((2, 0, 1))
    batchified = transposed.expand_dims(axis=0)
    casted = batchified.astype(dtype='float32')
    processed_input = casted.as_in_context(ctx)

    # prediction/inference
    mod.forward(Batch([processed_input]))

    # post-processing
    prob = mod.get_outputs()[0].asnumpy().tolist()
    prob_json = json.dumps(prob)
    return prob_json, output_content_type
```

MXNet Gluon

```
import numpy as np
import json
import mxnet as mx
import neomx  # noqa: F401

# Change the context to mx.cpu() if deploying to a CPU endpoint
ctx = mx.gpu()

def model_fn(model_dir):
    # The compiled model artifacts are saved with the prefix 'compiled'
    block = mx.gluon.nn.SymbolBlock.imports('compiled-symbol.json',
['data'],'compiled-0000.params', ctx=ctx)
```

```
    # Hybridize the model & pass required options for Neo: static_alloc=True &
 static_shape=True
    block.hybridize(static_alloc=True, static_shape=True)

    # Run warm-up inference on empty data during model load (required for GPU)
    data = mx.nd.empty((1,3,224,224), ctx=ctx)
    warm_up = block(data)
    return block


def input_fn(image, input_content_type):
    # pre-processing
    decoded = mx.image.imdecode(image)
    resized = mx.image.resize_short(decoded, 224)
    cropped, crop_info = mx.image.center_crop(resized, (224, 224))
    normalized = mx.image.color_normalize(cropped.astype(np.float32) / 255,
                                  mean=mx.nd.array([0.485, 0.456, 0.406]),
                                  std=mx.nd.array([0.229, 0.224, 0.225]))
    transposed = normalized.transpose((2, 0, 1))
    batchified = transposed.expand_dims(axis=0)
    casted = batchified.astype(dtype='float32')
    processed_input = casted.as_in_context(ctx)
    return processed_input


def predict_fn(processed_input_data, block):
    # prediction/inference
    prediction = block(processed_input_data)
    return prediction

def output_fn(prediction, output_content_type):
    # post-processing
    prob = prediction.asnumpy().tolist()
    prob_json = json.dumps(prob)
    return prob_json, output_content_type
```

PyTorch 1.4 and Older

```
import os
import torch
import torch.nn.parallel
import torch.optim
import torch.utils.data
import torch.utils.data.distributed
import torchvision.transforms as transforms
from PIL import Image
import io
import json
import pickle


def model_fn(model_dir):
    """Load the model and return it.
    Providing this function is optional.
    There is a default model_fn available which will load the model
    compiled using SageMaker Neo. You can override it here.

    Keyword arguments:
    model_dir -- the directory path where the model artifacts are present
    """

    # The compiled model is saved as "compiled.pt"
    model_path = os.path.join(model_dir, 'compiled.pt')
```

```
    with torch.neo.config(model_dir=model_dir, neo_runtime=True):
        model = torch.jit.load(model_path)
        device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        model = model.to(device)

    # We recommend that you run warm-up inference during model load
    sample_input_path = os.path.join(model_dir, 'sample_input.pkl')
    with open(sample_input_path, 'rb') as input_file:
        model_input = pickle.load(input_file)
    if torch.is_tensor(model_input):
        model_input = model_input.to(device)
        model(model_input)
    elif isinstance(model_input, tuple):
        model_input = (inp.to(device) for inp in model_input if
 torch.is_tensor(inp))
        model(*model_input)
    else:
        print("Only supports a torch tensor or a tuple of torch tensors")
        return model


def transform_fn(model, request_body, request_content_type,
                 response_content_type):
    """Run prediction and return the output.
    The function
    1. Pre-processes the input request
    2. Runs prediction
    3. Post-processes the prediction output.
    """
    # preprocess
    decoded = Image.open(io.BytesIO(request_body))
    preprocess = transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(
            mean=[
                0.485, 0.456, 0.406], std=[
                0.229, 0.224, 0.225]),
    ])
    normalized = preprocess(decoded)
    batchified = normalized.unsqueeze(0)
    # predict
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    batchified = batchified.to(device)
    output = model.forward(batchified)

    return json.dumps(output.cpu().numpy().tolist()), response_content_type
```

PyTorch 1.5 and Newer

使用neopytorch以指定模型的文件路径（如果你训练过） PyTorch 版本 1.5 及更新。

```
import os
import torch
import neopytorch
import torch.nn.parallel
import torch.optim
import torch.utils.data
import torch.utils.data.distributed
import torchvision.transforms as transforms
from PIL import Image
import io
import json
```

```
import pickle


def model_fn(model_dir):
    """Load the model and return it.
    Providing this function is optional.
    There is a default_model_fn available, which will load the model
    compiled using SageMaker Neo. You can override the default here.
    The model_fn only needs to be defined if your model needs extra
    steps to load, and can otherwise be left undefined.

    Keyword arguments:
    model_dir -- the directory path where the model artifacts are present
    """

    # The compiled model is saved as "compiled.pt"
    model_path = os.path.join(model_dir, 'compiled.pt')
    neopytorch.config(model_dir=model_dir,neo_runtime=True)
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model = torch.jit.load(model_path, map_location=device)
    model = model.to(device)

    # We recommend that you run warm-up inference during model load
    sample_input_path = os.path.join(model_dir, 'sample_input.pkl')
    with open(sample_input_path, 'rb') as input_file:
        model_input = pickle.load(input_file)
    if torch.is_tensor(model_input):
        model_input = model_input.to(device)
        model(model_input)
    elif isinstance(model_input, tuple):
        model_input = (inp.to(device)
                       for inp in model_input if torch.is_tensor(inp))
        model(*model_input)
    else:
        print("Only supports a torch tensor or a tuple of torch tensors")
    return model


def transform_fn(model, request_body, request_content_type,
                     response_content_type):
    """Run prediction and return the output.
    The function
    1. Pre-processes the input request
    2. Runs prediction
    3. Post-processes the prediction output.
    """
    # preprocess
    decoded = Image.open(io.BytesIO(request_body))
    preprocess = transforms.Compose([
                                transforms.Resize(256),
                                transforms.CenterCrop(224),
                                transforms.ToTensor(),
                                transforms.Normalize(
                                    mean=[
                                        0.485, 0.456, 0.406], std=[
                                        0.229, 0.224, 0.225]),
                                    ])
    normalized = preprocess(decoded)
    batchified = normalized.unsqueeze(0)

    # predict
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    batchified = batchified.to(device)
    output = model.forward(batchified)
    return json.dumps(output.cpu().numpy().tolist()), response_content_type
```

b. 对于 INF1 实例或 onnx、xgboost、keras 容器镜像

对于所有其他 Neo Inference 优化的容器映像或推断实例类型，入口点脚本必须为 Neo 深度学习运行时实现以下功能：

- `neo_preprocess`：将传入的请求有效负载转换为 numpy 数组。
- `neo_postprocess`：将 Neo 深度学习运行时的预测输出转换为响应正文。

Note

前面的两个函数不使用 MXNet、PyTorch 或 TensorFlow 的任何功能。

有关如何使用这些函数的示例，请参阅Neo 模型编译示例笔记本.

c. 适用于 TensorFlow 模型

如果模型在将数据发送到模型之前需要自定义的预处理和后处理逻辑，则必须指定入口点脚本`inference.py`推理时可以使用的文件。脚本应该实现任一对`input_handler`和`output_handler`函数或单个处理函数。

Note

请注意，如果实现了处理函数，`input_handler`和`output_handler`将忽略。

以下是代码示例`inference.py`脚本可以与编译模型一起组合在一起，以便对图像分类模型执行自定义的预处理和后处理。这些区域有： SageMaker 客户端将图像文件作为`application/x-image`将内容类型转换为`input_handler`函数，其中它被转换为 JSON。然后将转换后的图像文件发送到Tensorflow 模型服务器 (TFX)使用 REST API。

```
import json
import numpy as np
import json
import io
from PIL import Image

def input_handler(data, context):
    """ Pre-process request input before it is sent to TensorFlow Serving REST API

    Args:
    data (obj): the request data, in format of dict or string
    context (Context): an object containing request and configuration details

    Returns:
    (dict): a JSON-serializable dict that contains request body and headers
    """
    f = data.read()
    f = io.BytesIO(f)
    image = Image.open(f).convert('RGB')
    batch_size = 1
    image = np.asarray(image.resize((512, 512)))
    image = np.concatenate([image[np.newaxis, :, :]] * batch_size)
    body = json.dumps({"signature_name": "serving_default", "instances":
 image.tolist()})
    return body

def output_handler(data, context):
    """Post-process TensorFlow Serving output before it is returned to the client.

    Args:
    data (obj): the TensorFlow serving response
    context (Context): an object containing request and configuration details
```

```
    Returns:
    (bytes, string): data to return to client, response content type
    """
    if data.status_code != 200:
        raise ValueError(data.content.decode('utf-8'))

    response_content_type = context.accept_header
    prediction = data.content
    return prediction, response_content_type
```

如果没有自定义的预处理或后处理，则 SageMaker 客户端先以类似的方式将文件图像转换为 JSON，然后再将其发送到 SageMaker 终端节点。

有关更多信息，请参阅 。部署到 TensorFlow 在中提供终端节点 SageMaker Python 开发工具包.

3. 包含已编译模型构件的 Amazon S3 存储桶 URI。

使用部署编译模型 SageMaker 开发工具包

你必须满足先决条件部分是否使用编译模型Amazon SDK for Python (Boto3)、Amazon CLI，或者亚马逊 SageMaker 控制台。按照以下使用案例之一部署编译的模型 SageMaker Neo 取决于你编译模型的方式。

主题

- 如果你使用 SageMaker 开发工具包 (p. 2349)
- 如果你使用 MxNet 或 PyTorch 编译模型 (p. 2349)
- 如果你使用 Boto3 编译模型， SageMaker 控制台或 TensorFlow 的 CLI (p. 2351)

如果你使用 SageMaker 开发工具包

这些区域有：SageMaker.模型已编译模型的对象句柄提供部署 ()函数，这允许您创建终端节点来服务推理请求。使用该函数，您可以设置用于终端节点的实例的数量和类型。您必须选择已经为其编译了模型的实例。例如，在中的编译的作业编译模型（亚马逊） SageMaker SDK)部分，这是`ml_c5`.

```
predictor = compiled_model.deploy(initial_instance_count = 1, instance_type =
 'ml.c5.4xlarge')

# Print the name of newly created endpoint
print(predictor.endpoint_name)
```

如果你使用 MxNet 或 PyTorch 编译模型

创建 SageMaker 建模并使用框架特定模型 API 下的 deploy () API 进行部署。对于 MxNet 来说，它是MxNet 模型而对于 PyTorch 来说，这是Pytorch 模型. 当你创建和部署 SageMaker 模型，你必须设置`MMS_DEFAULT_RESPONSE_TIMEOUT`环境变量`500`然后指定`entry_point`参数作为推理脚本 (`inference.py`) 和`source_dir`参数作为目录位置 (`code`) 推理脚本。准备推理脚本 (`inference.py`) 按照先决条件步骤操作。

以下示例演示如何使用 SageMaker 适用于 Python 的 SDK：

MXNet

```
from sagemaker.mxnet import MXNetModel

# Create SageMaker model and deploy an endpoint
sm_mxnet_compiled_model = MXNetModel(
```

```
    model_data='insert S3 path of compiled MXNet model archive',
    role='AmazonSageMaker-ExecutionRole',
    entry_point='inference.py',
    source_dir='code',
    framework_version='1.8.0',
    py_version='py3',
    image_uri='insert appropriate ECR Image URI for MXNet',
    env={'MMS_DEFAULT_RESPONSE_TIMEOUT': '500'},
)

# Replace the example instance_type below to your preferred instance_type
predictor = sm_mxnet_compiled_model.deploy(initial_instance_count = 1, instance_type =
 'ml.p3.2xlarge')

# Print the name of newly created endpoint
print(predictor.endpoint_name)
```

PyTorch 1.4 and Older

```
from sagemaker.pytorch import PyTorchModel

# Create SageMaker model and deploy an endpoint
sm_pytorch_compiled_model = PyTorchModel(
    model_data='insert S3 path of compiled PyTorch model archive',
    role='AmazonSageMaker-ExecutionRole',
    entry_point='inference.py',
    source_dir='code',
    framework_version='1.4.0',
    py_version='py3',
    image_uri='insert appropriate ECR Image URI for PyTorch',
    env={'MMS_DEFAULT_RESPONSE_TIMEOUT': '500'},
)

# Replace the example instance_type below to your preferred instance_type
predictor = sm_pytorch_compiled_model.deploy(initial_instance_count = 1, instance_type
 = 'ml.p3.2xlarge')

# Print the name of newly created endpoint
print(predictor.endpoint_name)
```

PyTorch 1.5 and Newer

```
from sagemaker.pytorch import PyTorchModel

# Create SageMaker model and deploy an endpoint
sm_pytorch_compiled_model = PyTorchModel(
    model_data='insert S3 path of compiled PyTorch model archive',
    role='AmazonSageMaker-ExecutionRole',
    entry_point='inference.py',
    source_dir='code',
    framework_version='1.5',
    py_version='py3',
    image_uri='insert appropriate ECR Image URI for PyTorch',
)

# Replace the example instance_type below to your preferred instance_type
predictor = sm_pytorch_compiled_model.deploy(initial_instance_count = 1, instance_type
 = 'ml.p3.2xlarge')

# Print the name of newly created endpoint
print(predictor.endpoint_name)
```

Note

这些区域有：AmazonSageMakerFullAccess和AmazonS3ReadOnlyAccess策略必须附加到AmazonSageMaker-ExecutionRoleIAM 角色。

如果你使用 Boto3 编译模型， SageMaker 控制台或 TensorFlow 的 CLI

构造TensorFlowModel对象，然后调用 deploy:

```
role='AmazonSageMaker-ExecutionRole'
model_path='S3 path for model file'
framework_image='inference container arn'
tf_model = TensorFlowModel(model_data=model_path,
                framework_version='1.15.3',
                role=role,
                image_uri=framework_image)
instance_type='ml.c5.xlarge'
predictor = tf_model.deploy(instance_type=instance_type,
                    initial_instance_count=1)
```

请参阅直接从模型项目部署了解更多信息。

您可以从以下位置选择满足您需求的 Docker 映像 Amazon ECR URI此列表.

有关如何构建TensorFlowModel对象，请参阅SageMaker SDK.

Note

如果在 GPU 上部署模型，您的第一个推理请求可能会有很高的延迟。这是因为在第一个推理请求中创建了一个优化的计算内核。我们建议您在将推理请求的预热文件发送到 TFX 之前将其与模型文件一起存储。这被称为 “热身” 模型。

下面的代码片段演示了如何为图像分类示例生成预热文件先决条件部分:

```
import tensorflow as tf
from tensorflow_serving.apis import classification_pb2
from tensorflow_serving.apis import inference_pb2
from tensorflow_serving.apis import model_pb2
from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_log_pb2
from tensorflow_serving.apis import regression_pb2
import numpy as np

with tf.python_io.TFRecordWriter("tf_serving_warmup_requests") as writer:
    img = np.random.uniform(0, 1, size=[224, 224, 3]).astype(np.float32)
    img = np.expand_dims(img, axis=0)
    test_data = np.repeat(img, 1, axis=0)
    request = predict_pb2.PredictRequest()
    request.model_spec.name = 'compiled_models'
    request.model_spec.signature_name = 'serving_default'
    request.inputs['Placeholder:0'].CopyFrom(tf.compat.v1.make_tensor_proto(test_data,
 shape=test_data.shape, dtype=tf.float32))
    log = prediction_log_pb2.PredictionLog(
    predict_log=prediction_log_pb2.PredictLog(request=request))
    writer.write(log.SerializeToString())
```

有关如何 “预热” 模型的更多信息，请参阅TensorFlow TFX 页面.

使用 Boto3 部署编译模型

你必须满足先决条件部分是否使用编译模型Amazon SDK for Python (Boto3)、Amazon CLI，或者亚马逊 SageMaker 控制台。按照以下步骤创建和部署 SageMaker 使用新编译模型适用于 Python 的 Amazon Web Services 开发工具包 (Boto3).

主题

- 部署模型 (p. 2352)

部署模型

在你满意之后先决条件，使用create_model、create_enpoint_config, 和create_endpointAPI。

以下示例演示如何使用这些 API 部署使用 Neo 编译的模型：

```
import boto3
client = boto3.client('sagemaker')

# create sagemaker model
create_model_api_response = client.create_model(
                                    ModelName='my-sagemaker-model',
                                    PrimaryContainer={
                                        'Image': <insert the ECR Image URI>,
                                        'ModelDataUrl': 's3://path/to/model/artifact/
model.tar.gz',
                                        'Environment': {}
                                    },
                                    ExecutionRoleArn='ARN for AmazonSageMaker-
ExecutionRole'
                            )

print ("create_model API response", create_model_api_response)

# create sagemaker endpoint config
create_endpoint_config_api_response = client.create_endpoint_config(
                                            EndpointConfigName='sagemaker-neomxnet-
endpoint-configuration',
                                            ProductionVariants=[
                                                {
                                                    'VariantName': <provide your variant
 name>,
                                                    'ModelName': 'my-sagemaker-model',
                                                    'InitialInstanceCount': 1,
                                                    'InstanceType': <provide your instance
 type here>
                                                },
                                            ]
                                        )

print ("create_endpoint_config API response", create_endpoint_config_api_response)

# create sagemaker endpoint
create_endpoint_api_response = client.create_endpoint(
                                    EndpointName='provide your endpoint name',
                                    EndpointConfigName=<insert your endpoint config name>,
                                )

print ("create_endpoint API response", create_endpoint_api_response)
```

> **Note**
>
> 这些区域有：AmazonSageMakerFullAccess和AmazonS3ReadOnlyAccess策略必须附加到AmazonSageMaker-ExecutionRoleIAM 角色。

对于的完整语法create_model、create_endpoint_config, 和create_endpointAPI，请参阅create_model、create_endpoint_config, 和create_endpoint，分别。

如果您没有使用 SageMaker 训练模型，请指定以下环境变量：

MXNet and PyTorch

```
"Environment": {
    "SAGEMAKER_PROGRAM": "inference.py",
    "SAGEMAKER_SUBMIT_DIRECTORY": "/opt/ml/model/code",
    "SAGEMAKER_CONTAINER_LOG_LEVEL": "20",
    "SAGEMAKER_REGION": "insert your region",
    "MMS_DEFAULT_RESPONSE_TIMEOUT": "500"
}
```

TensorFlow

```
"Environment": {
    "SAGEMAKER_PROGRAM": "inference.py",
    "SAGEMAKER_SUBMIT_DIRECTORY": "/opt/ml/model/code",
    "SAGEMAKER_CONTAINER_LOG_LEVEL": "20",
    "SAGEMAKER_REGION": "insert your region"
}
```

如果您使用 SageMaker 训练模型，请指定环境变量`SAGEMAKER_SUBMIT_DIRECTORY`作为包含训练脚本的完整 Amazon S3 存储桶 URI。

使用Amazon CLI

你必须满足先决条件部分是否使用编译模型Amazon SDK for Python (Boto3)、Amazon CLI，或者亚马逊 SageMaker 控制台。按照以下步骤创建和部署 SageMaker 使用新编译模型Amazon CLI.

主题

- 部署模型 (p. 2353)

部署模型

在你满意之后先决条件，使用`create-model`、`create-enpoint-config`, 和`create-endpoint` Amazon CLI命令。以下步骤说明了如何使用这些命令部署使用 Neo 编译的模型：

创建模型

从Neo 推理容器映像，选择推理图像 URI 然后使用`create-model`用于创建 SageMaker 模型。您可以通过两步执行此操作：

1. 创建 `create_model.json` 文件。在文件中，指定模型的名称、图像 URI、`model.tar.gz`Amazon S3 存储桶中的文件和 SageMaker 执行角色：

```
{
    "ModelName": "insert model name",
    "PrimaryContainer": {
        "Image": "insert the ECR Image URI",
        "ModelDataUrl": "insert S3 archive URL",
        "Environment": {"See details below"}
    },
    "ExecutionRoleArn": "ARN for AmazonSageMaker-ExecutionRole"
}
```

如果您使用 SageMaker 训练了模型，请指定以下环境变量：

```
"Environment": {
```

```
    "SAGEMAKER_SUBMIT_DIRECTORY" : "[Full S3 path for *.tar.gz file containing the
 training script]"
}
```

如果您没有使用 SageMaker 训练模型，请指定以下环境变量：

MXNet and PyTorch

```
"Environment": {
    "SAGEMAKER_PROGRAM": "inference.py",
    "SAGEMAKER_SUBMIT_DIRECTORY": "/opt/ml/model/code",
    "SAGEMAKER_CONTAINER_LOG_LEVEL": "20",
    "SAGEMAKER_REGION": "insert your region",
    "MMS_DEFAULT_RESPONSE_TIMEOUT": "500"
}
```

TensorFlow

```
"Environment": {
    "SAGEMAKER_PROGRAM": "inference.py",
    "SAGEMAKER_SUBMIT_DIRECTORY": "/opt/ml/model/code",
    "SAGEMAKER_CONTAINER_LOG_LEVEL": "20",
    "SAGEMAKER_REGION": "insert your region"
}
```

Note

这些区域有：AmazonSageMakerFullAccess和AmazonS3ReadOnlyAccess策略必须附加到AmazonSageMaker-ExecutionRoleIAM 角色。

2. 运行以下 命令：

```
aws sagemaker create-model --cli-input-json file://create_model.json
```

有关 create-model API 的完整语法，请参阅 create-model。

创建终端节点配置

创建之后 SageMaker 模型中，使用create-endpoint-configAPI。为此，请使用终端节点配置规范创建一个 JSON 文件。例如，您可以使用以下代码模板并将其另存为create_config.json：

```
{
    "EndpointConfigName": "<provide your endpoint config name>",
    "ProductionVariants": [
        {
            "VariantName": "<provide your variant name>",
            "ModelName": "my-sagemaker-model",
            "InitialInstanceCount": 1,
            "InstanceType": "<provide your instance type here>",
            "InitialVariantWeight": 1.0
        }
    ]
}
```

现在运行以下步骤Amazon CLI命令来创建终端节点配置：

```
aws sagemaker create-endpoint-config --cli-input-json file://create_config.json
```

有关 create-endpoint-config API 的完整语法，请参阅 create-endpoint-config。

创建终端节点

创建终端节点配置后，请使用`create-endpoint`API：

```
aws sagemaker create-endpoint --endpoint-name '<provide your endpoint name>' --endpoint-
config-name '<insert your endpoint config name>'
```

有关 `create-endpoint` API 的完整语法，请参阅 `create-endpoint`。

使用控制台部署已编译的模型

你必须满足先决条件部分是否使用编译模型Amazon SDK for Python (Boto3)，Amazon CLI，或者亚马逊 SageMaker 控制台。按照以下步骤使用 SageMaker 控制台创建和部署 SageMaker 新编译模型https://console.aws.amazon.com/ SageMaker.

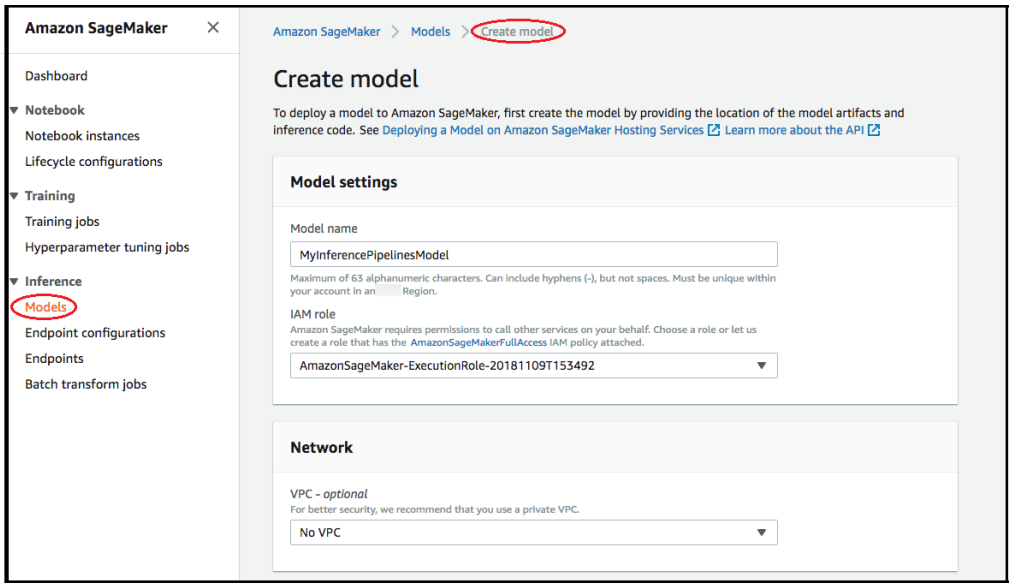
主题

- 部署模型 (p. 2355)

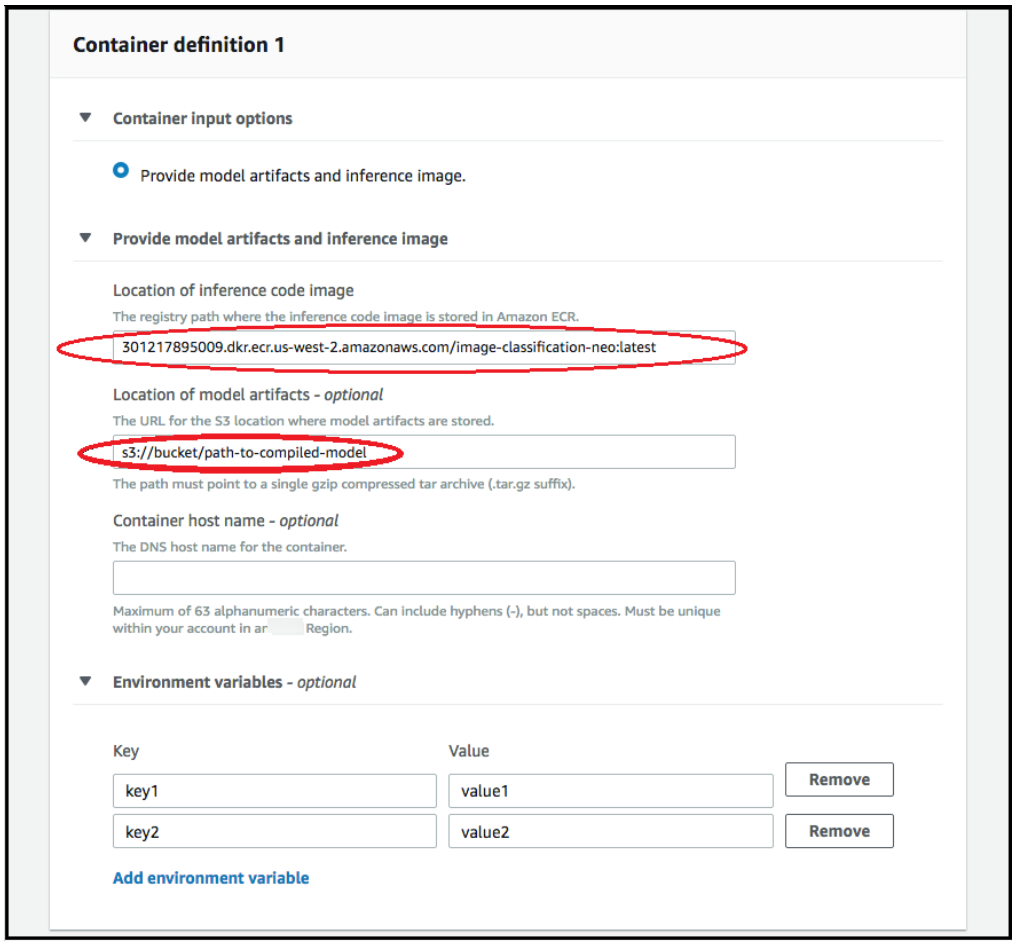
部署模型

在你满意之后先决条件中，使用以下步骤部署使用 Neo 编译的模型：

1. 选择模型，然后选择创建模型来自 的推理组中)。在存储库的创建模型页面，完成模型名称、IAM 角色，和VPC如果需要，字段（可选）。



2. 要添加有关用于部署模型的容器的信息，请选择添加容器容器，然后选择下一步. 填写 Container input options (容器输入选项)、Location of inference code image (推理代码映像位置) 和 Location of model artifacts (模型构件位置) 以及可选的 Container host name (容器主机名) 和 Environmental variables (环境变量) 字段。



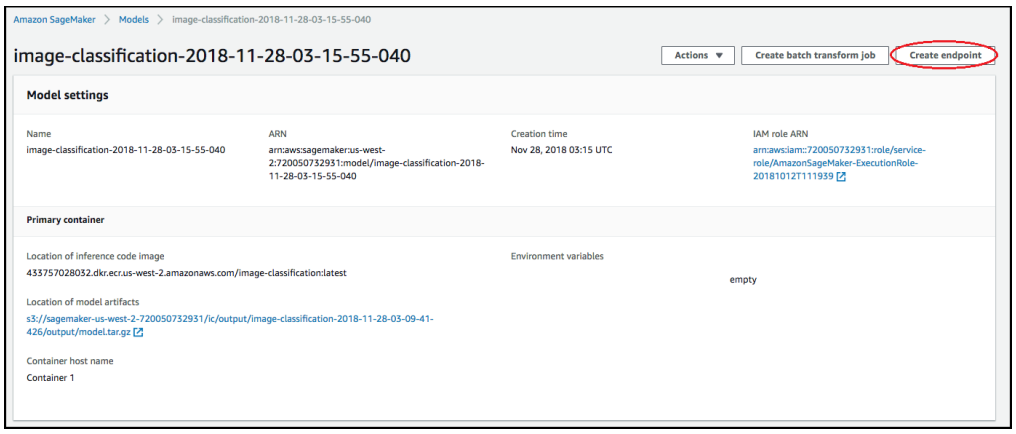
3. 要部署 Neo 编译的模型，请选择以下内容：

   - 容器输入选项：选择提供模型构件和推理映像.
   - 推理代码图像的位置：从中选择推理图像 URINeo 推理容器映像，具体取决于Amazon地区和应用程序的种类。
   - 模型神器的位置：输入 Neo 编译 API 生成的已编译模型构件的 Amazon S3 存储桶 URI。
   - 环境变量：
     - 将此字段留空SageMaker XGBoost.
     - 如果您使用 SageMaker 训练了模型，请指定环境变量`SAGEMAKER_SUBMIT_DIRECTORY`作为包含训练脚本的 Amazon S3 存储桶 URI。
     - 如果您没有使用 SageMaker 训练模型，请指定以下环境变量：

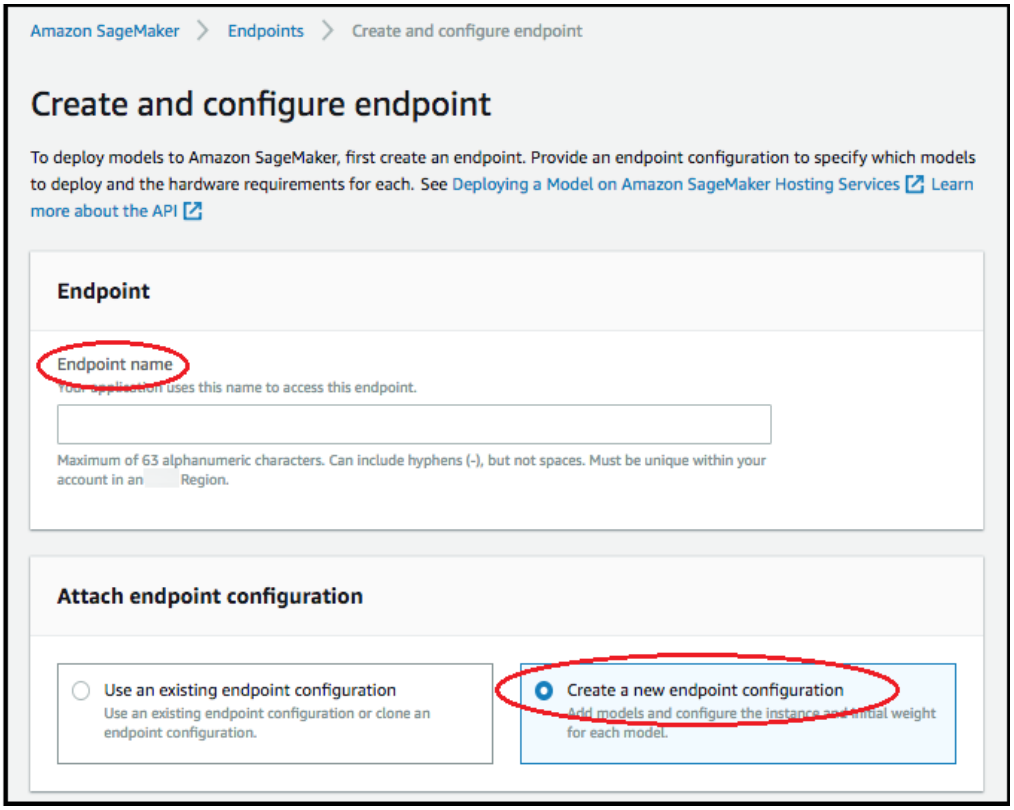
密钥	MxNet 和 PyTorch 的值	TensorFlow
SAGEMAKER_计划	inference.py	inference.py
SAGEMAKER_SUBMIT_目录	/opt/ml/模型/代码	/opt/ml/模型/代码
SAGEMAKER_容器_LOG_LEVEL	20	20
SAGEMAKER_区域	<your region>	<your region>

密钥	MxNet 和 PyTorch 的值	TensorFlow
MMS_DEFAULT_响应 SE_TIMEOUT	500	将此字段留空对于 TF

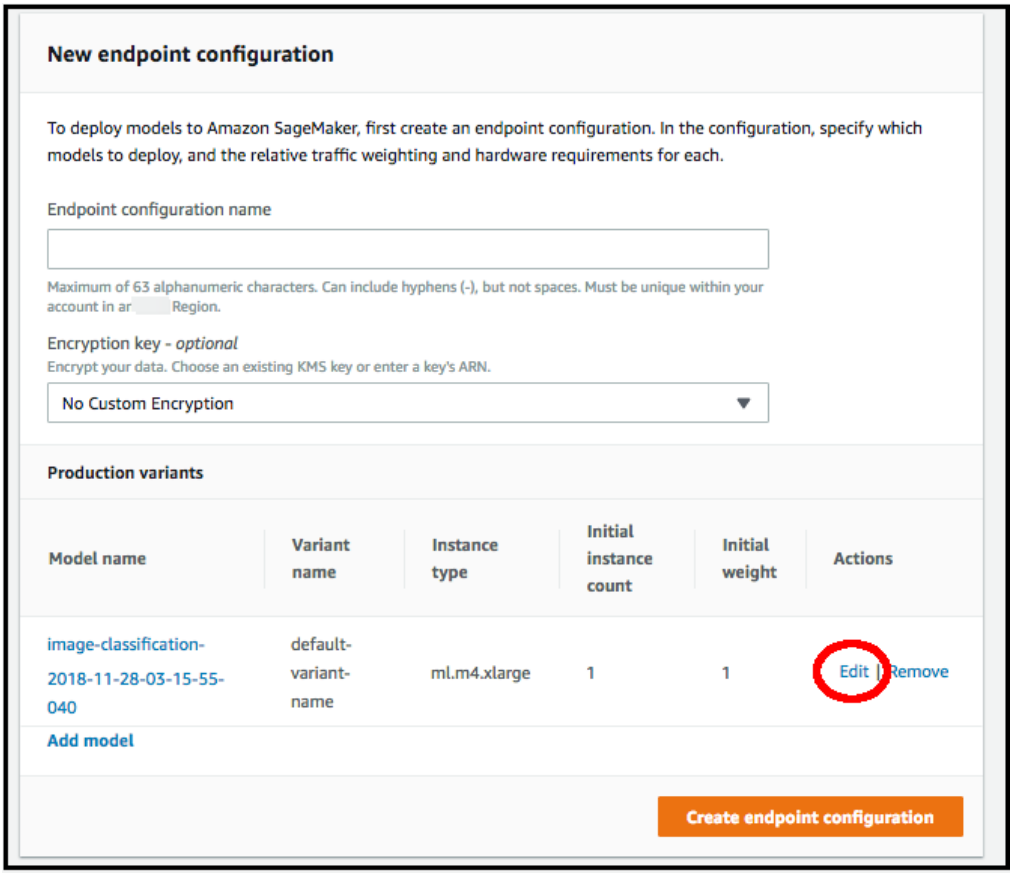
4. 确认容器的信息准确无误，然后选择 Create model (创建模型)。在存储库的创建模型登录页面，选择创建端节点.



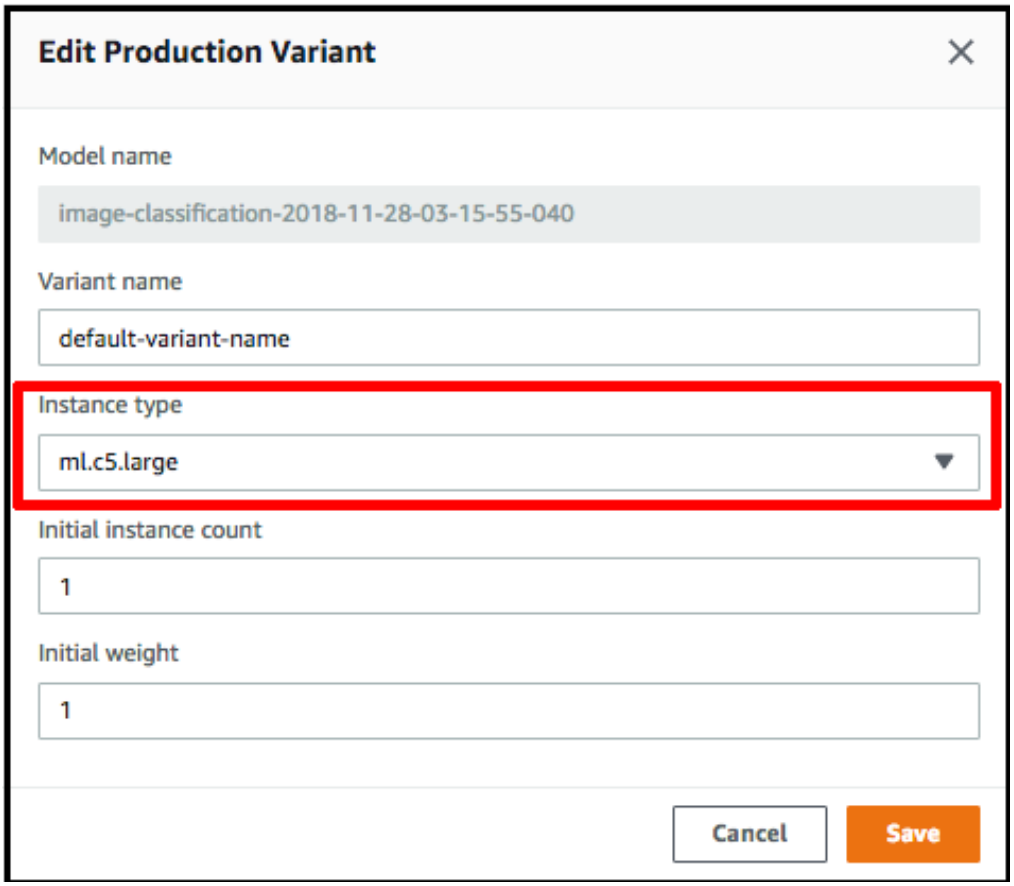
5. 在 Create and configure endpoint (创建和配置终端节点) 图表中，指定 Endpoint name (终端节点名称)。适用于附加终端节点配，选择创建新终端节点配置.



6. 在 New endpoint configuration (新建终端节点配置) 页面中，指定 Endpoint configuration name (终端节点配置名称)。



7. 选择编辑在模型名称旁边并指定正确的实例类型在编辑生产变体页. Instance type (实例类型) 值必须与在编译作业中指定的值匹配。



8. 选择 Save（保存）。
9. 在存储库的新终端节点配置页面，选择创建端节点配置，然后选择创建端节点.

请求从已部署服务推理

如果你遵循了中的说明部署模型 (p. 2342)请求您将 SageMaker 终端节点设置和运行。无论你如何部署新编译模型，都可以通过三种方式提交推理请求：

主题

- 请求从已部署服务推理（Amazon SageMaker SDK） (p. 2359)
- 请求从已部署服务推理 (Boto3) (p. 2361)
- 请求从已部署服务推理 (AmazonCLI) (p. 2362)

请求从已部署服务推理（Amazon SageMaker SDK）

使用以下代码示例，根据用于训练模型的框架，请求部署的服务进行推断。不同框架的代码示例类似。主要区别在于 TensorFlow 需要`application/json`作为内容类型。

PyTorch 和 MXNet

如果您正在使用PyTorch v1.4 或更高版本要么MXNet 1.7.0 或更高版本而且你有一个 Amazon SageMaker 终端节点`InService`，你可以使用`predictor`适用于 Python 的 SageMaker 开发工具包。

Note

API 取决于适用于 Python 的 SageMaker 开发工具包版本的不同：

- 对于 1.x 版本，请使用`RealTimePredictor`和`Predict`API。
- 对于版本 2.x，请使用`Predictor`和`Predict`API。

下面的代码示例演示了如何使用这些 API 发送图像进行推理：

SageMaker Python SDK v1.x

```
from sagemaker.predictor import RealTimePredictor

endpoint = 'insert name of your endpoint here'

# Read image into memory
payload = None
with open("image.jpg", 'rb') as f:
    payload = f.read()

predictor = RealTimePredictor(endpoint=endpoint, content_type='application/x-image')
inference_response = predictor.predict(data=payload)
print (inference_response)
```

SageMaker Python SDK v2.x

```
from sagemaker.predictor import Predictor

endpoint = 'insert name of your endpoint here'

# Read image into memory
payload = None
with open("image.jpg", 'rb') as f:
    payload = f.read()

predictor = Predictor(endpoint)
inference_response = predictor.predict(data=payload)
print (inference_response)
```

TensorFlow

下面的代码示例演示了如何使用 SageMaker Python SDK API 发送图像进行推理：

```
from sagemaker.predictor import Predictor
from PIL import Image
import numpy as np
import json

endpoint = 'insert the name of your endpoint here'

# Read image into memory
image = Image.open(input_file)
batch_size = 1
image = np.asarray(image.resize((224, 224)))
image = image / 128 - 1
image = np.concatenate([image[np.newaxis, :, :]] * batch_size)
body = json.dumps({"instances": image.tolist()})

predictor = Predictor(endpoint)
inference_response = predictor.predict(data=body)
```

```
print(inference_response)
```

请求从已部署服务推理 (Boto3)

您可以使用适用于 Python 的 SageMaker 开发工具包 (Boto3) 客户端和`invoke_endpoint()`拥有 SageMaker 终端节点后的 API`InService`. 下面的代码示例演示了如何发送图像进行推理：

PyTorch and MXNet

```
import boto3

import json

endpoint = 'insert name of your endpoint here'

runtime = boto3.Session().client('sagemaker-runtime')

# Read image into memory
with open(image, 'rb') as f:
    payload = f.read()
# Send image via InvokeEndpoint API
response = runtime.invoke_endpoint(EndpointName=endpoint, ContentType='application/x-
image', Body=payload)

# Unpack response
result = json.loads(response['Body'].read().decode())
```

TensorFlow

对于 TensorFlow，请使用`application/json`对于内容类型。

```
from PIL import Image
import numpy as np
import json
import boto3

client = boto3.client('sagemaker-runtime')
input_file = 'path/to/image'
image = Image.open(input_file)
batch_size = 1
image = np.asarray(image.resize((224, 224)))
image = image / 128 - 1
image = np.concatenate([image[np.newaxis, :, :]] * batch_size)
body = json.dumps({"instances": image.tolist()})
ioc_predictor_endpoint_name = 'insert name of your endpoint here'
content_type = 'application/json'
ioc_response = client.invoke_endpoint(
    EndpointName=ioc_predictor_endpoint_name,
    Body=body,
    ContentType=content_type
 )
```

XGBoost

对于 XGBoost 应用程序，您应改为提交 CSV 文本：

```
import boto3
import json

endpoint = 'insert your endpoint name here'

runtime = boto3.Session().client('sagemaker-runtime')
```

```
csv_text = '1,-1.0,1.0,1.5,2.6'
# Send CSV text via InvokeEndpoint API
response = runtime.invoke_endpoint(EndpointName=endpoint, ContentType='text/csv',
 Body=csv_text)
# Unpack response
result = json.loads(response['Body'].read().decode())
```

请注意，BYOM 允许自定义内容类型。有关更多信息，请参阅 `runtime_InvokeEndpoint`。

请求从已部署服务推理 (AmazonCLI)

推理请求可以使用`sagemaker-runtime invoke-endpoint`一旦你有了 Amazon SageMaker 终端节点`InService`. 你可以使用Amazon Command Line Interface(Amazon CLI)。下面的示例演示了如何发送图像进行推理：

```
aws sagemaker-runtime invoke-endpoint --endpoint-name 'insert name of your endpoint here'
 --body fileb://image.jpg --content-type=application/x-image output_file.txt
```

网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`output_file.txt`如果推理成功，就会提出有关你的推理请求的信息。

对于 TensorFlow，请使用`application/json`作为内容类型。

```
aws sagemaker-runtime invoke-endpoint --endpoint-name 'insert name of your endpoint here'
 --body fileb://input.json --content-type=application/json output_file.txt
```

推理容器映像

根据您的使用案例，将下面提供的推理图像 URI 模板中突出显示的部分替换为适当的值。

亚马逊SageMakerXGBoost

```
aws_account_id.dkr.ecr.aws_region.amazonaws.com/xgboost-neo:latest
```

Replace*aws_account_id*从本页末尾的表中，基于*aws_region*你用过。

Keras

```
aws_account_id.dkr.ecr.aws_region.amazonaws.com/sagemaker-neo-keras:fx_version-
instance_type-py3
```

Replace*aws_account_id*从本页末尾的表中，基于*aws_region*你用过。

Replace*fx_version*和`2.2.4`.

Replace*instance_type*或者`cpu`要么`gpu`.

MXNet

CPU or GPU instance types

```
aws_account_id.dkr.ecr.aws_region.amazonaws.com/sagemaker-inference-
mxnet:fx_version-instance_type-py3
```

Replace*aws_account_id*从本页末尾的表中，基于*aws_region*你用过。

Replace*fx_version*和`1.8.0`.

Replace*instance_type*或者cpu要么gpu.

Inferentia instance types

```
aws_account_id.dkr.ecr.aws_region.amazonaws.com/sagemaker-neo-
mxnet:fx_version-instance_type-py3
```

Replace*aws_region*或者us-east-1要么us-west-2.

Replace*aws_account_id*从本页末尾的表中，基于*aws_region*你用过。

Replace*fx_version*和1.5.1.

将 *instance_type* 替换为 inf。

ONNX

```
aws_account_id.dkr.ecr.aws_region.amazonaws.com/sagemaker-neo-onnx:fx_version-
instance_type-py3
```

Replace*aws_account_id*从本页末尾的表中，基于*aws_region*你用过。

Replace*fx_version*和1.5.0.

Replace*instance_type*或者cpu要么gpu.

PyTorch

CPU or GPU instance types

```
aws_account_id.dkr.ecr.aws_region.amazonaws.com/sagemaker-inference-
pytorch:fx_version-instance_type-py3
```

Replace*aws_account_id*从本页末尾的表中，基于*aws_region*你用过。

Replace*fx_version*和1.4、1.5、1.6、1.7要么1.8.

Replace*instance_type*或者cpu要么gpu.

Inferentia instance types

```
aws_account_id.dkr.ecr.aws_region.amazonaws.com/sagemaker-neo-
pytorch:fx_version-instance_type-py3
```

Replace*aws_region*或者us-east-1要么us-west-2.

Replace*aws_account_id*从本页末尾的表中，基于*aws_region*你用过。

Replace*fx_version*和1.5.1.

将 *instance_type* 替换为 inf。

TensorFlow

CPU or GPU instance types

```
aws_account_id.dkr.ecr.aws_region.amazonaws.com/sagemaker-inference-
tensorflow:fx_version-instance_type-py3
```

Replace*aws_account_id*从本页末尾的表中，基于*aws_region*你用过。

Replace*fx_version*和`1.15.3`要么`2.4`.

Replace*instance_type*或者`cpu`要么`gpu`.

Inferentia instance types

```
aws_account_id.dkr.ecr.aws_region.amazonaws.com/sagemaker-neo-
tensorflow:fx_version-instance_type-py3
```

Replace*aws_account_id*从本页末尾的表中，基于*aws_region*你用过。请注意，例如实例类型`inf`仅限`us-east-1`和`us-west-2`支持。

Replace*fx_version*和`1.15.0`

Replace*instance_type*和`inf`.

下表地图*aws_account_id*和*aws_region*. 使用此表查找应用程序所需的正确推理图像 URI。

aws_account_id	aws_region
785573368785	us-east-1
007439368137	us-east-2
710691900526	us-west-1
301217895009	us-west-2
802834080501	eu-west-1
205493899709	eu-west-2
254080097072	eu-west-3
601324751636	eu-north-1
966458181534	eu-south-1
746233611703	eu-central-1
110948597952	ap-east-1
763008648453	ap-south-1
941853720454	ap-northeast-1
151534178276	ap-northeast-2
324986816169	ap-southeast-1
355873309152	ap-southeast-2
474822919863	cn-northwest-1
472730292857	cn-north-1
756306329178	sa-east-1
464438896020	ca-central-1
836785723513	me-south-1

aws_account_id	aws_region
774647643957	af-south-1

Edge 设备

Amazon SageMaker Neo 为流行的机器学习框架提供编译支持。您可以部署新编译的边缘设备，例如树莓派 3、德州仪器的 Sitara、Jetson TX1 等。有关支持的框架和 Edge 设备的完整列表，请参阅支持的框架、设备、系统和架构.

您必须配置边缘设备才能使用Amazon服务。实现此目标的一种方法是在设备上安装 DLR 和 Boto3。要执行此操作，必须设置身份验证凭据。请参阅Boto3Amazon配置有关. 编译模型并配置了边缘设备后，您可以将模型从 Amazon S3 下载到边缘设备。从那里，你可以使用深度学习运行时 (DLR)阅读编译后的模型并作出推论。

对于初次使用的用户，我们建议您查看开始使用指南。本指南将向您介绍如何设置凭据、编译模型、将模型部署到树莓派 3 以及对图像进行推论。

主题

- 支持的框架、设备、系统和架构 (p. 2365)
- 部署模型 (p. 2375)
- Edge 设备上的 Neo 入门 (p. 2375)

支持的框架、设备、系统和架构

亚马逊 SageMaker Neo 支持常见的机器学习框架、边缘设备、操作系统和芯片架构。通过选择以下主题之一，了解 Neo 是否支持您的框架、边缘设备、操作系统和芯片架构。

你可以找到经过亚马逊测试的型号清单 SageMaker 中的 Neo 团队测试模型 (p. 2369)部分。

Note

- Ambarella 设备需要在压缩的 TAR 文件中包含其他文件，然后才能将其发送进行编译。有关更多信息，请参阅 解决 Amparlla 错误 (p. 2385)。
- i.mx 8M Plus 需要 TIM-VX (libtim-vx.so)。有关如何构建 TIM-VX 的信息，请参阅TIM-VX GitHub 知识库.

主题

- 支持的框架 (p. 2365)
- 支持的设备、芯片架构和系统 (p. 2366)
- 测试模型 (p. 2369)

支持的框架

亚马逊 SageMaker Neo 支持以下框架。

框架	框架版本	模型版本	模型	模型格式（包装在 *.tar.gz 中）	工具箱
MXNet	1.8.0	支持 1.8.0 或更高版本	图像分类、目标检测、语义分割、姿势估计、活动识别	一个符号文件 (.json) 和一个参数文件 (.params)	GluonCV v0.8.0

框架	框架版本	模型版本	模型	模型格式（包装在 *.tar.gz 中）	工具箱
ONNX	1.7.0	支持 1.7.0 或更高版本	SVM 映像分类	一个模型文件 (.onnx)	
Keras	2.2.4	支持 2.2.4 或更高版本	图像分类	一个模型定义文件 (.h5)	
PyTorch	1.7.1、1.8.1	支持 1.7.1、1.8.1 或更高版本	图像分类、物体检测	一个模型定义文件 (.pth)	
TensorFlow	1.15.0、2.4.2	支持 1.15.0、2.4.2 或更高版本	图像分类、物体检测	* 对于保存的模型，需要一个 .pb 或一个 .pbtxt 文件，以及包含变量的变量目录 * 对于冻结的模型，只有一个 .pb 或 .pbtxt 文件。	
TensorFlow-精简版	1.15.2	支持 1.15.2 或更高版本	图像分类、物体检测	一个模型定义平缓冲区文件 (.tflite)	
XGBoost	1.3.3	支持 1.3.3 或更高版本	决策树	一个 XGBoost 模型文件 (.model)，其中树的节点数小于 2^31 个	
暗网			图像分类、物体检测	一个配置 (.cfg) 文件和一个权重 (.weight) 文件	

支持的设备、芯片架构和系统

亚马逊 SageMaker Neo 支持以下设备、芯片架构和操作系统。

设备

您可以使用亚马逊 SageMaker 控制台或者通过指定`TargetDevice`在的输出配置中`CreateCompilationJob`API。

可以从以下边缘设备中选择一个：

设备列表	芯片上系统 (SoC)	操作系统	架构	Accelerator	示例编译器选项
Aisage		Linux	ARM64	马里	
amba_cv2	CV2	Arch Linux	ARM64	cvflow	
amba_cv22	CV22	Arch Linux	ARM64	cvflow	
amba_cv25	CV25	Arch Linux	ARM64	cvflow	

设备列表	芯片上系统 (SoC)	操作系统	架构	Accelerator	示例编译器选项
coreml		iOS、macOS			`{"class_labels": "imagenet_labels_1000.`
深入	Intel 凌动	Linux	X86_64	英特尔显卡	
imx8qm	恩智浦 imx8	Linux	ARM64		
imx8mplus	i.mx 8M Plus	Linux	ARM64	NPU	
jacinto_tda4vm	TDA4VM	Linux	手臂	TDA4VM	
jetson_nano		Linux	ARM64	NVIDIA	`{'gpu-code': 'sm_53', 'trt-ver': '5.0.6', 'cuda-ver': '10.0'}`
jetson_tx1		Linux	ARM64	NVIDIA	`{'gpu-code': 'sm_53', 'trt-ver': '6.0.1', 'cuda-ver': '10.0'}`
jetson_tx2		Linux	ARM64	NVIDIA	`{'gpu-code': 'sm_62', 'trt-ver': '6.0.1', 'cuda-ver': '10.0'}`
jetson_xavier		Linux	ARM64	NVIDIA	`{'gpu-code': 'sm_72', 'trt-ver': '5.1.6', 'cuda-ver': '10.0'}`
qcs605		Android	ARM64	马里	`{'ANDROID_PLATFORM': 27}`
qcs603		Android	ARM64	马里	`{'ANDROID_PLATFORM': 27}`
asp3b	手臂 A56	Linux	ARM_EABIHF		`{'mattr': ['+neon']}`
asp4	手臂 A72				
rk3288		Linux	ARM_EABIHF	马里	
rk3399		Linux	ARM64	马里	

设备列表	芯片上系统 (SoC)	操作系统	架构	Accelerator	示例编译器选项
sbe_c		Linux	x86_64		`{'mcpu': 'core-avx2'}`
sitara_am57x	AM57X	Linux	ARM64	EVE 和/或 c66x DSP	
x86_win32		Windows 10	X86_32		
x86_win64		Windows 10	X86_32		

有关每个目标设备的 JSON 键值编译器选项的更多信息，请参阅`CompilerOptions`字段中的`OutputConfig`API数据类型。

系统和芯片架构

以下查询表提供了有关 Neo 模型编译作业的可用操作系统和架构的信息。

Linux

	X86_64	X86	ARM64	ARM_EABIHF	ARM_EABI
没有加速器（CPU）	X		X	X	X
NVIDIA GPU	X		X		
英特尔 _ 图形	X				
ARM 马里			X	X	X

Android

	X86_64	X86	ARM64	ARM_EABIHF	ARM_EABI
没有加速器（CPU）	X	X	X		X
NVIDIA GPU					
英特尔 _ 图形	X	X			
ARM 马里			X		X

Windows

	X86_64	X86	ARM64	ARM_EABIHF	ARM_EABI
没有加速器（CPU）	X	X			

测试模型

以下可折叠部分提供有关亚马逊测试的机器学习模型的信息 SageMaker Neo 团队。根据您的框架展开可折叠部分，以检查模型是否已测试。

> Note
>
> 这是不全面的可以用 Neo 编译的模型列表。

请参阅支持的框架 (p. 2365)和SageMaker Neo 支持的操作找出你是否可以用来编译你的模型 SageMaker Neo。

DarkNet

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panorar	TI TDA4VM	高通 QCS603	X86_Lin	x86_ 视窗		
AlexNet											
Resnet50	X	X		X	X	X		X	X		
yoLov2				X	X	X		X	X		
yolov2_Tiny	X	X		X	X	X		X	X		
YOLOV3_416				X	X	X		X	X		
yolov3_Tiny	X	X		X	X	X		X	X		

MXNet

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panorama	TI TDA4VM	高通 QCS603	X86_Linux	x86_ 视窗
AlexNet			X						
Densenet121			X						
Densenet201	X	X	X	X	X	X		X	X
GoogleNet	X	X		X	X	X		X	X
Inceptionv3				X	X	X		X	X
MobileNet 0.75	X	X		X	X	X			X
MobileNet 1.0	X	X	X	X	X	X			X
Mobilenetv2_0.5	X	X		X	X	X			X
Mobilenetv2_1.0	X	X	X	X	X	X	X	X	X
MobilenetV3_Lar	X	X	X	X	X	X	X	X	X
MobileNet V3_Small	X	X	X	X	X	X	X	X	X

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panorama	TI TDA4VM	高通 QCS603	X86_Linux	x86_ 视窗
Resnest50				X	X			X	X
resnet18_v1	X	X	X	X	X	X			X
resnet18_v2	X	X		X	X	X			X
resnet50_v1	X	X	X	X	X	X		X	X
resnet50_v2	X	X	X	X	X	X		X	X
resnext101_32x4D									
resnext50_32x4D			X	X	X			X	X
senet_154				X	X	X		X	X
SE_resnext50_32x4D	X			X	X	X		X	X
squeezenet1.0	X	X	X	X	X	X			X
squeezenet1.1	X	X	X	X	X	X		X	X
VGG11	X	X	X	X	X			X	X
例外	X	X	X	X	X	X		X	X
暗网 53	X	X		X	X	X		X	X
resnet18_v1b_0.89	X	X		X	X	X			X
resnet50_v1d_0.11	X	X		X	X	X			X
resnet50_v1d_0.86	X	X	X	X	X	X		X	X
ssd_512_mobilenet1.0_coco	X	X	X	X	X	X		X	X
ssd_512_mobilenet1.0_voc	X	X	X	X	X	X		X	X
ssd_resnet50_v1	X		X	X	X			X	X
yolo3_darknet53_coco	X			X	X			X	X
yolo3_mobilenet1.0_coco	X	X		X	X	X		X	X
deeplab_resnet50			X						

Keras

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panorama	TI TDA4VM	高通 QCS603	X86_Linux	x86_ 视窗
densenet121	X	X	X	X	X	X		X	X
densenet201	X	X	X	X	X	X			X
Ineption_v3	X	X		X	X	X		X	X
mobilenet_v1	X	X	X	X	X	X		X	X

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panorama	TI TDA4VM	高通 QCS603	X86_Linux	x86_ 视窗
mobilenet_v2	X	X	X	X	X	X		X	X
resnet152_v1				X	X				X
resnet152_v2				X	X				X
resnet50_v1	X	X	X	X	X			X	X
resnet50_v2	X	X	X	X	X	X		X	X
vgg16			X	X	X			X	X

ONNX

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panorama	TI TDA4VM	高通 QCS603	X86_Linux	x86_ 视窗
Alexnet			X						
Mobilenetv2	X-1.0	X	X	X	X	X		X	X
resnet18v1	X			X	X				X
resnet18v2	X			X	X				X
resnet50v1	X		X	X	X			X	X
resnet50v2	X		X	X	X			X	X
resnet152v1				X	X	X			X
resnet152v2				X	X	X			X
squeezenet	X1.1		X	X	X	X		X	X
vgg19			X						X

pyTorch (FP32)

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	安巴雷拉 CV25	NVIDIA	Panoram	TI TDA4VM	高通 QCS603	X86_Linu	x86_ 视窗
densenet121	X	X	X	X	X	X	X		X	X
Ineption_v3		X			X	X	X		X	X
resnet152					X	X	X			X
resnet18	X	X			X	X	X			X
Resnet50	X	X	X	X	X	X			X	X
squeezenet	X1.0	X			X	X	X			X

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	安巴雷拉 CV25	NVIDIA	Panoram	TI TDA4VM	高通 QCS603	X86_Linu	x86_ 视窗
squeezenet1.1	X	X	X	X	X	X	X		X	X
yolov4					X	X				
yolov5				X	X	X				
fasterrcnn_resnet50_fpn					X	X				
maskrcnn_resnet50_fpn					X	X				

TensorFlow

TensorFlow

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	安巴雷拉 CV25	NVIDIA	Panorama	TI TDA4VM	高通 QCS	X86_	x86_x 视窗
densenet201	X	X	X	X	X	X	X		X	X
Ineption_v3	X	X	X		X	X	X		X	X
mobilenet100_v1	X	X	X		X	X	X			X
mobilenet100_v2.0	X	X	X		X	X	X		X	X
mobilenet130_v2	X	X			X	X	X			X
mobilenet140_v2	X	X	X		X	X	X		X	X
resnet50_v1.5	X	X			X	X	X		X	X
resnet50_v2	X	X	X	X	X	X	X		X	X
ResezenetX	X	X	X	X	X	X	X		X	X
mask_rcnn_ception_resnet_v2					X					
ssd_mobilenet_v2					X	X				
faster_rcnn_resnet50_低建议					X					
rfcn_resnet101					X					

TensorFlow.Keras

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panorama	TI TDA4VM	高通 QCS	X86_	x86_ 视窗
Densenet121	X	X		X	X	X		X	X
Densenet201	X	X		X	X	X			X

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panorama	TI TDA4VM	高通 QCS	X86_	x86_ 视窗
Inceptionv3	X	X		X	X	X		X	X
MobileNet	X	X		X	X	X		X	X
MobileNetv2	X	X		X	X	X		X	X
NasnetLarge				X	X			X	X
NasnetMobile	X	X		X	X	X		X	X
resnet101				X	X	X			X
resnet101v2				X	X	X			X
resnet152				X	X				X
resnet152v2				X	X				X
ResNet50	X	X		X	X			X	X
resnet50v2	X	X		X	X	X		X	X
VGG16				X	X			X	X
例外	X	X		X	X	X		X	X

TensorFlow-精简版

TensorFlow-Lite (FP32)

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panoran	TI TDA4VM	高通 QCS603	X86_Lin	x86_ 视窗	i.mx 8M Plus
densenet_2018_04_27	X			X	X	X			X	
启动 _resnet_v2_2018_04_27				X	X	X			X	
启动 _v3_2018_04_27				X	X	X			X	X
启动 _v4_2018_04_27				X	X	X			X	X
mnasnet_0.5_224_09_09_07_2018	X			X	X	X			X	
mnasnet_1.0_224_09_07_2018	X			X	X	X			X	
mnasnet_1.3_224_09_07_2018	X			X	X	X			X	
mobilenet_v1_0.25_128	X			X	X	X			X	X
mobilenet_v1_0.25_224	X			X	X	X			X	X
mobilenet_v1_0.5_128	X			X	X	X			X	X

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panoran	TI TDA4VM	高通 QCS603	X86_Lin	x86_ 视窗	i.mx 8M Plus
mobilenet_v1_0.5_224	X			X	X	X			X	X
mobilenet_v1_0.75_128	X			X	X	X			X	X
mobilenet_v1_0.75_224	X			X	X	X			X	X
mobilenet_v1_1.0_128	X			X	X	X			X	X
mobilenet_v1_1.0_192	X			X	X	X			X	X
mobilenet_v2_1.0_224	X			X	X	X			X	X
resnet_v2_101				X	X	X			X	
squeezenet_2018_04_27	X			X	X	X			X	

TensorFlow-Lite (INT8)

模型	手臂 V8	ARM 马里	安巴雷拉 CV22	NVIDIA	Panoran	TI TDA4VM	高通 QCS603	X86_Lin	x86_ 视窗	i.mx 8M Plus
Ineption_v1							X			X
初创_v2							X			X
Ineption_v3	X					X	X		X	X
启动_v4_299	X					X	X		X	X
mobilenet_v1_0.25_128	X					X			X	X
mobilenet_v1_0.25_224	X					X			X	X
mobilenet_v1_0.5_128	X					X			X	X
mobilenet_v1_0.5_224	X					X			X	X
mobilenet_v1_0.75_128	X					X			X	X
mobilenet_v1_0.75_224	X					X	X		X	X
mobilenet_v1_1.0_128	X					X			X	X
mobilenet_v1_1.0_224	X					X	X		X	X
mobilenet_v2_1.0_224	X					X	X		X	X
deeplab-v3_513							X			

部署模型

您可以通过以下方式将计算模块部署到资源受限的边缘设备：将编译后的模型从 Amazon S3 下载到您的设备并使用DLR，或者您可以使用。AmazonIoT Greengrass.

继续之前，请确保边缘设备必须受 SageMaker Neo 的支持。请参阅,支持的框架、设备、系统和架构以了解支持哪些边缘设备。确保在提交编译作业时指定了目标边缘设备，请参阅使用 Neo 编译模型.

部署编译模型 (DLR)

DLR是深度学习模型和决策树模型的紧凑、常见的运行时间。DLR 使用TVM运行时间，Treelite运行时 NVIDIA TensorRT™，可以包括其他特定于硬件的运行时。DLR 提供了统一的 Python/C++ API，用于在各种设备上加载和运行编译后的模型。

您可以使用以下 pip 命令安装最新版本的 DLR 软件包：

```
pip install dlr
```

要在 GPU 目标或非 x86 边缘设备上安装 DLR，请参阅版本对于预构建的二进制文件，或者安装 DLR用于从源头构建 DLR。例如，要为树莓派 3 安装 DLR，您可以使用：

```
pip install https://neo-ai-dlr-release.s3-us-west-2.amazonaws.com/v1.3.0/pi-armv7l-
raspbian4.14.71-glibc2_24-libstdcpp3_4/dlr-1.3.0-py3-none-any.whl
```

部署模型 (AmazonIoT Greengrass)

AmazonIoT Greengrass将云功能扩展到本地设备中。它使得设备可以更靠近信息源来收集和分析数据，自主响应本地事件，同时在本地网络上彼此安全地通信。与Amazon借助 IoT Greengrass，您可以使用云训练模型对本地生成的数据在边缘站点执行机器学习推理。目前，您可以将模型部署到所有AmazonIoT Greengrass 设备基于 ARM Cortex-A、Intel Atom 和 Nvidia Jetson 系列处理器。有关部署 Lambda 推理应用程序以执行机器学习推理的更多信息Amazon请参阅 IoT Greengrass如何使用 配置优化的机器学习推理Amazon管理控制台.

Edge 设备上的 Neo 入门

本亚马逊入门指南 SageMaker Neo 向您展示了如何编译模型、设置设备以及在设备上做出推论。大多数代码示例都使用 Boto3。我们提供命令使用Amazon CLI以及关于如何满足 Neo 先决条件的说明。

> **Note**
>
> 您可以在本地计算机上运行以下代码片段 SageMaker 笔记本， SageMaker Studio，或者（取决于边缘设备）在边缘设备上。设置类似；但是，如果在 SageMaker 笔记本实例 SageMaker 演播室会议：
>
> - 您不需要安装 Boto3。
> - 您不需要添加‘AmazonSageMakerFullAccess’IAM 策略

本指南假设您正在边缘设备上运行以下说明。

先决条件

1. 安装 Boto3

   如果您在边缘设备上运行这些命令，则必须安装Amazon SDK for Python (Boto3). 在 Python 环境（最好是虚拟环境）中，在边缘设备的终端或在 Jupyter 笔记本实例中本地运行以下命令：

Terminal

```
pip install boto3
```

Jupyter Notebook

```
!pip install boto3
```

2. 设置Amazon凭证

   您需要在设备上设置 Amazon Web Services 凭证才能运行 SDK for Python (Boto3)。默认为，Amazon 凭据应存储在文件中`~/.aws/credentials`在边缘设备上。在凭据文件中，您应该看到两个环境变量：`aws_access_key_id`和`aws_secret_access_key`.

   在您的终端中，运行：

   ```
   $ more ~/.aws/credentials

   [default]
   aws_access_key_id = YOUR_ACCESS_KEY
   aws_secret_access_key = YOUR_SECRET_KEY
   ```

   这些区域有：Amazon一般参考指南有了有关如何获得必要物品的说明`aws_access_key_id`和`aws_secret_access_key`. 有关如何在您的设备上设置凭据的更多信息，请参阅Boto3文档中）。

3. 设置 IAM 角色并附加策略。

   Neo 需要访问您的 S3 存储桶 URI。创建一个可以运行 SageMaker 并有权访问 S3 URI 的 IAM 角色。您可以通过使用适用于 Python (Boto3) 的软件开发工具包 (Boto3)、控制台或Amazon CLI. 以下示例说明了如何使用适用于 Python (Boto3) 的开发工具包创建 IAM 角色：

   ```
   import boto3

   AWS_REGION = 'aws-region'

   # Create an IAM client to interact with IAM
   iam_client = boto3.client('iam', region_name=AWS_REGION)
   role_name = 'role-name'
   ```

   有关如何使用控制台创建 IAM 角色的更多信息，Amazon CLI，或通过AmazonAPI，请参阅在您的中创建 IAM 用户Amazon帐户.

   创建描述您要附加的 IAM 策略的字典。此策略用于创建新的 IAM 角色。

   ```
   policy = {
       'Statement': [
           {
               'Action': 'sts:AssumeRole',
               'Effect': 'Allow',
               'Principal': {'Service': 'sagemaker.amazonaws.com'},
           }],
        'Version': '2012-10-17'
   }
   ```

   使用您上面定义的策略创建新的 IAM 角色：

```
import json

new_role = iam_client.create_role(
    AssumeRolePolicyDocument=json.dumps(policy),
    Path='/',
    RoleName=role_name
)
```

在后面的步骤中创建编译作业时，您需要知道您的亚马逊资源名称 (ARN) 是什么，因此也将其存储在变量中。

```
role_arn = new_role['Role']['Arn']
```

现在您已经创建了新角色，请附加与 Amazon 交互所需的权限。 SageMaker 和 Amazon S3：

```
iam_client.attach_role_policy(
    RoleName=role_name,
    PolicyArn='arn:aws:iam::aws:policy/AmazonSageMakerFullAccess'
)

iam_client.attach_role_policy(
    RoleName=role_name,
    PolicyArn='arn:aws:iam::aws:policy/AmazonS3FullAccess'
);
```

4. 创建 Amazon S3 存储桶以存储您的模型构件

   SageMaker Neo 将从 Amazon S3 访问您的模型工件

   Boto3

   ```
   # Create an S3 client
   s3_client = boto3.client('s3', region_name=AWS_REGION)

   # Name buckets
   bucket='name-of-your-bucket'

   # Check if bucket exists
   if boto3.resource('s3').Bucket(bucket) not in boto3.resource('s3').buckets.all():
       s3_client.create_bucket(
           Bucket=bucket,
           CreateBucketConfiguration={
               'LocationConstraint': AWS_REGION
           }
       )
   else:
       print(f'Bucket {bucket} already exists. No action needed.')
   ```

   CLI

   ```
   aws s3 mb s3://'name-of-your-bucket' --region specify-your-region

   # Check your bucket exists
   aws s3 ls s3://'name-of-your-bucket'/
   ```

5. 训练机器学习模型

   请参阅使用 Amazon SageMaker 训练模型有关如何使用 Amazon SageMaker 训练机器学习模型的更多信息。您可以选择将本地训练的模型直接上传到 Amazon S3 URI 存储桶中。

Note

确保模型格式正确，具体取决于您使用的框架。请参阅输入数据形状有什么作用 SageMaker Neo 期望？

如果你还没有模型，请使用curl命令以获取coco_ssd_mobilenet来自 TensorFlow 网站的模型。你刚复制的模型是从COCO 数据集. 在你的 Jupyter 笔记本中键入以下内容：

```
model_zip_filename = './coco_ssd_mobilenet_v1_1.0.zip'
!curl http://storage.googleapis.com/download.tensorflow.org/models/tflite/
coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip \
    --output {model_zip_filename}
```

请注意，此特定示例打包在 .zip 文件中。解压该文件并将其重新打包为压缩的 tarfile (.tar.gz) 后面的步骤中使用它。在你的 Jupyter 笔记本中键入以下内容：

```
# Extract model from zip file
!unzip -u {model_zip_filename}

model_filename = 'detect.tflite'
model_name = model_filename.split('.')[0]

# Compress model into .tar.gz so SageMaker Neo can use it
model_tar = model_name + '.tar.gz'
!tar -czf {model_tar} {model_filename}
```

6. 将训练过的模型上传到 S3 存储桶

   训练机器学习模式后，将其存储在 S3 存储桶中。

   Boto3

   ```
   # Upload model
   s3_client.upload_file(Filename=model_filename, Bucket=bucket, Key=model_filename)
   ```

   CLI

   Replaceyour-model-filename和your-S3-bucket使用您的 S3 存储桶的名称。

   ```
   aws s3 cp your-model-filename s3://your-S3-bucket
   ```

第 1 步：编译模型

一旦您满意了先决条件，您可以使用 Amazon SageMaker Neo 编译模型。你可以使用Amazon CLI、控制台或适用于 Python (Boto3) 的 Amazon Web Services 软件开发工具包，请参阅使用 Neo 编译模型. 在此示例中，您将使用 Boto3 编译模型。

要编译模型， SageMaker Neo 需要以下信息：

1. 存储训练模型的 Amazon S3 存储桶 URI。

   如果您遵循了先决条件，则存储桶的名称将存储在名为的变量中bucket. 以下代码段说明了如何使用 Amazon CLI：

   ```
   aws s3 ls
   ```

例如：

```
$ aws s3 ls
2020-11-02 17:08:50 bucket
```

2. 您希望将编译后的模型保存到的 Amazon S3 存储桶 URI。

   以下代码段将您的 Amazon S3 存储桶 URI 与名为的输出目录的名称连接起来。`output`：

   ```
   s3_output_location = f's3://{bucket}/output'
   ```

3. 你用来训练模型的机器学习框架。

   定义您用于训练模型的框架。

   ```
   framework = 'framework-name'
   ```

   例如，如果你想编译使用 TensorFlow 训练的模型，你可以使用`tflite`要么`tensorflow`. 使用`tflite`如果您希望使用更轻的版本 TensorFlow 这使用的存储内存更少。

   ```
   framework = 'tflite'
   ```

   有关 Neo 支持框架的完整列表，请参阅支持的框架、设备、系统和架构.

4. 模型输入的形状。

   Neo 需要输入张量的名称和形状。名称和形状以键值对的形式传递到。`value`是输入张量的整数维度的列表，`key`是模型中输入张量的确切名称。

   ```
   data_shape = '{"name": [tensor-shape]}'
   ```

   例如：

   ```
   data_shape = '{"normalized_input_image_tensor":[1, 300, 300, 3]}'
   ```

   > **Note**
   >
   > 确保模型格式正确，具体取决于您使用的框架。请参阅输入数据形状有什么作用 SageMaker Neo 期望？ 本字典中的键必须更改为新的输入张量的名称。

5. 要编译的目标设备的名称或硬件平台的一般详细信息

   ```
   target_device = 'target-device-name'
   ```

   例如，如果要部署到树莓派 3，请使用：

   ```
   target_device = 'rasp3b'
   ```

   您可以在中找到支持的边缘设备的完整列表支持的框架、设备、系统和架构.

现在你已经完成了前面的步骤，你可以向 Neo 提交编译作业了。

```
# Create a SageMaker client so you can submit a compilation job
sagemaker_client = boto3.client('sagemaker', region_name=AWS_REGION)
```

```
# Give your compilation job a name
compilation_job_name = 'getting-started-demo'
print(f'Compilation job for {compilation_job_name} started')

response = sagemaker_client.create_compilation_job(
    CompilationJobName=compilation_job_name,
    RoleArn=role_arn,
    InputConfig={
        'S3Uri': s3_input_location,
        'DataInputConfig': data_shape,
        'Framework': framework.upper()
    },
    OutputConfig={
        'S3OutputLocation': s3_output_location,
        'TargetDevice': target_device
    },
    StoppingCondition={
        'MaxRuntimeInSeconds': 900
    }
)

# Optional - Poll every 30 sec to check completion status
import time

while True:
    response =
 sagemaker_client.describe_compilation_job(CompilationJobName=compilation_job_name)
    if response['CompilationJobStatus'] == 'COMPLETED':
        break
    elif response['CompilationJobStatus'] == 'FAILED':
        raise RuntimeError('Compilation failed')
    print('Compiling ...')
    time.sleep(30)
print('Done!')
```

如果您想要进行调试的其他信息，请包括以下 print 语句：

```
print(response)
```

如果编译作业成功，您的编译模型将存储在您之前指定的输出 Amazon S3 存储桶中（`s3_output_location`）。在本地下载编译后的模型：

```
object_path = f'output/{model}-{target_device}.tar.gz'
neo_compiled_model = f'compiled-{model}.tar.gz'
s3_client.download_file(bucket, object_path, neo_compiled_model)
```

第 2 步：设置您的 设备

您需要在边缘设备上安装软件包，以便设备可以进行推断。你还需要安装AmazonIoT Greengrass核心或深度学习运行时 (DLR). 在此示例中，您将安装所需的软件包，以便为`coco_ssd_mobilenet`对象检测算法，你将使用 DLR。

1. 安装更多软件包

   除了 Boto3 之外，您还必须在边缘设备上安装某些库。安装哪些库取决于您的使用案例。

   例如，对于`coco_ssd_mobilenet`您之前下载的对象检测算法，您需要安装NumPy对于数据操作和统计数据，PIL加载图像，以及matplotlib来生成情节。您还需要一份 TensorFlow 如果你想衡量使用 Neo 编译与基线的影响。

```
!pip3 install numpy pillow tensorflow matplotlib
```

2. 在设备上安装推理引擎

   要运行你的新编译模型，请安装深度学习运行时 (DLR)在您的设备上。DLR 是深度学习模型和决策树模型的紧凑、常见的运行时间。在运行 Linux 的 x86_64 CPU 目标上，您可以使用以下命令安装最新版本的 DLR 软件包`pip`命令：

   ```
   !pip install dlr
   ```

   有关在 GPU 目标或非 x86 边缘设备上安装 DLR 的信息，请参阅版本对于预构建的二进制文件，或者安装 DLR用于从源头构建 DLR。例如，要为树莓派 3 安装 DLR，您可以使用：

   ```
   !pip install https://neo-ai-dlr-release.s3-us-west-2.amazonaws.com/v1.3.0/pi-armv7l-
   raspbian4.14.71-glibc2_24-libstdcpp3_4/dlr-1.3.0-py3-none-any.whl
   ```

第 3 步：在你的设备上做出推论

在此示例中，您将使用 Boto3 将编译作业的输出下载到边缘设备上。然后，您将导入 DLR，从数据集下载示例图像，调整此图像的大小以匹配模型的原始输入，然后进行预测。

1. 将编译后的模型从 Amazon S3 下载到设备，然后从压缩的 tarfile 中提取它。

   ```
   # Download compiled model locally to edge device
   object_path = f'output/{model_name}-{target_device}.tar.gz'
   neo_compiled_model = f'compiled-{model_name}.tar.gz'
   s3_client.download_file(bucket_name, object_path, neo_compiled_model)

   # Extract model from .tar.gz so DLR can use it
   !mkdir ./dlr_model # make a directory to store your model (optional)
   !tar -xzvf ./compiled-detect.tar.gz --directory ./dlr_model
   ```

2. 导入 DLR 和已初始化**`DLRModel`**对象。

   ```
   import dlr

   device = 'cpu'
   model = dlr.DLRModel('./dlr_model', device)
   ```

3. 下载图像进行推理并根据模型的训练方式对其进行格式化.

   对于`coco_ssd_mobilenet`例如，你可以从COCO 数据集然后将形象改造为`300x300`：

   ```
   from PIL import Image

   # Download an image for model to make a prediction
   input_image_filename = './input_image.jpg'
   !curl https://farm9.staticflickr.com/8325/8077197378_79efb4805e_z.jpg --output
    {input_image_filename}

   # Format image so model can make predictions
   resized_image = image.resize((300, 300))

   # Model is quantized, so convert the image to uint8
   x = np.array(resized_image).astype('uint8')
   ```

4. 使用 DLR 进行推断.

最后，您可以使用 DLR 对刚下载的图像进行预测：

```
out = model.run(x)
```

有关使用 DLR 从边缘设备上的新编译模型推断的更多示例，请参阅neo-ai-DLR Github 存储库.

错误排查

此部分包含的信息介绍如何了解和预防常见错误，这些错误所生成的错误消息，以及如何解决这些错误的指南。继续之前，请问自己以下问题：

你在部署模型之前遇到错误吗？ 如果是的话，请参阅排查 Neo 编译错误.

编译模型后你遇到错误吗？ 如果是的话，请参阅排查 Neo 推理错误.

你在尝试为 Amberlla 设备编译模型时遇到错误吗？ 如果是的话，请参阅解决 Amparlla 错误 (p. 2385).

错误分类类型

此列表对您可能从 Neo 收到的用户错误 进行了分类。这包括各个所支持框架的访问和权限错误以及加载错误。所有其他错误为系统错误。

客户端权限错误

Neo 直接传递来自相关服务的这些错误。

- 调用 sts:AssumeRole 时拒绝访问
- 任何 400调用 Amazon S3 下载或上传客户端模型时错误
- PassRole 错误

加载错误

假设 Neo 编译器成功从 Amazon S3 加载 .tar.gz，请检查 tarball 是否包含编译所需的文件。检查标准特定于框架：

- TensorFlow：仅需要 Protobuf 文件（ *.pb 或 *.pbtxt ）。对于保存的模型，需要一个变量文件夹。
- PyTorch：只需要一个 Pytorch 文件 (*.pth)。
- MXNET：只需要一个符号文件 (*.json) 和一个参数文件 (*.params)。
- XGBoost：仅需要一个 XGBoost 模型文件 (*.model)。输入模型有大小限制。

编译错误

假设 Neo 编译器成功从 Amazon S3 加载 .tar.gz，并且 tarball 包含编译所需的文件。检查标准是：

- OperatorNotImplemented：运算符尚未实施。
- OperatorAttributeNotImplemented：指定运算符中的属性尚未实施。
- OperatorAttributeRequired：对于内部符号图，某个属性是必需的，但未在用户输入模型图中列出。
- OperatorAttributeValueNotValid：特定运算符中属性的值无效。

主题

- 排查 Neo 编译错误 (p. 2383)
- 排查 Neo 推理错误 (p. 2384)

- 解决 Amparlla 错误 (p. 2385)

排查 Neo 编译错误

此部分包含的信息介绍如何了解和预防常见编译错误，这些错误所生成的错误消息，以及如何解决这些错误的指南。

主题

- 如何使用此页 (p. 2383)
- 框架相关错误 (p. 2383)
- 基础设施相关错误 (p. 2384)
- 检查你的编译日志 (p. 2384)

如何使用此页

尝试按以下顺序浏览这些部分来解决您的错误：

1. 检查编译作业的输入是否满足输入要求。请参阅 SageMaker Neo 需要什么输入数据形状？ (p. 2327)。
2. 检查常见框架特定的错误.
3. 检查你的错误是否是基础设施错.
4. 检查您的编译日志.

框架相关错误

TensorFlow

错误	解决方案
`InputConfiguration: Exactly one .pb file is allowed for TensorFlow models.`	确保只提供一个 .pb 或一个 .pbtxt 文件。
`InputConfiguration: Exactly one .pb or .pbtxt file is allowed for TensorFlow models.`	确保只提供一个 .pb 或一个 .pbtxt 文件。
`ClientError: InputConfiguration: TVM cannot convert <model zoo> model. Please make sure the framework you selected is correct. The following operators are not implemented: {<operator name>}`	检查您选择的操作员是否受支持。请参阅SageMaker Neo 支持的框架和操作.

Keras

错误	解决方案
`InputConfiguration: No h5 file provided in <model path>`	检查您的 H5 文件位于您指定的 Amazon S3 URI 中。 或者 检查h5 文件格式正确.
`InputConfiguration: Multiple h5 files provided, <model path>, when only one is allowed`	检查你只提供了一个h5文件。

错误	解决方案
`ClientError: InputConfiguration: Unable to load provided Keras model. Error: 'sample_weight_mode'`	检查您指定的 Keras 版本是否受支持。请参阅，支持的框架云实例和Edge 设备.
`ClientError: InputConfiguration: Input input has wrong shape in Input Shape dictionary. Input shapes should be provided in NCHW format.`	检查您的模型输入是否遵循 NCHW 格式。请参阅输入数据形状有什么作用 SageMaker Neo 需要吗？

MXNet

错误	解决方案
`ClientError: InputConfiguration: Only one parameter file is allowed for MXNet model. Please make sure the framework you select is correct.`	SageMaker Neo 将选择给出的第一个参数文件进行编译。

基础设施相关错误

错误	解决方案
`ClientError: InputConfiguration: S3 object does not exist. Bucket: <bucket>, Key: <bucket key>`	检查您提供的 Amazon S3 URI。
`ClientError: InputConfiguration: Bucket <bucket name> is in region <region name> which is different from Amazon Sagemaker service region <service region>`	创建与服务位于同一区域的 Amazon S3 存储桶。
`ClientError: InputConfiguration: Unable to untar input model. Please confirm the model is a tar.gz file`	检查您在 Amazon S3 中的模型是否已压缩成`tar.gz`文件。

检查你的编译日志

1. 导航到亚马逊 CloudWatch 在https://console.aws.amazon.com/cloudwatch/.
2. 从区域右上角的下拉列表。
3. 在 Amazon CloudWatch 的导航窗格中，选择日志. Select日志组.
4. 搜索名为的日志组`/aws/sagemaker/CompilationJobs`. 选择 日志组。
5. 搜索以编译作业名称命名的 Logstream。选择日志流。

排查 Neo 推理错误

本节包含有关如何防止和解决部署和/或调用终端节点时可能遇到的一些常见错误的信息。本节适用于PyTorch 1.4.0 或更高版本和MXNet v1.7.0 或更高版本.

- 确保在中完成对有效输入数据的第一个推理（预热推理）`model_fn()`，如果你定义了`model_fn`在推理脚本中，否则在终端上可能会看到以下错误消息：`predict API`被称作：

```
An error occurred (ModelError) when calling the InvokeEndpoint operation: Received server
 error (0) from <users-sagemaker-endpoint> with message "Your invocation timed out
 while waiting for a response from container model. Review the latency metrics for each
 container in Amazon CloudWatch, resolve the issue, and try again."
```

- 确保设置了下表中的环境变量。如果未设置它们，可能会显示以下错误消息：

  在终端上：

```
An error occurred (ModelError) when calling the InvokeEndpoint operation: Received
 server error (503) from <users-sagemaker-endpoint> with message "{ "code": 503, "type":
 "InternalServerException", "message": "Prediction failed" } ".
```

  在 CloudWatch 中：

```
W-9001-model-stdout com.amazonaws.ml.mms.wlm.WorkerLifeCycle - AttributeError: 'NoneType'
 object has no attribute 'transform'
```

密钥	值
SAGEMAKER_计划	inference.py
SAGEMAKER_SUBMIT_目录	/opt/ml/模型/代码
SAGEMAKER_容器_LOG_LEVEL	20
SAGEMAKER_区域	<your region>

- 确保`MMS_DEFAULT_RESPONSE_TIMEOUT`创建亚马逊时，环境变量设置为 500 或更高的值 SageMaker 模型；否则，可能会在终端上看到以下错误消息：

```
An error occurred (ModelError) when calling the InvokeEndpoint operation: Received server
 error (0) from <users-sagemaker-endpoint> with message "Your invocation timed out
 while waiting for a response from container model. Review the latency metrics for each
 container in Amazon CloudWatch, resolve the issue, and try again."
```

解决 Amparlla 错误

SageMaker Neo 要求将模型打包在压缩的 TAR 文件中（`*.tar.gz`）。Amberlla 设备需要在压缩的 TAR 文件中包含其他文件，然后才能将其发送进行编译。如果要为 Amberlla 目标编译模型，请将以下文件包含在压缩的 TAR 文件中 SageMaker Neo：

- 使用支持的框架的训练模型 SageMaker 新
- JSON 配置文件
- 校准图像

举例来说，您的 TAR 文件的内容看起来应如下示例所示：

```
###amba_config.json
###calib_data
|    ### data1
|    ### data2
|    ### .
|    ### .
```

```
|    ### .
|    ### data500
###mobilenet_v1_1.0_0224_frozen.pb
```

该目录的配置如下：

- `amba_config.json`：配置文件
- `calib_data`：包含校准图像的文件夹
- `mobilenet_v1_1.0_0224_frozen.pb`： TensorFlow 保存为冻结图表的模型

有关支持的框架的信息 SageMaker Neo，请参阅支持的框架 (p. 2365).

设置配置文件

配置文件提供了 Amberlla 工具链编译模型所需的信息。配置文件必须保存为 JSON 文件，文件的名称必须以`*config.json`. 下图显示配置文件的内容。

密钥	描述	示例
inputs	字典将输入图层映射到属性。	`{inputs:{"data": {...},"data1":{...}}}`
“数据”	输入图层名称。注意：“data” 是您可以用来标注输入图层的名称示例。	“数据”
形状	描述模型输入的形状。这遵循的是同样的约定 SageMaker Neo 使用。	“形状”：“1,3,224,224”
文件路径	包含校准图像的目录的相对路径。这些文件可以是二进制文件或图像文件，如 JPG 或 PNG。	“文件路径”：“calib_data/”
颜色格式	模型期望的颜色格式。这将在将图像转换为二进制文件时使用。支持的值：[RGB，BGR]。默认为 RGB。	“颜色格式”: “RGB”
意思	要从输入中减去的平均值。可以是单个值或值列表。当以列表形式给出均值时，条目数必须与输入的通道维度匹配。	“意思” :128.0
小数位数	用于标准化输入的缩放值。可以是单个值或值列表。当将比例作为列表提供时，条目的数量必须与输入的频道维度匹配。	“规模”：255.0

下面是示例配置文件：

```
{
    "inputs": {
        "data": {
                "shape": "1, 3, 224, 224",
```

```
                "filepath": "calib_data/",
                "colorformat": "RGB",
                "mean":[128,128,128],
                "scale":[128.0,128.0,128.0]
        }
    }
}
```

校准映像

通过提供校准图像来量化训练过的模型。量化模型可提高 CvFlow 引擎在安巴雷拉芯片系统 (SoC) 上的性能。Amberlla 工具链使用校准图像来确定如何量化模型中的每个图层以实现最佳性能和准确性。每个图层都按照 INT8 或 INT16 格式独立量化。量化后，最终模型混合了 INT8 和 INT16 层。

你应该使用多少张图片？

建议您包括代表模型应处理的场景类型的 100—200 张图像。模型编译时间线性增加至输入文件中校准图像的数量。

推荐的图片格式是什么？

校准图像可以是原始二进制格式，也可以是 JPG 和 PNG 等图像格式。

校准文件夹可以包含图像和二进制文件的混合物。如果校准文件夹同时包含图像和二进制文件，则工具链首先将图像转换为二进制文件。转换完成后，它将使用新生成的二进制文件以及最初位于该文件夹中的二进制文件。

我可以先将图像转换成二进制格式吗？

是。您可以使用开源软件包将图像转换为二进制格式，例如OpenCV要么PIL. 裁剪图像并调整其大小，以便它们满足训练模型的输入层。

平均值和规模

您可以在 Amberalla 工具链中指定均值和缩放预处理选项。这些操作嵌入到网络中，并在推断每个输入时应用。如果指定平均值或比例，请勿提供处理过的数据。更具体地说，不要提供您从均值中减去或已应用缩放比例的数据。

使用 Amazon SageMaker Elastic Inference (EI)

此功能在中国区域中不可用。

使用 Amazon Elastic Inference (EI)，您可以对部署作为部署的深度学习模型，提升吞吐量并缩短获取实时推理的延迟。Amazon SageMaker 托管模型，但仅需要将 GPU 实例用于终端节点的一小部分成本。通过 EI，您可以付出使用完整 GPU 实例的一小部分成本，将推理加速添加到托管的终端节点。除了 CPU 实例类型之外，将可用大小之一的 EI 加速器添加到可部署模型中，然后将该模型作为生产变体添加到您用于部署托管终端节点的终端节点配置中。您也可以将 EI 加速程序添加到 SageMaker。笔记本实例这样您可以在生成模型时测试和评估推理性能。

启用 EI 的 TensorFlow、Apache MXNet 和 PyTorch 版本中支持 Elastic Inference。要使用任何其他深度学习框架，请使用 ONNX 导出您的模型，然后将模型导入到 MXNet。然后，您可以将带有 EI 的模型用作 MXNet 模型。有关将 ONNX 模型导入 MXNet 中的信息，请参阅将 ONNX 模型导入 MXNet 中。

主题

- EI 的工作原理 (p. 2388)
- 选择 EI 加速器类型 (p. 2388)
- 在 SageMaker 笔记本实例中使用 EI (p. 2389)
- 在托管终端节点上使用 EI (p. 2389)
- 支持 EI 的框架 (p. 2389)
- 将 EI 与 SageMaker 内置算法结合使用 (p. 2389)
- EI 示例笔记本 (p. 2389)
- 设置以使用 EI。 (p. 2390)
- 将 EI 附加到笔记本实例 (p. 2393)
- 将 EI 在 Amazon SageMaker 托管终端节点上 (p. 2395)

EI 的工作原理

Amazon Elastic Inference 加速器是网络附加设备，该设备与终端节点中的 SageMaker 实例结合使用来加速推理调用。Elastic Inference 允许您将部分 GPU 附加到任何 SageMaker 实例，从而加速推理。您可以选择客户端实例来运行您的应用程序，并附加一个 Elastic Inference 加速器来使用适当数量的 GPU 加速，从而满足推理需求。在未充分利用 GPU 实例进行推理时，Elastic Inference 将帮助您降低成本。我们建议您使用不同的 CPU 实例和加速器大小来对模型试用 Elastic Inference。

EI 加速器有以下类型可用。您可以使用任意 EI 加速器类型配置终端节点或笔记本实例。

在该表中，列出了单精度浮点 (F32) 和半精度浮点 (F16) 操作的吞吐量，以万亿次浮点运算 (TFLOPS) 为单位。此外还以 GB 为单位列出了内存。

加速器类型	F32 吞吐量 (TFLOPS)	F16 吞吐量 (TFLOPS)	内存 (GB)
ml.eia2.medium	1	8	2
ml.eia2.large	2	16	4
ml.eia2.xlarge	4	32	8
ml.eia1.medium	1	8	1
ml.eia1.large	2	16	2
ml.eia1.xlarge	4	32	4

选择 EI 加速器类型

在为托管模型选择加速器类型时，请考虑以下因素：

- 模型、输入张量和批量大小影响您需要的加速器内存数量。在开始时，使用提供的内存至少与您训练模型的文件大小相当的加速器类型。一个因素是模型在运行时使用的内存可能比文件大小大得多。
- 对 CPU 计算资源、主系统内存以及基于 GPU 的加速和加速器内存的需求因不同类型的深度学习模型而异。应用程序的延迟和吞吐量需求也会确定您需要的计算和加速的数量。全面测试不同实例类型和 EI 加速器大小的配置，确保选择最适合您应用程序性能需求的配置。

有关如何选择 EI 加速器的更多信息，请参阅：

- Amazon Elastic Inference 概览
- 为模型选择实例和加速器类型

- 利用 TensorFlow 优化 Amazon Elastic Inference 成本

在 SageMaker 笔记本实例中使用 EI

通常情况下，您首先在 SageMaker 笔记本中生成和测试机器学习模型，然后部署模型用于生产。在创建笔记本实例时，您可以将 EI 附加到笔记本实例。您可以通过使用中的 TensorFlow、MXNet 和 PyTorch 评估程序及模型支持的本地模式，设置本地托管在笔记本实例上的终端节点。Amazon SageMaker Python 开发工具包来测试推理表现。笔记本实例上目前不支持启用 Elastic Inference 的 PyTorch。有关如何将 EI 连接到笔记本实例并设置本地终端节点用于推理的说明，请参阅将 EI 附加到笔记本实例 (p. 2393)。针对 TensorFlow 和 Apache MXNet 的启用了 Elastic Inference 的版本，还有启用 Elastic Inference 的 SageMaker 笔记本 Jupyter 内核。有关使用 SageMaker 笔记本实例的信息，请参阅使用 Amazon SageMaker 笔记本实例

在托管终端节点上使用 EI

在准备好部署模型用于生产以提供推理时，您将创建 SageMaker 托管终端节点。您可以将 EI 附加到其中托管了终端节点的实例，用于在提供推理时提升其性能。有关如何将 EI 附加到托管终端节点实例的说明，请参阅将 EI 在 Amazon SageMaker 托管终端节点上 (p. 2395)。

支持 EI 的框架

Amazon Elastic Inference 旨在与AmazonTensorFlow、Apache MXNet 或 PyTorch 机器学习框架的增强版本。这些框架的增强版本在您使用 Amazon SageMaker Python 开发工具包时自动构建到容器中，或者您可以将其作为二进制文件下载并导入到自己的 Docker 容器中。

您可以从公开下载启用了 EI 的 TensorFlow 二进制文件amazonei-tensorflow将 Amazon S3 存储桶转到 TensorFlow 服务容器。有关构建容器（使用启用了 EI 的 TensorFlow 版本）的更多信息，请参阅 SageMaker 中具有 TensorFlow 的 Amazon Elastic Inference。

您可以向公众下载启用了 EI 的 MXNet 二进制文件Amazonei-apachemxnet将 Amazon S3 存储桶转到 MxNet 提供服务的容器。有关构建容器（使用启用了 EI 的 MXNet 版本）的更多信息，请参阅 SageMaker 中具有 MXNet 的 Amazon Elastic Inference。

您可以从公开下载启用了 EI 的 PyTorch 二进制文件Amazon pytorch将 Amazon S3 存储桶转到 PyTorch 提供服务的容器。有关构建容器（使用启用了 EI 的 PyTorch 版本）的更多信息，请参阅 SageMaker 中具有 PyTorch 的 Amazon Elastic Inference。

要在托管终端节点中使用 Elastic Inference，您可以根据需要选择以下任意框架。

- SageMaker Python 开发工具包 - 部署 TensorFlow 模型
- SageMaker Python 开发工具包 - 部署 MXNet 模型
- SageMaker Python 开发工具包 - 部署 PyTorch 模型

如果您需要创建一个自定义容器来部署复杂的模型，并且此模型需要扩展到 SageMaker 预构建容器不支持的框架，请使用低级别Amazon适用于 Python 的 SDK (Boo 3).

将 EI 与 SageMaker 内置算法结合使用

目前，图像分类算法 (p. 1620)和对象检测算法 (p. 1691)内置算法支持 EI。有关将图像分类算法与 EI 结合使用的示例，请参阅端到端多类别图像分类示例。

EI 示例笔记本

以下示例笔记本提供了在 SageMaker 中使用 EI 的示例：

- 在 Amazon SageMaker 上将 Amazon Elastic Inference 与 MXNet 结合使用
- 在 Amazon SageMaker 笔记本实例上将 Amazon Elastic Inference 与 MXNet 结合使用
- 在 SageMaker 上将 Amazon Elastic Inference 与 NEO 编译 TensorFlow 模型结合使用
- 在 SageMaker 上将 Amazon Elastic Inference 与预训练 TensorFlow 服务模型结合使用

设置以使用 EI。

如果您遇到以下情况之一，则使用本主题中的说明：

- 您希望使用自定义角色或权限策略。
- 您希望为托管模型或笔记本实例使用 VPC。

> **Note**
>
> 如果您已有一个执行角色，请参阅`AmazonSageMakerFullAccess`附加了托管策略（适用于您在控制台中创建笔记本实例、训练作业或模型时创建的任何 IAM 角色，并且您未连接到 VPC 中的 EI 模型或笔记本实例，您无需进行这些更改中的任何更改即可在 Amazon SageMaker 中使用 EI。

主题

- 设置所需权限 (p. 2390)
- 使用自定义 VPC 来连接 EI (p. 2392)

设置所需权限

要在 SageMaker 中使用 EI，您用于打开笔记本实例的角色或者创建可部署模型的角色必须附加了具有所需权限的策略。您可以将 `AmazonSageMakerFullAccess` 托管策略（包含所需权限）附加到角色，或者添加具有所需权限的自定义策略。有关创建 IAM 角色的信息，请参阅为创建角色Amazon服务（控制台）中的Amazon Identity and Access Management用户指南. 有关将策略附加到角色的信息，请参阅添加和删除 IAM 策略。

在 IAM 策略中添加这些专门用于连接 EI 的权限。

```
{
    "Effect": "Allow",
    "Action": [
        "elastic-inference:Connect",
        "ec2:DescribeVpcEndpoints"
    ],
    "Resource": "*"
}
```

以下 IAM 策略是在 SageMaker 中使用 EI 所需权限的完整列表。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elastic-inference:Connect",
                "ec2:DescribeVpcEndpoints"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability",
                "cloudwatch:PutMetricData",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DeleteAlarms",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DeleteScheduledAction",
                "application-autoscaling:DeregisterScalableTarget",
                "application-autoscaling:DescribeScalableTargets",
                "application-autoscaling:DescribeScalingActivities",
                "application-autoscaling:DescribeScalingPolicies",
                "application-autoscaling:DescribeScheduledActions",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:PutScheduledAction",
                "application-autoscaling:RegisterScalableTarget",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::*SageMaker*",
                "arn:aws:s3:::*Sagemaker*",
                "arn:aws:s3:::*sagemaker*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:GetBucketLocation",
                "s3:ListBucket",
```

```
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "s3:ExistingObjectTag/SageMaker": "true"
                }
            }
        },
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/aws-service-role/sagemaker.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "sagemaker.application-autoscaling.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "sagemaker.amazonaws.com"
                }
            }
        }
    ]
}
```

使用自定义 VPC 来连接 EI

要在 VPC 中将 EI 与 SageMaker 结合使用，您需要创建和配置两个安全组，并设置 PrivateLink VPC 接口终端节点。EI 使用 VPC 接口终端节点来与 VPC 中的 SageMaker 终端节点通信。您创建的安全组用于连接到 VPC 接口终端节点。

设置安全组以连接到 EI

要在 VPC 中使用 EI，您需要创建两个安全组：

- 一个安全组用于控制对您将为 EI 设置的 VPC 接口终端节点的访问。
- 一个安全组允许 SageMaker 调用第一个安全组。

配置两个安全组

1. 创建没有出站连接的安全组。您将此项附加到在下一部分中创建的 VPC 终端节点接口。
2. 创建没有入站连接的第二个安全组，不过具有对第一个安全组的出站连接。
3. 编辑第一个安全组以仅允许对第二个安全组的入站连接和所有出站连接。

有关 VPC 安全组的更多信息，请参阅您的 VPC 的安全组中的Amazon Virtual Private Cloud 用户指南.

设置 VPC 接口终端节点以连接到 EI

要在自定义 VPC 中将 EI 与 SageMaker 结合使用，您需要为 EI 服务设置 VPC 接口终端节点 (PrivateLink)。

- 为 EI 设置 VPC 接口终端节点 (PrivateLink)。按照创建接口终端节点中的说明操作。在服务列表中，选择 com.amazonaws.<region>.elastic-inference.runtime。对于 Security group (安全组)，确保您为终端节点选择在上一部分中创建的第一个安全组。
- 设置接口终端节点时，选择 EI 可用的所有可用区。如果您未设置至少两个可用区，EI 将失败。有关 VPC 子网的信息，请参阅 VPC 和子网。

将 EI 附加到笔记本实例

要使用 EI 测试和评估推理性能，您可以在创建或更新笔记本实例时将 EI 附加到笔记本实例。您可在本地模式中使用 EI，用于在托管于笔记本实例上的终端节点中托管模型。您应测试笔记本实例和 EI 加速器的各种大小，用于评估最适合您的使用案例的配置。

设置以使用 EI。

要在笔记本实例中本地使用 EI，请创建具有 EI 实例的笔记本实例。

使用 EI 实例创建笔记本实例

1. 打开位于 https://console.aws.amazon.com/sagemaker/ 的 Amazon SageMaker 控制台。
2. 在导航窗格中，选择 Notebook instances (笔记本实例)。
3. 选择创建笔记本实例。
4. 对于 Notebook instance name (笔记本实例名称)，为笔记本实例提供唯一名称。
5. 对于 notebook instance type (笔记本实例类型)，选择 CPU 实例，例如 ml.t2.medium。
6. 对于 Elastic Inference (EI) (弹性推理 (EI))，请从列表中选择一个实例，例如 ml.eia2.medium。
7. 适用于IAM 角色中，请选择具有使用 SageMaker 和 EI 所需权限的 IAM 角色。
8. （可选）对于 VPC - Optional (VPC - 可选)，如果您希望笔记本实例使用 VPC，请从可用列表中选择一个。否则，将其保留为 No VPC (无 VPC)。如果您使用 VPC，请按照使用自定义 VPC 来连接 EI (p. 2392) 的说明操作。
9. （可选）对于 Lifecycle configuration - optional (生命周期配置 - 可选)，请将其保留作为 No configuration (无配置) 或者选择生命周期配置。有关更多信息，请参阅 使用生命周期配置脚本自定义笔记本实例 (p. 152)。
10. （可选）对于加密密钥-可选，可选）如果您希望 SageMaker 使用Amazon Key Management Service(Amazon KMS) 密钥要加密附加到笔记本实例的 ML 存储卷中的数据，请指定密钥。
11. （可选）对于 Volume Size In GB - optional (卷大小 (以 GB 为单位) - 可选)，请保留默认值 5。
12. （可选）对于 Tags (标签)，请将标签添加到笔记本实例。标签是您分配用于帮助管理笔记本实例的标注。标签包含您定义的一个键和一个值。
13. 选择 Create Notebook Instance (创建笔记本实例)。

在创建附加了 EI 的笔记本实例之后，您可以创建 Jupyter 笔记本并设置在笔记本实例本地托管的 EI 终端节点。

主题

- 在 SageMaker 中以本地模式使用 EI (p. 2394)

在 SageMaker 中以本地模式使用 EI

要在笔记本实例上托管的终端节点中本地使用 EI，请使用Amazon SageMaker Python 开发工具包TensorFlow、MXNet 或 PyTorch 评估程序或模型的版本。有关 SageMaker Python 开发工具包中支持的本地模式的更多信息，请参阅https://github.com/aws/sagemaker-python-sdk#sagemaker-python-sdk-overview.

主题

- 将 EI 以本地模式与 SageMaker TensorFlow 评估程序和模型结合使用 (p. 2394)
- 将 EI 在本地模式下与 SageMaker Apache MXNet 评估程序和模型结合使用 (p. 2394)
- 将 EI 与 SageMaker PyTorch 评估程序和模型结合使用 (p. 2394)

将 EI 以本地模式与 SageMaker TensorFlow 评估程序和模型结合使用

要在本地模式下将 EI 与 TensorFlow 结合使用，在您调用评估程序或模型对象的 `deploy` 方法时，请为 `instance_type` 指定 `local`，为 `accelerator_type` 指定 `local_sagemaker_notebook`。有关 的更多信息Amazon SageMaker Python 开发工具包TensorFlow 评估程序和模型，请参阅https://sagemaker.readthedocs.io/en/stable/frameworks/tensorflow/index.html.

以下代码演示如何将本地模式用于评估程序对象。要调用 `deploy` 方法，您以前必须已经执行以下操作之一：

- 通过调用评估程序的 `fit` 方案训练了模型。
- 初始化模型对象时传递了模型构件。

```
# Deploys the model to a local endpoint
tf_predictor = tf_model.deploy(initial_instance_count=1,
                               instance_type='local',
                               accelerator_type='local_sagemaker_notebook')
```

将 EI 在本地模式下与 SageMaker Apache MXNet 评估程序和模型结合使用

要在本地模式下将 EI 与 MXNet 结合使用，在您调用评估程序或模型对象的 `deploy` 方法时，请为 `instance_type` 指定 `local`，为 `accelerator_type` 指定 `local_sagemaker_notebook`。有关 的更多信息Amazon SageMaker Python 开发工具包MXNet 评估程序和模型，请参阅https://sagemaker.readthedocs.io/en/stable/frameworks/mxnet/index.html.

以下代码演示如何将本地模式用于评估程序对象。您必须以前已调用过评估程序的 `fit` 方法来训练模型。

```
# Deploys the model to a local endpoint
mxnet_predictor = mxnet_estimator.deploy(initial_instance_count=1,
                                         instance_type='local',
                                         accelerator_type='local_sagemaker_notebook')
```

有关在本地模式下将 EI 与 MXNet 结合使用的完整示例，请参阅位于https://sagemaker-examples.readthedocs.io/en/latest/sagemaker-python-sdk/mxnet_mnist/mxnet_mnist_elastic_inference_local.html.

将 EI 与 SageMaker PyTorch 评估程序和模型结合使用

要在本地模式下将 EI 与 PyTorch 结合使用，在您调用评估程序或模型对象的 `deploy` 方法时，请为 `instance_type` 指定 `local`，并为 `accelerator_type` 指定 `local_sagemaker_notebook`。有关 的更多信息Amazon SageMaker Python 开发工具包PyTorch 估计器和模型，请参阅SageMaker PyTorch 评估程序和模型.

以下代码演示如何将本地模式用于评估程序对象。您必须以前已调用过评估程序的 `fit` 方法来训练模型。

```
# Deploys the model to a local endpoint
pytorch_predictor = pytorch_estimator.deploy(initial_instance_count=1,
                                             instance_type='local',
                                             accelerator_type='local_sagemaker_notebook')
```

将 EI 在 Amazon SageMaker 托管终端节点上

要在 Amazon SageMaker 中将 Elastic Inference (EI) 用于托管终端节点进行实时推理，请在您创建托管在终端节点上的可部署模型时，请指定 EI 加速器。您可以通过下列方式之一来执行该操作：

- 使用Amazon SageMaker Python 开发工具包TensorFlow、MXNet 或 PyTorch 的版本，以及适用于 TensorFlow、MXNet 和 PyTorch 的 SageMaker 预构建容器
- 构建您自己的容器，然后使用低级别 SageMaker API (Boo 3)。您需要从提供的 Amazon S3 位置，将 TensorFlow、MXNet 或 PyTorch 的启用 EI 的版本导入容器中，并使用这些版本之一来编写训练脚本。
- 使用 图像分类算法 (p. 1620) 或 对象检测算法 (p. 1691) 内置算法，并使用 Amazon SDK for Python (Boto3) 来运行训练作业和创建您的可部署模型及托管终端节点。

主题

- 将 EI 与 SageMaker TensorFlow 容器结合使用 (p. 2395)
- 将 EI 与 SageMaker MXNet 容器结合使用 (p. 2396)
- 将 EI 与 SageMaker PyTorch 容器结合使用 (p. 2396)
- 将 EI 用于自己的容器 (p. 2397)

将 EI 与 SageMaker TensorFlow 容器结合使用

要在 SageMaker 中将 TensorFlow 与 EI 结合使用，你需要调用`deploy`任何一种方法估计程序要么模型对象。然后，您可以使用 accelerator_type 输入参数指定加速器类型。有关在 SageMaker Python 开发工具包中使用 TensorFlow 的信息，请参阅：https://sagemaker.readthedocs.io/en/stable/frameworks/tensorflow/index.html.

SageMaker 提供了默认模型训练和推理代码以便您使用。对于自定义文件格式，您可能需要实现自定义模型训练和推理代码。

使用评估程序对象

要将评估程序对象与 EI 结合使用，请在使用部署方法时包含 `accelerator_type` 输入参数。评估程序返回我们调用其部署方法的预测器对象，如示例代码中所示。

```
# Deploy an estimator using EI (using the accelerator_type input argument)
predictor = estimator.deploy(initial_instance_count=1,
                             instance_type='ml.m4.xlarge',
                             accelerator_type='ml.eia2.medium')
```

使用模型对象

要将模型对象与 EI 结合使用，请在使用部署方法时包含 `accelerator_type` 输入参数。评估程序返回我们调用其部署方法的预测器对象，如示例代码中所示。

```
# Deploy a model using EI (using the accelerator_type input argument)
predictor = model.deploy(initial_instance_count=1,
```

```
                            instance_type='ml.m4.xlarge',
                            accelerator_type='ml.eia2.medium')
```

将 EI 与 SageMaker MXNet 容器结合使用

要在 SageMaker 中将 MxNet 与 EI 结合使用，你需要调用deploy任何一种方法估计程序要么模型对象。然后，您可以使用 accelerator_type 输入参数指定加速器类型。有关在中使用 MXNet 的信息Amazon SageMaker Python 开发工具包，请参阅https://sagemaker.readthedocs.io/en/stable/frameworks/mxnet/index.html

为方便起见，SageMaker 提供了默认模型训练和推理代码。对于自定义文件格式，您可能需要编写自定义模型训练和推理代码。

使用评估程序对象

要将评估程序对象与 EI 结合使用，请在使用部署方法时包含 accelerator_type 输入参数。评估程序返回我们调用其部署方法的预测器对象，如示例代码中所示。

```
# Deploy an estimator using EI (using the accelerator_type input argument)
predictor = estimator.deploy(initial_instance_count=1,
                            instance_type='ml.m4.xlarge',
                            accelerator_type='ml.eia2.medium')
```

使用模型对象

要将模型对象与 EI 结合使用，请在使用部署方法时包含 accelerator_type 输入参数。评估程序返回我们调用其部署方法的预测器对象，如示例代码中所示。

```
# Deploy a model using EI (using the accelerator_type input argument)
predictor = model.deploy(initial_instance_count=1,
                            instance_type='ml.m4.xlarge',
                            accelerator_type='ml.eia2.medium')
```

有关在 SageMaker 中将 EI 与 MXNet 结合使用的完整示例，请参阅位于https://github.com/awslabs/amazon-sagemaker-examples/blob/master/sagemaker-python-sdk/mxnet_mnist/mxnet_mnist_elastic_inference.ipynb

将 EI 与 SageMaker PyTorch 容器结合使用

要在 SageMaker 中将 PyTorch 与 EI 结合使用，你需要调用deploy任何一种方法估计程序要么模型对象。然后，您可以使用 accelerator_type 输入参数指定加速器类型。有关在中使用 PyTorch 的信息Amazon SageMaker Python 开发工具包，请参阅SageMaker PyTorch 评估程序和模型.

为方便起见，SageMaker 提供了默认模型训练和推理代码。对于自定义文件格式，您可能需要编写自定义模型训练和推理代码。

使用评估程序对象

要将评估程序对象与 EI 结合使用，请在使用部署方法时包含 accelerator_type 输入参数。评估程序返回我们调用其部署方法的预测器对象，如此示例代码中所示。

```
# Deploy an estimator using EI (using the accelerator_type input argument)
predictor = estimator.deploy(initial_instance_count=1,
                            instance_type='ml.m4.xlarge',
                            accelerator_type='ml.eia2.medium')
```

使用模型对象

要将模型对象与 EI 结合使用，请在使用部署方法时包含 `accelerator_type` 输入参数。此模型返回我们调用其部署方法的预测器对象，如示例代码中所示：

```
# Deploy a model using EI (using the accelerator_type input argument)
predictor = model.deploy(initial_instance_count=1,
                             instance_type='ml.m4.xlarge',
                             accelerator_type='ml.eia2.medium')
```

将 EI 用于自己的容器

要将 EI 用于您构建的自定义容器中的一个模型，请使用适用于 Python 的低级 Amazon 开发工具包 (Boto 3)。下载并导入 Amazon 启用了 EI 的 TensorFlow、Apache MXNet 或 PyTorch 机器学习框架版本，并使用这些框架编写训练脚本。

将 TensorFlow、MXNet 或 PyTorch 的 EI 版本导入您的 Docker 容器

要将 EI 用于您自己的容器，您需要将 Amazon EI TensorFlow Serving 库、Amazon EI Apache MXNet 库或启用 Elastic Inference 的 PyTorch 库导入到容器中。TensorFlow 和 MXNet 的启用 EI 的版本当前作为存储在 Amazon S3 位置中的二进制文件可用。您可以从 Amazon S3 存储桶中下载启用了 EI 的 TensorFlow 二进制文件，网址为：console e.aws.amazon.com/s3/buckets/amazonei-tensorflow. 有关构建使用启用了 EI 的 TensorFlow 版本的容器的信息，请参阅 https://github.com/aws/sagemaker-tensorflow-container#building-the-sagemaker-elastic-inference-tensorflow-serving-container。您可以从公有 Amazon S3 存储桶中下载启用了 EI 的 Apache MXNet 二进制文件，网址为：console e.aws.amazon.com/s3/buckets/Amazon apachemxnet. 有关构建使用启用了 EI 的 MXNet 版本的容器的信息，请参阅 https://github.com/aws/sagemaker-mxnet-container#building-the-sagemaker-elastic-inference-mxnet-container。您可以从公有 Amazon S3 存储桶中下载启用了 Elastic Inference 的 PyTorch 二进制文件，网址为：console.aws.amazon.com/s3/buckets/Amazon onei-pytorch. 有关构建容器（该容器使用启用了 Elastic Inference 的 PyTorch 版本）的信息，请参阅构建映像。

使用创建 EI 终端节点Amazon适用于 Python 的 SDK (Boo 3)

要使用适用于 Python 的 Amazon 开发工具包 (Boto 3) 创建终端节点，首先需要创建终端节点配置。终端节点配置指定您要托管在终端节点上的一个或多个模型（称为生产变体）。要将 EI 附加到托管在终端节点上的一个或多个生产变体，您可以指定 EI 实例类型之一作为该 `ProductionVariant` 的 `AcceleratorType` 字段。然后，创建终端节点时，传递该终端节点配置。

创建终端节点配置

要使用 EI，您需要在终端节点配置中指定加速器类型。

```
# Create Endpoint Configuration
from time import gmtime, strftime

endpoint_config_name = 'ImageClassificationEndpointConfig-' + strftime("%Y-%m-%d-%H-%M-%S",
 gmtime())
print(endpoint_config_name)
create_endpoint_config_response = sagemaker.create_endpoint_config(
    EndpointConfigName = endpoint_config_name,
    ProductionVariants=[{
        'InstanceType':'ml.m4.xlarge',
        'InitialInstanceCount':1,
        'ModelName':model_name,
        'VariantName':'AllTraffic',
        'AcceleratorType':'ml.eia2.medium'}])

print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])
```

创建终端节点

创建具有加速器类型的终端节点配置之后，您可以创建终端节点。

```
endpoint_name = 'ImageClassificationEndpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
endpoint_response = sagemaker.create_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=endpoint_config_name)
```

创建终端节点后，您可以使用 Boto3 运行时对象中的 `invoke_endpoint` 方法调用它，就像对待任何其他终端节点一样。

在 GPU 驱动程序升级过程中尽量减少中断的最佳

SageMaker 模型部署会随着时间的推移升级机器学习实例上的 GPU 驱动程序，以获得实时、Batch 和异步推理选项，以便客户可以从驱动程序提供商获得 下面你可以看到每个推理选项支持的 GPU 版本。不同的驱动程序版本可以改变模型与 GPU 的交互方式。以下是一些策略，可帮助您了解应用程序如何在不同驱动程序版本中工作。

当前版本和支持的实例系列

亚马逊 SageMaker 推理支持以下驱动程序和实例系列：

服务	GPU	驱动程序版本	实例类型
实时	NVIDIA	440.33.01	ml.p2.*、ml.p3.*、ml.g4dn.*
Batch	NVIDIA	470.57.02	ml.p2.*、ml.p3.*、ml.g4dn.*
异步推理	NVIDIA	440.33.01	ml.p2.*、ml.p3.*、ml.g4dn.*

使用 GPU 功能排除模型容器故障

如果您在运行 GPU 工作负载时遇到问题，请参阅以下指南：

GPU 卡检测失败或 NVIDIA 初始化错误

运行`nvidia-smi`Docker 容器中的（NVIDIA 系统管理界面）命令。如果 NVIDIA 系统管理界面检测到 GPU 检测错误或 NVIDIA 初始化错误，它将返回以下错误消息：

```
Failed to initialize NVML: Driver/library version mismatch
```

根据您的使用案例，请遵循以下最佳实践来解决失败或错误：

- 遵循中所述的最佳实践建议如果你自带 (BYO) 模型容器 (p. 2399)“。
- 遵循中所述的最佳实践建议如果你使用 CUDA 兼容层 (p. 2399)“。

请参阅NVIDIA 系统管理界面页请访问 NVIDIA 网站了解更多信息。

使用不匹配驱动程序版本的最佳实践

以下内容提供了有关如何更新 GPU 驱动程序的信息：

我的容器所依赖的驱动程序低于 ML GPU 实例上的版本

无需采取行动。NVIDIA 提供向后兼容性。

我的容器所依赖的驱动程序大于 ML GPU 实例的版本

如果是小版本差异，则不需要执行任何操作。NVIDIA 提供次要版本向前兼容性。

如果是主要版本差异，则需要安装 CUDA 兼容性 Package。请参阅CUDA 兼容性 Package在 NVIDIA 文档中。

> **Important**
>
> CUDA 兼容性 Package 不向后兼容，因此如果实例上的驱动程序版本大于 CUDA 兼容性 Package 版本，则需要禁用它。

如果你自带 (BYO) 模型容器

确保镜像中没有绑定可能与主机上的 NVIDIA 驱动程序版本冲突的 NVIDIA 驱动程序包。

如果你使用 CUDA 兼容层

将正在进行的代码片段添加到ENTRYPOINT脚本如果你使用 CUDA 兼容性 (compat) 层。

截图的代码演示了如何根据实例上的版本动态切换 CUDA 兼容性 Package 的使用，以便在发生驱动程序升级时，不会中断。继续的代码段提供了一个示例，说明如何根据实例上的版本动态切换 CUDA 兼容性的使用。这应该包括在ENTRYPOINT脚本：

```
#!/bin/bash

verlte() {
    [ "$1" = "$2" ] $$ return 1 || [ "$2" = "`echo -e "$1\n$2" | sort -V | head -n1`" ]
}

if [ -f /usr/local/cuda/compat/libcuda.so.1 ]; then
    cat /usr/local/cuda/version.txt
    CUDA_COMPAT_MAX_DRIVER_VERSION=$(readlink /usr/local/cuda/compat/libcuda.so.1 |cut -
d'.' -f 3-)
    echo "CUDA compat package requires Nvidia driver #${CUDA_COMPAT_MAX_DRIVER_VERSION}"
    NVIDIA_DRIVER_VERSION=$ (nvidia-smi --query-gpu=driver_version --format=csv,noheader --
id=0)
    echo "Current installed Nvidia driver version is ${NVIDIA_DRIVER_VERSION}"
    if [ $(verlte $CUDA_COMPAT_MAX_DRIVER_VERSION $NVIDIA_DRIVER_VERSION) ]; then
        echo "Setup CUDA compatibility libs path to LD_LIBRARY_PATH"
        export LD_LIBRARY_PATH=/usr/local/cuda/compat:$LD_LIBRARY_PATH
        echo $LD_LIBRARY_PATH
    else
        echo "Skip CUDA compat libs setup as newer Nvidia driver is installed"
    fi
else
    echo "Skip CUDA compat libs setup as package not found"
fi
```

将 Docker 容器用于 SageMaker

亚马逊 SageMaker 广泛使用Docker 容器用于构建和运行时任务。 SageMaker 为其内置算法提供了预构建的 Docker 映像，并且提供了支持的深度学习框架以用于训练和推理。使用容器，您可以在任意规模快速可靠地训练机器学习算法并部署模型。本节中的主题展示了如何为自己的使用案例部署这些容器。有关如何携带自己的容器以便与亚马逊一起使用的信息 SageMaker 工作室，请参阅自带 SageMaker 图像 (p. 101).

主题

- 使用 运行脚本、训练算法或部署模型的场景 SageMaker (p. 2400)
- Docker 容器基础知识 (p. 2400)
- 使用预构建 SageMaker Docker 映像 (p. 2401)
- 调整你自己的 Docker 容器以与 SageMaker 配合使用 (p. 2414)
- 使用自己的算法和模型创建容器 (p. 2428)
- 示例笔记本：使用您自己的算法或模型 (p. 2442)
- 排除 Docker 容器故障排除 (p. 2444)

使用 运行脚本、训练算法或部署模型的场景 SageMaker

亚马逊 SageMaker 在运行脚本、训练算法和部署模型时，始终使用 Docker 容器。但是，您与容器的参与程度取决于您的使用案例。

- 使用内置 SageMaker 算法或框架. 对于大多数使用案例，您可以使用内置算法和框架而不必担心容器。您可以从 SageMaker 控制台，Amazon Command Line Interface(Amazon CLI)、Python 笔记本或亚马逊 SageMaker Python 开发工具包通过在创建 Estastator 时指定算法或框架版本。可用的内置算法将会分项列出，并且在使用 Amazon SageMaker 内置算法或预训练模型 (p. 787)主题中进行了介绍。有关可用框架的更多信息，请参阅ML 框架和工具包 (p. 11). 有关如何使用在中运行的 Jupyter 笔记本训练和部署内置算法的示例，请参阅一个示例。 SageMaker 笔记本实例，请参阅开始使用 Amazon SageMaker (p. 30)主题。
- 使用预构建的 SageMaker 容器镜像. 或者，您可以使用 Docker 容器使用内置算法和框架。 SageMaker 为其内置算法提供了容器，并为一些最常见的机器学习框架提供了预构建的 Docker 映像，如 Apache MXNet、TensorFlow、PyTorch 和 Chainer。有关可用的完整列表 SageMaker 图片，请参阅可用的 Deep Learning Containers 映像. 它还支持机器学习库，如 scikit-learn 和 SparkML。如果您将亚马逊 SageMaker Python 开发工具包，您可以通过将完整的容器 URI 传递给各自的容器 URI 来部署容器 SageMaker 开发工具包`Estimator`类。有关 SageMaker 当前支持的深度学习框架的完整列表，请参阅用于深度学习的预构建 SageMaker Docker 镜像 (p. 2401). 有关 scikit-learn 和 SparkML 预构建容器映像的信息，请参阅预构建的 Amazon SageMaker Scikit-learn 和 Spark ML 的 Docker 映像 (p. 2402). 有关将框架与亚马逊 SageMaker Python 开发工具包，请参阅中的各自主题将 Machine Learning 框架、Python 和 R 与 Amazon SageMaker 结合使用 (p. 11).
- 扩展预构建的 SageMaker 容器镜像. 如果您想扩展预构建的 SageMaker 算法或模型 Docker 镜像，你可以修改 SageMaker 图片来满足你的需求。有关示例，请参阅 。扩展我们的 PyTorch 容器.
- 调整现有的容器镜像：如果您想调整预先存在的容器映像以便与 SageMaker 配合使用，则需要修改 Docker 容器以启用 SageMaker 培训或推理工具包。有关说明如何构建您自己的容器来训练和托管算法的示例，请参阅使用自己的 R 算法。

Docker 容器基础知识

Docker 是一个程序，该程序执行操作系统级虚拟化，以便安装、分发和管理软件。它将应用程序及其依赖项打包到虚拟容器中，后者提供了隔离、可移植性和安全性。通过使用 Docker，您可以更快地发布代码、实现

应用程序操作的标准化、无缝地迁移代码，并通过改进资源利用率来实现经济效益。有关 Docker 的更多一般信息，请参阅Docker 概述.

以下信息概述了将 Docker 容器与 Amazon SageMaker 结合使用的最重要方面。

SageMaker 函数

SageMaker 在后端使用 Docker 容器来管理训练和推理过程。 SageMaker 从这个过程中抽象出来，所以当使用估计器时，它会自动发生。尽管您不需要明确使用 Docker 容器 SageMaker 在大多数使用案例中，您可以使用 Docker 容器进行扩展和自定义。 SageMaker 功能。

中的容器 SageMaker 工作室

SageMaker Studio 从 Docker 容器中运行，并使用它来管理功能。因此，您必须按照中的步骤创建 Docker 容器自带 SageMaker 图像 (p. 101).

使用预构建 SageMaker Docker 映像

亚马逊 SageMaker 为其内置算法提供了容器，并为一些最常见的机器学习框架提供了预构建的 Docker 映像，如 Apache MXNet、TensorFlow、PyTorch 和 Chainer。它还支持机器学习库，如 scikit-learn 和 SparkML。

你可以使用你的 SageMaker 笔记本实例或 SageMaker 工作室。你也可以扩展预构建 SageMaker 图片以包含库和所需的功能。以下主题介绍有关可用映像以及如何使用它们的信息。

> Note
>
> 有关在 SageMaker 中开发强化学习 (RL) 解决方案的 Docker 映像的信息，请参阅SageMaker RL 容器.

主题

- 用于深度学习的预构建 SageMaker Docker 镜像 (p. 2401)
- 预构建的 Amazon SageMaker Scikit-learn 和 Spark ML 的 Docker 映像 (p. 2402)
- 训练深度图表网络 (p. 2403)
- 扩展预构建的容器 (p. 2405)

用于深度学习的预构建 SageMaker Docker 镜像

SageMaker 提供了预构建的 Docker 映像，包括深度学习框架库以及训练和推理所需的其他依赖项。有关可用的预构建 Docker 映像的完整列表，请参阅深度学习容器映像。

如果不使用Amazon SageMaker Python 开发工具包还有其评估程序之一来检索预构建映像，则必须自行检索它们。

使用 SageMaker Python 开发工具包

借助 SageMaker Python 开发工具包，您可以使用这些流行的深度学习框架训练和部署模型。有关安装和使用该开发工具包的说明，请参阅Amazon SageMaker Python 开发工具包. 下表列出了可用的框架以及有关如何将这些框架与SageMaker Python 开发工具包：

框架	说明
`TensorFlow`	将 TensorFlow 与 SageMaker Python 开发工具包结合使用

框架	说明
MXNet	将 MXNet 与 SageMaker Python 开发工具包结合使用
PyTorch	将 PyTorch 与 SageMaker Python 开发工具包结合使用
Chainer	将 Chainer 与 SageMaker Python 开发工具包结合使用
Hugging Face	将拥抱面对与 SageMaker Python 开发工具包结合使用

扩展预构建的 SageMaker Docker 映像

您可以自定义或扩展这些预构建的容器，以满足预构建的 SageMaker Docker 映像不支持的算法或模型的任何其他功能要求。有关示例，请参阅扩展 PyTorch 容器。

您还可以使用预构建的容器部署自定义模型或在 SageMaker 之外的框架中训练的模型。有关将训练的模型构件引入 SageMaker 并在终端节点中托管这些构件的过程概述，请参阅将自己的预训练的 MxNet 或 TensorFlow 模型带到 Amazon SageMaker.

预构建的 Amazon SageMaker Scikit-learn 和 Spark ML 的 Docker 映像

SageMaker 提供安装 scikit-learn 和 Spark ML 库的预构建的 Docker 映像。这些库还包括构建与兼容的 Docker 映像所需的依赖项 SageMaker 使用亚马逊 SageMaker Python 开发工具包. 通过使用该开发工具包，您可以使用 scikit-learn 执行机器学习任务，并使用 Spark ML 创建和优化机器学习管道。有关安装和使用该开发工具包的说明，请参阅 SageMaker Python 开发工具包。

使用 SageMaker Python 开发工具包

下表包含指向 GitHub 具有 scikit-learn 和 Spark ML 容器源代码的存储库。该表还包含说明如何将这些容器与 Python 开发工具包评估程序结合使用，以运行您自己的训练算法和托管您自己的模型的说明链接。

Library	预构建的 Docker 映像源代码	说明
scikit-learn	SageMaker Scikit-learn 容器	将 Scikit-learn 与 Amazon 结合使用 SageMaker Python 开发工具包
Spark ML	SageMaker Spark ML 服务容器	SparkML Python 开发工具包文档

手动指定预构建的映像

如果您没有使用 SageMaker Python 开发工具包及评估程序之一来管理容器，则必须手动检索相关的预构建容器。SageMaker 预构建的 Docker 映像存储在 Amazon Elastic Container Registry (Amazon ECR) 中。您可以使用全名注册表地址推送或提取它们。 SageMaker 将以下 Docker 映像 URL 模式用于 scikit-learn 和 Spark M：

- `<ACCOUNT_ID>.dkr.ecr.<REGION_NAME>.amazonaws.com/sagemaker-scikit-learn:<SCIKIT-LEARN_VERSION>-cpu-py<PYTHON_VERSION>`

  例如，`746614075791.dkr.ecr.us-west-1.amazonaws.com/sagemaker-scikit-learn:0.23-1-cpu-py3`
- `<ACCOUNT_ID>.dkr.ecr.<REGION_NAME>.amazonaws.com/sagemaker-sparkml-serving:<SPARK-ML_VERSION>`

例如，*341280168497*.dkr.ecr.*ca-central-1*.amazonaws.com/*sagemaker-sparkml-serving:2.4*

下表列出了账户 ID 和相应的支持值。Amazon区域名称。

ACCOUNT_ID	REGION_NAME
746614075791	us-west-1
246618743249	us-west-2
683313688378	us-east-1
257758044811	us-east-2
354813040037	ap-northeast-1
366743142698	ap-northeast-2
121021644041	ap-southeast-1
783357654285	ap-southeast-2
720646828776	ap-south-1
141502667606	eu-west-1
764974769150	eu-west-2
492215442770	eu-central-1
341280168497	ca-central-1
414596584902	us-gov-west-1

查找可用映像

使用以下命令找出哪些版本的图像可用。例如，使用以下内容来查找可用sagemaker-sparkml-serving映像ca-central-1区域:

```
aws \
    ecr describe-images \
    --region ca-central-1 \
    --registry-id 341280168497 \
    --repository-name sagemaker-sparkml-serving
```

训练深度图表网络

在本概述中，您将了解如何使用 Amazon Elastic Container Registry (Amazon ECR) 中的 DGL 容器之一来开始使用深度图表网络。您还可以查看深度图表网络的实例的链接。

什么是深度图表网络？

深度图表网络是指一种经过训练的用于解决图表问题的神经网络。深度图表网络使用底层深度学习框架，如 PyTorch 或 MXNet。在以下方面的 Amazon SageMaker 教程中，强调了图表网络在实际 AI 应用中的潜力。深度图表库(DGL)。图表数据集上的训练模型示例包括社交网络、知识库、生物学和化学。

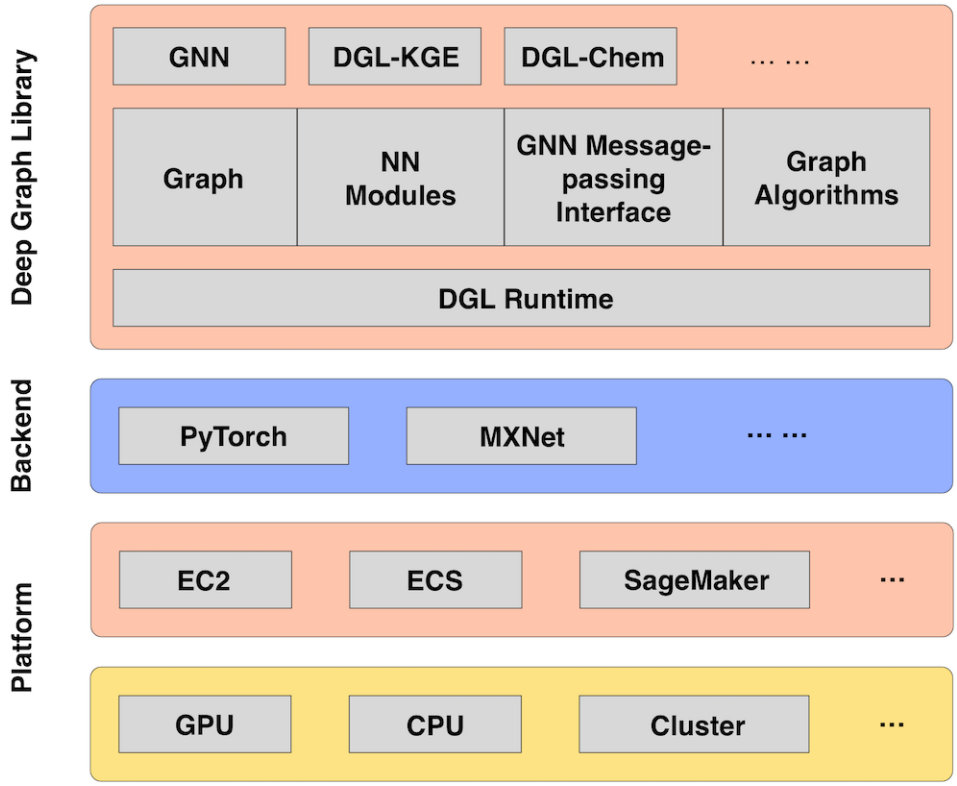


图 1. DGL 生态系统

使用预先配置了 DGL 的 Amazon SageMaker 深度学习容器提供了几个示例。如果您有要与 DGL 结合使用的特殊模块，则也可以构建您自己的容器。这些示例涉及异质图表，即具有多种类型的节点和边缘的图表，并利用了跨不同科学领域的各种应用，如生物信息学和社交网络分析。DGL 提供了一系列广泛的针对不同类型的模型的图表神经网络实现。部分亮点包括：

- 图表卷积网络 (GCN)
- 关系图表卷积网络 (R-GCN)
- 图表关注网络 (GAT)
- 图表的深度生成模型（DGMG）
- 联合树神经网络 (JTNN)

入门

DGL 可在 Amazon ECR 中用作深度学习容器。在 Amazon SageMaker 笔记本中编写评估程序函数时，可以选择深度学习容器。您还可以按照以下操作，使用 DGL 精心构建自己的自定义容器构建您自己的容器指南。 开始使用深度图表网络的最简单方法是使用 Amazon ECR 中的 DGL 容器之一。

> Note
>
> 后端框架支持仅限于 PyTorch 和 MXNet。

设置

如果您使用的是 Amazon SageMaker Studio，则需要先克隆示例存储库。如果您使用的是笔记本实例，则可以通过选择左侧工具栏底部的 SageMaker 图标来查找示例。

克隆 Amazon SageMaker 开发工具包和笔记本示例存储库

1. 从JupyterLab在 Amazon SageMaker 中查看，转到文件浏览器在左侧工具栏顶部。从文件浏览器面板，您可以通过查看新导航在面板顶部。
2. 选择图标在最右侧的是克隆 Git 存储库。
3. 添加存储库 URL：https://github.com/awslabs/amazon-sagemaker-examples.git
4. 浏览新添加的文件夹及其内容。DGL 示例将存储在sagemaker-python-sdkfolder。

运行图表网络训练示例

训练深度图表网络

1. 从JupyterLab在 Amazon SageMaker 中查看，浏览示例笔记本然后查找DGL文件夹。可以包含多个文件来支持示例。查看自述文件以了解任何先决条件。
2. 运行 .ipynb 笔记本示例。
3. 查找评估程序函数，并记下它在 DGL 和特定实例类型中使用 Amazon ECR 容器的行。您可能需要更新此项以使用首选区域中的容器。
4. 运行此函数启动实例并使用 DGL 容器来训练图表网络。启动此实例将会产生费用。在训练完成后，实例将自行终止。

示例

提供了知识图表嵌入 (KGE) 的示例。它使用 Freebase 数据集，这是一般事实的知识库。一个示例使用案例是绘制人员关系图并预测人员的国籍。

图表卷积网络 (GCN) 的示例实施说明如何训练图表网络来预测毒性。生理学数据集 Tox21 提供了物质如何影响生物反应的毒性测量。

另一个 GCN 示例说明如何在一个名为 Cora 的科学出版物书目数据集上训练图表网络。您可以使用它查找作者、主题和会议之间的关系。

最后一个示例是电影评论的推荐系统。它使用在 MovieLens 数据集上训练的图表卷积矩阵完成 (GCMC) 网络。这些数据集包含电影片名、类型和用户评分。

将深度学习容器与 DGL 结合使用

以下示例使用预配置的深度学习容器。这是最容易尝试的，因为它可以在 Amazon SageMaker 上直接使用。

- 使用 GCN 的知识库的半监督分类

使用 DGL 创建您自己的容器

以下示例允许您创建您自己的容器 (BYOC)。阅读 BYOC 指南，并在尝试它们前熟悉此过程。配置是必需的。

- 使用 GCN 预测毒性的分子特性
- 使用 GCMC 实施的电影推荐系统

扩展预构建的容器

如果预构建的 SageMaker 容器不能满足您的所有要求，则可以扩展现有映像以满足您的需求。即使您的环境或框架直接支持，您也可能需要添加其他功能或以不同的方式配置容器环境。通过扩展预构建的映像，您

可以利用随附的深度学习库和设置，而无需从头开始创建映像。您可以扩展容器以添加库、修改设置和安装其他依赖项。

以下教程演示了如何扩展预构建的 SageMaker 映像并将其发布到 Amazon ECR。

主题

- 扩展预构建容器的要求 (p. 2406)
- 扩展 SageMaker 容器以运行 Python 脚本 (p. 2406)

扩展预构建容器的要求

要扩展预先构建的 SageMaker 映像，您需要在 Dockerfile 中设置以下环境变量。有关使用 SageMaker 容器的环境变量的更多信息，请参阅SageMaker 培训工具包 GitHub 回购.

- `SAGEMAKER_SUBMIT_DIRECTORY`：用于训练的 Python 脚本所在容器中的目录。
- `SAGEMAKER_PROGRAM`：应该调用并用作训练切入点的 Python 脚本。

还可以通过在 Dockerfile 中包含以下内容来安装其他库：

```
RUN pip install <library>
```

以下教程介绍如何使用这些环境变量。

扩展 SageMaker 容器以运行 Python 脚本

在本教程中，您将学习如何使用使用 CIFAR-10 数据集的 Python 文件扩展 SageMaker PyTorch 容器。通过扩展 SageMaker PyTorch 容器，您可以利用与 SageMaker 配合使用的现有培训解决方案。本教程扩展了训练图像，但是可以采取相同的步骤来扩展推理图像。有关可用映像的完整列表，请参阅可用的 Deep Learning Containers 映像.

要使用 SageMaker 容器运行您自己的训练模型，请通过 SageMaker 笔记本实例构建 Docker 容器。

第 1 步：创建 SageMaker 笔记本实例

1. 打开SageMaker 控制台.
2. 在左侧导航窗格中，选择笔记本，选择笔记本实例，然后选择创建笔记本实例.
3. 在 Create notebook instance (创建笔记本实例) 页面上提供以下信息：
   a. 对于 Notebook instance name (笔记本实例名称)，输入 **`RunScriptNotebookInstance`**。
   b. 对于 Notebook instance type (笔记本实例类型)，选择 **`ml.t2.medium`**。
   c. 在权限和加密部分中，执行以下操作：
      i. 对于 IAM Role (IAM 角色)，选择 Create a new role (创建新角色)。
      ii. 在存储库的创建 IAM 角色页面上，选择特定的 S3 存储桶中，指定一个名为的 Amazon S3 存储桶**`sagemaker-run-script`**，然后选择创建角色.

      SageMaker 创建 IAM 角色`AmazonSageMaker-ExecutionRole-`*`YYYYMMDD`*`T`*`HHmmSS`*之外的压缩算法（例如`AmazonSageMaker-ExecutionRole-20190429T110788`. 请注意，执行角色命名约定使用创建角色的日期和时间，由 `T` 分隔。
   d. 适用于根访问，选择启用.
   e. 选择创建笔记本实例。
4. 在存储库的笔记本实例页面，状态是Pending. Amazon SageMaker 可能需要几分钟的时间来启动机器学习计算实例（在此例中为启动笔记本实例）并将 ML 存储卷附加到实例。笔记本实例有一个预配置的 Jupyter 笔记本服务器和一组 Anaconda 库。有关更多信息，请参阅 。CreateNotebookInstance.

5. 在权限和加密部分，复制IAM 角色 ARN 号码，然后将其粘贴到记事本文件中以临时保存它。稍后，您可以使用此 IAM 角色 ARN 编号在笔记本实例中配置本地训练估算器。The IAM role ARN number (IAM 角色 ARN 编号) 如下所示：`'arn:aws:iam::111122223333:role/service-role/AmazonSageMaker-ExecutionRole-20190429T110788'`
6. 在笔记本实例的状态更改为之后InService，选择打开 JupyterLab.

第 2 步：创建并上传 Dockerfile 和 Python 训练脚本

1. 打开 JupyterLab 后，请在 JupyterLab 的主目录中创建新文件夹。在左上角，选择新文件夹图标，然后输入文件夹名称`docker_test_folder`.
2. 创建`Dockerfile`中的文本文件`docker_test_folder`目录。

   a. 选择新启动器图标 (+) 位于左上角。

   b. 在右窗格中其他部分，选择文本文件.

   c. 粘贴以下内容`Dockerfile`示例代码放入文本文件。

```
# SageMaker PyTorch image
FROM 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training:1.5.1-cpu-py36-ubuntu16.04

ENV PATH="/opt/ml/code:${PATH}"

# this environment variable is used by the SageMaker PyTorch container to determine our user code directory.
ENV SAGEMAKER_SUBMIT_DIRECTORY /opt/ml/code

# /opt/ml and all subdirectories are utilized by SageMaker, use the /code subdirectory to store your user code.
COPY cifar10.py /opt/ml/code/cifar10.py

# Defines cifar10.py as script entrypoint
ENV SAGEMAKER_PROGRAM cifar10.py
```

   Dockerfile 脚本将执行以下任务：

   - `FROM 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training:1.5.1-cpu-py36-ubuntu16.04`— 下载 SageMaker PyTorch 基本映像。您可以将其替换为构建容器时想要带来的任何 SageMaker 基本映像。
   - `ENV SAGEMAKER_SUBMIT_DIRECTORY /opt/ml/code`— 套装`/opt/ml/code`作为训练脚本目录。
   - `COPY cifar10.py /opt/ml/code/cifar10.py`— 将脚本复制到 SageMaker 预期的容器内的位置。该脚本必须位于此文件夹中。
   - `ENV SAGEMAKER_PROGRAM cifar10.py`— 设置您的`cifar10.py`将训练脚本作为入口点脚本。

   d. 在左侧目录导航窗格中，文本文件名可能会自动命名为。`untitled.txt`. 要重命名文件，请右键单击文件，选择重命名中，将文件重命名为`Dockerfile`没有`.txt`扩展程序，然后按`Ctrl+s`要么`Command+s`保存文件。
3. 创建或上传训练脚本`cifar10.py`中的`docker_test_folder`. 您可以在本练习中使用以下示例脚本。

```
import ast
import argparse
import logging

import os
```

```
import torch
import torch.distributed as dist
import torch.nn as nn
import torch.nn.parallel
import torch.optim
import torch.utils.data
import torch.utils.data.distributed
import torchvision
import torchvision.models
import torchvision.transforms as transforms
import torch.nn.functional as F

logger=logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)

classes=('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship',
 'truck')


# https://github.com/pytorch/tutorials/blob/master/beginner_source/blitz/
cifar10_tutorial.py#L118
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1=nn.Conv2d(3, 6, 5)
        self.pool=nn.MaxPool2d(2, 2)
        self.conv2=nn.Conv2d(6, 16, 5)
        self.fc1=nn.Linear(16 * 5 * 5, 120)
        self.fc2=nn.Linear(120, 84)
        self.fc3=nn.Linear(84, 10)

    def forward(self, x):
        x=self.pool(F.relu(self.conv1(x)))
        x=self.pool(F.relu(self.conv2(x)))
        x=x.view(-1, 16 * 5 * 5)
        x=F.relu(self.fc1(x))
        x=F.relu(self.fc2(x))
        x=self.fc3(x)
        return x


def _train(args):
    is_distributed=len(args.hosts) > 1 and args.dist_backend is not None
    logger.debug("Distributed training - {}".format(is_distributed))

    if is_distributed:
        # Initialize the distributed environment.
        world_size=len(args.hosts)
        os.environ['WORLD_SIZE']=str(world_size)
        host_rank=args.hosts.index(args.current_host)
        dist.init_process_group(backend=args.dist_backend, rank=host_rank,
 world_size=world_size)
        logger.info(
            'Initialized the distributed environment: \'{}\' backend on {} nodes.
 '.format(
                args.dist_backend,
                dist.get_world_size()) + 'Current host rank is {}. Using cuda: {}.
 Number of gpus: {}'.format(
                dist.get_rank(), torch.cuda.is_available(), args.num_gpus))

    device='cuda' if torch.cuda.is_available() else 'cpu'
    logger.info("Device Type: {}".format(device))

    logger.info("Loading Cifar10 dataset")
    transform=transforms.Compose(
        [transforms.ToTensor(),
```

```
         transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

    trainset=torchvision.datasets.CIFAR10(root=args.data_dir, train=True,
                                          download=False, transform=transform)
    train_loader=torch.utils.data.DataLoader(trainset, batch_size=args.batch_size,
                                             shuffle=True, num_workers=args.workers)

    testset=torchvision.datasets.CIFAR10(root=args.data_dir, train=False,
                                         download=False, transform=transform)
    test_loader=torch.utils.data.DataLoader(testset, batch_size=args.batch_size,
                                            shuffle=False, num_workers=args.workers)

    logger.info("Model loaded")
    model=Net()

    if torch.cuda.device_count() > 1:
        logger.info("Gpu count: {}".format(torch.cuda.device_count()))
        model=nn.DataParallel(model)

    model=model.to(device)

    criterion=nn.CrossEntropyLoss().to(device)
    optimizer=torch.optim.SGD(model.parameters(), lr=args.lr, momentum=args.momentum)

    for epoch in range(0, args.epochs):
        running_loss=0.0
        for i, data in enumerate(train_loader):
            # get the inputs
            inputs, labels=data
            inputs, labels=inputs.to(device), labels.to(device)

            # zero the parameter gradients
            optimizer.zero_grad()

            # forward + backward + optimize
            outputs=model(inputs)
            loss=criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            # print statistics
            running_loss += loss.item()
            if i % 2000 == 1999:  # print every 2000 mini-batches
                print('[%d, %5d] loss: %.3f' %
                      (epoch + 1, i + 1, running_loss / 2000))
                running_loss=0.0
    print('Finished Training')
    return _save_model(model, args.model_dir)


def _save_model(model, model_dir):
    logger.info("Saving the model.")
    path=os.path.join(model_dir, 'model.pth')
    # recommended way from http://pytorch.org/docs/master/notes/serialization.html
    torch.save(model.cpu().state_dict(), path)


def model_fn(model_dir):
    logger.info('model_fn')
    device="cuda" if torch.cuda.is_available() else "cpu"
    model=Net()
    if torch.cuda.device_count() > 1:
        logger.info("Gpu count: {}".format(torch.cuda.device_count()))
        model=nn.DataParallel(model)

    with open(os.path.join(model_dir, 'model.pth'), 'rb') as f:
```

```
        model.load_state_dict(torch.load(f))
    return model.to(device)


if __name__ == '__main__':
    parser=argparse.ArgumentParser()

    parser.add_argument('--workers', type=int, default=2, metavar='W',
                        help='number of data loading workers (default: 2)')
    parser.add_argument('--epochs', type=int, default=2, metavar='E',
                        help='number of total epochs to run (default: 2)')
    parser.add_argument('--batch-size', type=int, default=4, metavar='BS',
                        help='batch size (default: 4)')
    parser.add_argument('--lr', type=float, default=0.001, metavar='LR',
                        help='initial learning rate (default: 0.001)')
    parser.add_argument('--momentum', type=float, default=0.9, metavar='M',
 help='momentum (default: 0.9)')
    parser.add_argument('--dist-backend', type=str, default='gloo', help='distributed
 backend (default: gloo)')

    # The parameters below retrieve their default values from SageMaker environment
 variables, which are
    # instantiated by the SageMaker containers framework.
    # https://github.com/aws/sagemaker-containers#how-a-script-is-executed-inside-the-
container
    parser.add_argument('--hosts', type=str,
 default=ast.literal_eval(os.environ['SM_HOSTS']))
    parser.add_argument('--current-host', type=str,
 default=os.environ['SM_CURRENT_HOST'])
    parser.add_argument('--model-dir', type=str, default=os.environ['SM_MODEL_DIR'])
    parser.add_argument('--data-dir', type=str,
 default=os.environ['SM_CHANNEL_TRAINING'])
    parser.add_argument('--num-gpus', type=int, default=os.environ['SM_NUM_GPUS'])

    _train(parser.parse_args())
```

第 3 步：构建容器

1. 在 JupyterLab 主目录中，打开 Jupyter 笔记本。要打开新笔记本，请选择新启动图标然后选择conda_pytorch_p36中的笔记本部分。
2. 在第一个笔记本单元格中运行以下命令以更改为docker_test_folder目录:

```
% cd ~/SageMaker/docker_test_folder
```

这将返回您当前的目录，如下所示：

```
! pwd
```

```
output: /home/ec2-user/SageMaker/docker_test_folder
```

3. 登录 Docker 以访问基本容器：

```
! aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin 763104351884.dkr.ecr.us-east-1.amazonaws.com
```

4. 要构建 Docker 容器，请运行以下 Docker 构建命令，包括结尾处的空格：

```
! docker build -t pytorch-extended-container-test .
```

Docker 构建命令必须从您创建的 Docker 目录运行，在此例中为。`docker_test_folder`.

Note

如果您收到以下错误消息提示 Docker 找不到 Dockerfile，请确保 Dockerfile 的名称正确并已保存到目录中。

```
unable to prepare context: unable to evaluate symlinks in Dockerfile path:
lstat /home/ec2-user/SageMaker/docker/Dockerfile: no such file or directory
```

请记住那个`docker`查找专门名为的文件`Dockerfile`在当前目录中没有任何扩展名。如果您将其命名为其他名称，则可以使用`-f`标志。例如，如果你命名了 Dockerfile`Dockerfile-text.txt`中，运行以下命令：

```
! docker build -t tf-custom-container-test -f Dockerfile-text.txt .
```

第 4 步：测试容器

1. 要在笔记本实例中本地测试容器，请打开 Jupyter 笔记本。选择新启动器然后选择笔记本在**`conda_pytorch_p36`**框架。其余的代码片段必须从 Jupyter 笔记本实例中运行。
2. 下载 CIFAR-10 数据集。

```
import torch
import torchvision
import torchvision.transforms as transforms

def _get_transform():
    return transforms.Compose(
        [transforms.ToTensor(),
         transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])


def get_train_data_loader(data_dir='/tmp/pytorch/cifar-10-data'):
    transform=_get_transform()
    trainset=torchvision.datasets.CIFAR10(root=data_dir, train=True,
                                          download=True, transform=transform)
    return torch.utils.data.DataLoader(trainset, batch_size=4,
                                       shuffle=True, num_workers=2)


def get_test_data_loader(data_dir='/tmp/pytorch/cifar-10-data'):
    transform=_get_transform()
    testset=torchvision.datasets.CIFAR10(root=data_dir, train=False,
                                         download=True, transform=transform)
    return torch.utils.data.DataLoader(testset, batch_size=4,
                                       shuffle=False, num_workers=2)

trainloader=get_train_data_loader('/tmp/pytorch-example/cifar-10-data')
testloader=get_test_data_loader('/tmp/pytorch-example/cifar-10-data')
```

3. Set`role`转到用于创建 Jupyter 笔记本的角色。这用于配置您的 SageMaker 估算器。

```
from sagemaker import get_execution_role

role=get_execution_role()
```

4. 将以下示例脚本粘贴到笔记本代码单元中，以使用扩展容器配置 SageMaker Estator。

```
from sagemaker.estimator import Estimator

hyperparameters={'epochs': 1}

estimator=Estimator(
    image_uri='pytorch-extended-container-test',
    role=role,
    instance_count=1,
    instance_type='local',
    hyperparameters=hyperparameters
)

estimator.fit('file:///tmp/pytorch-example/cifar-10-data')
```

5. 运行代码单元。该测试输出训练环境配置、用于环境变量的值、数据源以及训练期间获得的损失和准确率。

第 5 步：将容器推送到 Amazon Elastic Container Registry (Amazon ECR)

1. 成功运行本地模式测试后，您可以将 Docker 容器推送到Amazon ECR然后用它来运行培训工作。

   在笔记本单元格中运行以下命令行。

```
%%sh

# Specify an algorithm name
algorithm_name=pytorch-extended-container-test

account=$(aws sts get-caller-identity --query Account --output text)

# Get the region defined in the current configuration (default to us-west-2 if none
 defined)
region=$(aws configure get region)

fullname="${account}.dkr.ecr.${region}.amazonaws.com/${algorithm_name}:latest"

# If the repository doesn't exist in ECR, create it.

aws ecr describe-repositories --repository-names "${algorithm_name}" > /dev/null 2>&1
if [ $? -ne 0 ]
then
aws ecr create-repository --repository-name "${algorithm_name}" > /dev/null
fi

# Log into Docker
aws ecr get-login-password --region ${region}|docker login --username AWS --password-
stdin ${fullname}

# Build the docker image locally with the image name and then push it to ECR
# with the full name.

docker build -t ${algorithm_name} .
docker tag ${algorithm_name} ${fullname}

docker push ${fullname}
```

2. 推送容器后，您可以从 SageMaker 环境中的任何位置调用 Amazon ECR 镜像。在下一个笔记本电脑单元中运行以下代码示例。

   如果要将此训练容器与 SageMaker Studio 结合使用以使用其可视化功能，还可以在 Studio 笔记本电脑单元中运行以下代码来调用训练容器的 Amazon ECR 映像。

```
import boto3

client=boto3.client('sts')
account=client.get_caller_identity()['Account']

my_session=boto3.session.Session()
region=my_session.region_name

algorithm_name="pytorch-extended-container-test"
ecr_image='{}.dkr.ecr.{}.amazonaws.com/{}:latest'.format(account, region,
 algorithm_name)

ecr_image
# This should return something like
# 12-digits-of-your-account.dkr.ecr.us-east-2.amazonaws.com/tf-2.2-test:latest
```

3. 使用ecr_image从上一步中检索以配置 SageMaker 估计器对象。以下代码示例配置 SageMaker PyTorch 估算器。

```
import sagemaker

from sagemaker import get_execution_role
from sagemaker.estimator import Estimator

estimator=Estimator(
    image_uri=ecr_image,
    role=get_execution_role(),
    base_job_name='pytorch-extended-container-test',
    instance_count=1,
    instance_type='ml.p2.xlarge'
)

# start training
estimator.fit()

# deploy the trained model
predictor=estimator.deploy(1, instance_type)
```

第 6 步：清理资源

完成使用入门示例后清除资源

1. 打开SageMaker 控制台，选择笔记本实例RunScriptNotebookInstance，选择操作，然后选择停止. 停止实例可能需要几分钟时间。
2. 在实例之后状态对 的更改Stopped，选择操作，选择Delete，然后选择Delete在对话框中。删除实例可能需要几分钟时间。删除后，笔记本实例将从表中消失。
3. 打开Amazon S3 控制台并删除为存储模型构件和训练数据集创建的存储桶。
4. 打开IAM 控制台然后删除 IAM 角色。如果您已创建权限策略，也可以将其删除。

> **Note**
>
> Docker 容器在运行后会自动关闭。您不需要删除该容器。

调整你自己的 Docker 容器以与 SageMaker 配合使用

您可以调整现有的 Docker 映像以便与 SageMaker 配合使用。您可能需要将现有的外部 Docker 映像与 SageMaker 当您的容器满足当前预构建的 SageMaker 映像不支持的功能或安全要求时。有两个工具包可以让你自己携带容器并对其进行调整以便与 SageMaker 配合使用：

- SageMaker 培训工具包
- SageMaker 推理工具包

以下主题介绍如何使用 SageMaker 训练和推理工具包：

主题

- 单个框架库 (p. 2414)
- 使用 SageMaker 训练和推理工具包 (p. 2414)
- 调整您自己的训练容器 (p. 2416)
- 调整自己的推理容器 (p. 2425)

单个框架库

除了 SageMaker 训练工具包和 SageMaker 推理工具包， SageMaker 此外，还提供了专为 TensorFlow、MXNet、PyTorch 和 Chainer 提供了专用工具包。下表提供指向 GitHub 包含每个框架及其各自服务工具包的源代码的存储库。链接的说明是为了使用 Python 开发工具包在 SageMaker 上运行训练算法和托管模型。这些单个库的功能包含在 SageMaker 培训工具包中， SageMaker 推理工具包。

框架	工具包源代码
`TensorFlow`	SageMaker TensorFlow 训练 SageMaker TensorFlow 服务
`MXNet`	SageMaker MXNet 训练 SageMaker MXNet 推理
`PyTorch`	SageMaker PyTorch 训练 SageMaker PyTorch 推理
`Chainer`	SageMaker Chainer SageMaker 容器

使用 SageMaker 训练和推理工具包

这些区域有：SageMaker 训练和SageMaker 推理工具包实施了一项功能，您需要这些功能来调整容器以在 SageMaker 上运行脚本、训练算法以及部署模型。安装后，此库会为用户定义以下内容：

- 用于存储代码和其他资源的位置。
- 包含要在容器启动时运行的代码的入口点。您的 Dockerfile 必须将需要运行的代码复制到与 SageMaker 兼容的容器预期的位置。

- 容器管理部署以进行训练和推理所需的其他信息。

SageMaker 工具包容器结构

当 SageMaker 训练模型时，它会在容器中创建以下文件夹结构。`/opt/ml`目录。

```
/opt/ml
### input
#   ### config
#   #   ### hyperparameters.json
#   #   ### resourceConfig.json
#   ### data
#       ### <channel_name>
#           ### <input data>
### model
#
### code
#
### output
#
### failure
```

当你运行模型时培训作业时，SageMaker 容器使用`/opt/ml/input/`目录，其中包含 JSON 文件，这些文件可以配置算法超参数以及用于分布式训练的网络布局。这些区域有：`/opt/ml/input/`目录还包含一些用于指定 SageMaker 访问存储在 Amazon Simple Storage Service (Amazon S3) 中的数据的文件。将 SageMaker 器将运行的脚本放置在`/opt/ml/code/`目录。您的脚本应该将算法生成的模型写入 `/opt/ml/model/` 目录。有关更多信息，请参阅 使用您自己的训练算法 (p. 2429)。

当您 时主办在 SageMaker 上进行推断的训练模型，您可以将模型部署到 HTTP 端点。作为对推理请求的响应，该模型进行实时预测。容器必须包含服务堆栈以处理这些请求。

在托管或批量转换容器中，模型文件位于在训练期间将它们写入到的同一个文件夹中。

```
/opt/ml/model
#
### <model files>
```

有关更多信息，请参阅 使用您自己的推理代码 (p. 2435)。

单个容器与多个容器

您可以为训练算法和推理代码提供单独的 Docker 镜像，也可以对两者使用一个 Docker 镜像。当创建 Docker 镜像以与 SageMaker 搭配使用时，请考虑以下事项：

- 提供两个 Docker 镜像可能会增加存储要求和成本，因为常见的库可能会重复。
- 通常对于训练和托管而言，容器越小，启动速度越快。模型训练速度更快，并且托管服务可通过更快速地自动扩展对流量增加做出反应。
- 您或许可以编写一个远小于训练容器的推理容器。这在使用 GPU 进行训练时尤其常见，但您的推理代码针对 CPU 进行了优化。
- 要求 Docker 容器在没有特权访问权限的情况下运行。
- 您构建的 Docker 容器和 SageMaker 提供的容器都可以向`Stdout`和`Stderr`文件。您的 SageMaker 将这些消息发送到您的 Amazon CloudWatch 日志Amazonaccount.

有关如何创建 SageMaker 容器以及如何在其中执行脚本的更多信息，请参阅SageMaker 培训工具包和SageMaker 推理工具包GitHub 上的存储库。它们还提供重要的环境变量和 SageMaker 容器提供的环境变量的列表。

调整您自己的训练容器

要运行您自己的训练模型，请使用Amazon SageMaker 培训工具包通过 Amazon SageMaker 笔记本实例。

第 1 步：创建 SageMaker 笔记本实例

1. 打开位于 https://console.aws.amazon.com/sagemaker/ 的 Amazon SageMaker 控制台。
2. 在左侧导航窗格中，选择笔记本，选择笔记本实例选择，然后选择创建笔记本实例.
3. 在 Create notebook instance (创建笔记本实例) 页面上提供以下信息：

   a. 对于 Notebook instance name (笔记本实例名称)，输入 **`RunScriptNotebookInstance`**。
   b. 对于 Notebook instance type (笔记本实例类型)，选择 **`ml.t2.medium`**。
   c. 在权限和加密部分中，执行以下操作：

      i. 对于 IAM Role (IAM 角色)，选择 Create a new role (创建新角色)。
      ii. 在存储库的创建 IAM 角色选择页面，选择特定的 S3 存储桶，指定一个名为的 Amazon S3 存储桶**`sagemaker-run-script`**选择，然后选择创建角色.

      IAM SageMaker 建一个名为的 IAM 角色`AmazonSageMaker-ExecutionRole-`*`YYYYMMDD`*`T`*`HHmmSS`*. 例如：`AmazonSageMaker-ExecutionRole-20190429T110788`。请注意，执行角色命名约定使用创建角色的日期和时间，由分隔。`T`.

   d. 适用于根访问，选择启用.
   e. 选择创建笔记本实例。
4. 在存储库的笔记本实例页面，状态是Pending. 启动机器学习计算实例（在这种情况下是启动笔记本实例），并将 ML 存储卷附加到该实例。笔记本实例有一个预配置的 Jupyter 笔记本服务器和一组 Anaconda 库。有关更多信息，请参阅 。CreateNotebookInstance.
5. 在权限和加密部分，复制IAM 角色 ARN 编号，然后将其粘贴到记事本文件中以临时保存它。稍后，您可以使用此 IAM 角色 ARN 编号在笔记本实例中配置本地训练估算器。The IAM role ARN number (IAM 角色 ARN 编号) 如下所示：`'arn:aws:iam::111122223333:role/service-role/AmazonSageMaker-ExecutionRole-20190429T110788'`
6. 在笔记本实例的状态更改为之后InService 中，选择打开 JupyterLab.

第 2 步：创建并上传 Dockerfile 和 Python 训练脚本

1. 打开 JupyterLab 后，请在 JupyterLab 的主目录中创建一个新文件夹。在左上角选择新文件夹图标，然后输入文件夹名称`docker_test_folder`.
2. 创建`Dockerfile`中的文本文件`docker_test_folder`目录。

   a. 选择新启动器左上角的图标 (+)。
   b. 在右窗格中其他选择部分，选择文本文件.
   c. 粘贴以下内容`Dockerfile`示例代码放入文本文件中。

```
FROM tensorflow/tensorflow:2.2.0rc2-gpu-py3-jupyter

# Install sagemaker-training toolkit that contains the common functionality
 necessary to create a container compatible with SageMaker and the Python SDK.
RUN pip3 install sagemaker-training

# Copies the training code inside the container
COPY train.py /opt/ml/code/train.py
```

```
# Defines train.py as script entrypoint
ENV SAGEMAKER_PROGRAM train.py
```

Dockerfile 脚本将执行以下任务：

- `FROM tensorflow/tensorflow:2.2.0rc2-gpu-py3-jupyter`— 下载 TensorFlow Docker 基本映像。你可以用你想要用来构建容器的任何 Docker 基本映像替换它，也可以用 Amazon预构建容器基本映像。
- `RUN pip install sagemaker-training`— 安装SageMaker 培训练套件其中包含创建与 SageMaker 兼容的容器所必须使用的常见功能。
- `COPY train.py /opt/ml/code/train.py`— 将脚本复制到 SageMaker 预期的容器内的位置。该脚本必须位于此文件夹中。
- `ENV SAGEMAKER_PROGRAM train.py`— 拿你的训练脚本`train.py`作为入口点脚本复制在`/opt/ml/code`容器的文件夹。这是构建您自己的容器时唯一必须指定的环境变量。

d. 在左侧目录导航窗格中，文本文件名可能会自动命名为。`untitled.txt`. 要重命名文件，请右键单击文件，请选择重命名将文件重命名为`Dockerfile`没有`.txt`扩展，然后按`Ctrl+s`要么`Command+s`保存文件。

3. 创建或上传训练脚本`train.py`中的`docker_test_folder`. 为此练习，您可以使用以下示例脚本。

```
import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
tf.keras.layers.Flatten(input_shape=(28, 28)),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

model.fit(x_train, y_train, epochs=1)

model.evaluate(x_test, y_test)
```

第 3 步：构建容器

1. 在 JupyterLab 主目录中，打开 Jupyter 笔记本。要打开新笔记本，请选择新启动图标然后选择conda_tensorflow2_p36中的笔记本部分。
2. 在第一个笔记本单元格中运行以下命令以更改为`docker_test_folder`目录:

```
% cd ~/SageMaker/docker_test_folder
```

这将返回您当前的目录，如下所示：

```
! pwd
```

```
output: /home/ec2-user/SageMaker/docker_test_folder
```

3. 要构建 Docker 容器，请运行以下 Docker 构建命令，包括结尾的空格，后跟句点：

```
! docker build -t tf-custom-container-test .
```

Docker 构建命令必须从您创建的 Docker 目录运行，在此例中为。`docker_test_folder`.

Note

如果您收到以下错误消息提示 Docker 找不到 Dockerfile，请确保 Dockerfile 的名称正确并已保存到目录中。

```
unable to prepare context: unable to evaluate symlinks in Dockerfile path:
lstat /home/ec2-user/SageMaker/docker/Dockerfile: no such file or directory
```

请记住，`docker`查找专门名为的文件`Dockerfile`在当前目录中没有任何扩展名。如果您将其命名为其他名称，则可以使用`-f`标志。例如，如果您将 Dockerfile 命名为`Dockerfile-text.txt`，运行以下命令：

```
! docker build -t tf-custom-container-test -f Dockerfile-text.txt .
```

第 4 步：测试容器

1. 要在本地测试笔记本实例中的容器，请打开 Jupyter 笔记本。选择新启动器然后选择笔记本在**`conda_tensorflow_p36`**框架。
2. 将以下示例脚本粘贴到笔记本代码单元格中以配置 SageMaker Estemator。

   SageMaker Python SDK v1

   ```
   from sagemaker.estimator import Estimator

   estimator = Estimator(image_name='tf-custom-container-test',
                         role='arn:aws:iam::111122223333:role/role-name',
                         instance_count=1,
                         instance_type='local')

   estimator.fit()
   ```

   SageMaker Python SDK v2

   ```
   from sagemaker.estimator import Estimator

   estimator = Estimator(image_uri='tf-custom-container-test',
                         role='arn:aws:iam::111122223333:role/role-name',
                         instance_count=1,
                         instance_type='local')

   estimator.fit()
   ```

3. 替换`'Put_Your_ARN_Here'`与价值IAM 角色 ARN 编号在配置笔记本实例时复制到记事本文件中。ARN 应如下所示：`'arn:aws:iam::111122223333:role/service-role/AmazonSageMaker-ExecutionRole-20190429T110788'`.
4. 运行代码单元。该测试输出训练环境配置、用于环境变量的值、数据源以及训练期间获得的损失和准确率。

第 5 步：将容器推送到 Amazon Elastic Container Registry (Amazon ECR)

1. 成功运行本地模式测试后，您可以将 Docker 容器推送到Amazon ECR然后用它来运行培训工作。

   在笔记本单元格中运行以下命令行。

```
%%sh

# Specify an algorithm name
algorithm_name=tf-custom-container-test

account=$(aws sts get-caller-identity --query Account --output text)

# Get the region defined in the current configuration (default to us-west-2 if none
 defined)
region=$(aws configure get region)
region=${region:-us-west-2}

fullname="${account}.dkr.ecr.${region}.amazonaws.com/${algorithm_name}:latest"

# If the repository doesn't exist in ECR, create it.

aws ecr describe-repositories --repository-names "${algorithm_name}" > /dev/null 2>&1
if [ $? -ne 0 ]
then
aws ecr create-repository --repository-name "${algorithm_name}" > /dev/null
fi

# Get the login command from ECR and execute it directly

aws ecr get-login-password --region ${region}|docker login --username AWS --password-
stdin ${fullname}

# Build the docker image locally with the image name and then push it to ECR
# with the full name.

docker build -t ${algorithm_name} .
docker tag ${algorithm_name} ${fullname}

docker push ${fullname}
```

   Note

   此 bash shell 脚本可能会引发类似于以下错误消息的权限问题：

```
"denied: User: [ARN] is not authorized to perform: ecr:InitiateLayerUpload on
 resource:
arn:aws:ecr:us-east-1:[id]:repository/tf-custom-container-test"
```

   如果发生此错误，你需要附加AmazonEC2ContainerRegistryFullAccess策略您的 IAM 角色。转到IAM 控制台，选择角色从左侧导航窗格中，查找您用于笔记本实例的 IAM 角色。在许可选项卡上，选择附加策略按钮，然后搜索AmazonEC2ContainerRegistryFullAccess政策。标记策略的复选框，然后选择附加策略完成操作。

2. 推送容器后，您可以从 SageMaker 环境中的任何位置调用 Amazon ECR 镜像。在下一个笔记本电脑单元中运行以下代码示例。

   如果您想将此训练容器与 SageMaker Studio 一起使用其可视化功能，还可以在 Studio 笔记本电脑单元中运行以下代码来调用训练容器的 Amazon ECR 映像。

```
import boto3

account_id = boto3.client('sts').get_caller_identity().get('Account')
ecr_repository = 'tf-custom-container-test'
tag = ':latest'

region = boto3.session.Session().region_name

uri_suffix = 'amazonaws.com'
if region in ['cn-north-1', 'cn-northwest-1']:
    uri_suffix = 'amazonaws.com.cn'

byoc_image_uri = '{}.dkr.ecr.{}.{}/{}'.format(account_id, region, uri_suffix,
 ecr_repository + tag)

byoc_image_uri
# This should return something like
# 111122223333.dkr.ecr.us-east-2.amazonaws.com/sagemaker-byoc-test:latest
```

3. 使用`ecr_image`从上一步中检索以配置 SageMaker 估计器对象。以下代码示 SageMaker 使用`byoc_image_uri`并在 Amazon EC2 实例上启动培训作业。

   SageMaker Python SDK v1

   ```
   import sagemaker
   from sagemaker import get_execution_role
   from sagemaker.estimator import Estimator

   estimator = Estimator(image_uri=byoc_image_uri,
                         role=get_execution_role(),
                         base_job_name='tf-custom-container-test-job',
                         instance_count=1,
                         instance_type='ml.p2.xlarge')

   # start training
   estimator.fit()

   # deploy the trained model
   predictor = estimator.deploy(1, instance_type)
   ```

   SageMaker Python SDK v2

   ```
   import sagemaker
   from sagemaker import get_execution_role
   from sagemaker.estimator import Estimator

   estimator = Estimator(image_uri=byoc_image_uri,
                         role=get_execution_role(),
                         base_job_name='tf-custom-container-test-job',
                         instance_count=1,
                         instance_type='ml.p2.xlarge')

   # start training
   estimator.fit()

   # deploy the trained model
   predictor = estimator.deploy(1, instance_type)
   ```

有关展示如何在本地测试自定义容器并将其推送到 Amazon ECR 映像的完整示例，请参阅构建您自己的 TensorFlow 容器笔记本示例。

Tip

要分析和调试培训作业以监控系统利用率问题（例如 CPU 瓶颈和 GPU 利用率不足）并识别训练问题（例如过度拟合、过度训练、张量爆炸和渐变消失），请使用 Amazon SageMaker 调试器。有关更多信息，请参阅

Amazon SageMaker 调试器适用于您带给 Amazon SageMaker 的任何深度学习模型。这些区域有：Amazon CLI，SageMaker`EstimatorAPI` 和调试器 API 使您能够使用任何 Docker 基本映像来构建和自定义容器以训练模型。要将调试器与自定义容器一起使用，您需要对训练脚本进行最小的更改，以实现调试器钩子回调并从训练作业中检索张量。

您需要以下资源来使用调试器构建自定义容器。

- Amazon SageMaker Python 开发工具包
- SMDebug 开源客户端库
- 你选择的 Docker 基本映像
- 注册了调试器钩子的训练脚本 — 有关将调试器钩子注册到训练脚本的更多信息，请参阅将调试器挂钩注册到你的训练脚本 (p. 1869).

有关将调试程序与自定义训练容器结合使用的端到端示例，请参阅以下示例笔记本。

- 使用调试器构建自定义训练容器和调试训练作业

  Tip

  这个带调试器指南的自定义容器是调整您自己的训练容器 (p. 2416)指南向您详细介绍如何构建自定义培训容器并将其推送到 Amazon ECR。

准备构建自定义训练容器

要构建 docker 容器，文件的基本结构应如下所示：

```
### debugger_custom_container_test_notebook.ipynb      # a
 notebook to run python snippet codes
### debugger_custom_container_test_folder              # this is a
 docker folder
    ###  your-training-script.py                       # your
 training script with Debugger hook
    ###  Dockerfile                                    # a
 Dockerfile to build your own container
```

将调试器挂钩注册到你的训练脚本

要调试模型训练，您需要在训练脚本中添加调试程序挂钩。

Note

此步骤是收集模型参数（输出张量）以调试模型训练所必需的。如果您只想监控和配置文件，则可跳过该挂钩注册步骤并排除`debugger_hook_config`构造估计器时的参数。

以下示例代码显示了使用 Keras resNet50 模型的训练脚本的结构以及如何将调试器挂接作为 Keras 回调函数传递以进行调试。要查找完整的训练脚本，请参阅使用 SageMaker 调试器挂钩的 TensorFlow 训练脚本.

```
# An example of training script (your-training-script.py)
import tensorflow.compat.v2 as tf
from tensorflow.keras.applications.resnet50 import ResNet50
import smdebug.tensorflow as smd

def train(batch_size, epoch, model, hook):

    ...
    model.fit(X_train, Y_train,
              batch_size=batch_size,
              epochs=epoch,
              validation_data=(X_valid, Y_valid),
              shuffle=True,

              # smdebug modification: Pass the Debugger hook in
 the main() as a Keras callback
              callbacks=[hook])

def main():
    parser=argparse.ArgumentParser(description="Train resnet50
 cifar10")

    # hyperparameter settings
    parser.add_argument(...)

    args = parser.parse_args()

    model=ResNet50(weights=None, input_shape=(32,32,3),
 classes=10)
```

在脚本模式下，挂钩配置部分将从您设置评估程序的脚本中删除。而将调试程序挂钩功能合并到训练脚本中，脚本模式下的 TensorFlow Keras resNet

训练脚本. 训练脚本在所需的 Tensorflow Keras 环境中导入 smdebug 库以与 Tensorflow ResNet50 算法进行通信。它还会手动实施 smdebug 挂钩功能，方法是在 train 函数中添加 callbacks=[hook] 参数（第 49 行），并添加通过 SageMaker Python 开发工具包提供的手动挂钩配置（第 89 行）。
该脚本模式示例在 TF 2.1 框架中运行训练作业，以便与 TF 2.1 示例中的零脚本更改进行直接比较。在脚本模式下设置调试程序的好处是可以灵活地选择框架版本未涵盖的框架版本。AmazonDeep Learning Containers。

- 以脚本模式在 PyTorch 容器中使用 Amazon SageMaker 调试程序
该笔记本在 PyTorch v1.3.1 框架中以脚本模式启用调试程序。SageMaker 容器支持 PyTorch v1.3.1，该示例显示了如何修改训练脚本的详细信息。默认情况下，SageMaker PyTorch 评估程序已处于脚本模式下。在该笔记本中，在评估程序配置中不包含激活 script_mode 的行。
本笔记本显示了更改的详细步骤原始 PyTorch 训练脚本到启用了调试器的修改版本. 此外，该示例还说明了如何使用调试程序内置规则检测训练问题（如梯度消失问题），以及如何使用调试程序试验功能调用和分析保存的张量。

创建和配置 Dockerfile

打开 SageMaker JupyterLab 并创建一个新文件夹，debugger_custom_container_test_folder在此示例中，要保存训练脚本并Dockerfile. 下面的代码示例是Dockerfile这包括必不可少的 docker 构建赞扬。将以下代码粘贴到Dockerfile并将其保存到文本文件。将训练脚本上传到同一个文件夹。

```
# Specify a docker base image

FROM tensorflow/tensorflow:2.2.0rc2-gpu-py3

RUN /usr/bin/python3 -m pip install --upgrade pip

RUN pip install --upgrade protobuf


# Install required packages to enable the SageMaker Python SDK and
 the smdebug library

RUN pip install sagemaker-training

RUN pip install smdebug

CMD ["bin/bash"]
```

如果你想使用预构建的Amazon深度学习容器镜像，请参阅AvailableAmazonDeep Learning Containers 映像.

构建自定义培训容器并将其推送到亚马逊 ECR

创建测试笔记本，debugger_custom_container_test_notebook.ipynb，并在笔记本单元中运行以下代码。这将访问debugger_byoc_test_docker目录中，使用指定的构建 dockeralgorithm_name，然后将码头容器推送到您的亚马逊 ECR 中。


```
import boto3


```

```
ecr_repository = 'sagemaker-debugger-mnist-byoc-tf2'
```

```
tag = ':latest'
```

```

```

```
region = boto3.session.Session().region_name
```

```

```

```
uri_suffix = 'amazonaws.com'
```

```
if region in ['cn-north-1', 'cn-northwest-1']:
```

```
    uri_suffix = 'amazonaws.com.cn'
```

```
byoc_image_uri = '{}.dkr.ecr.{}.{}/{}'.format(account_id, region,
```

```
 uri_suffix, ecr_repository + tag)
```

```

```

```
!docker build -t $ecr_repository docker
```

```
!$(aws ecr get-login --region $region --registry-ids $account_id
```

```
 --no-include-email)
```

```
!aws ecr create-repository --repository-name $ecr_repository
```

```
!docker tag {ecr_repository + tag} $byoc_image_uri
```

```
!docker push $byoc_image_uri
```

Tip

如果您使用其中一个Amazon深度学习容器基础映像，运行以下代码登录 Amazon ECR 并访问深度学习容器映像存储库。

```
! aws ecr get-login-password --region {region} |
```

```
 docker login --username Amazon --password-stdin
```

```
 763104351884.dkr.ecr.us-east-1.amazonaws.com
```

使用自定义训练容器运行和调试培训作业

在构建码头容器并将其推送到 Amazon ECR 之后，请使用训练脚本和调试器特定参数配置 SageMaker 估计器。执行之后`estimator.fit()`，调试器将收集输出张量、监控它们并检测训练问题。使用保存的张量，你可以使用`smdebug`核心功能和工具。使用 Amazon CloudWatch Events 配置调试器规则监控流程的工作流程和Amazon Lambda，只要调试器规则发现训练问题，就可以自动执行停止训练作业流程。

```
import sagemaker
```

```
from sagemaker.estimator import Estimator
```

```
from sagemaker.debugger import Rule, DebuggerHookConfig,
```

```
 CollectionConfig, rule_configs
```

```
)
```

```

# start training
estimator.fit()
```

(p. 1868)。

第 6 步：清理 资源

完成使用入门示例后清除资源

1. 打开SageMaker 控制台，选择笔记本实例RunScriptNotebookInstance，选择操作，然后选择停止. 停止实例可能需要几分钟时间。
2. 在实例之后状态对 的更改Stopped，选择操作，选择Delete选择，然后选择Delete在对话框中。删除实例可能需要几分钟时间。删除后，笔记本实例将从表中消失。
3. 打开Amazon S3 控制台删除为存储模型构件和训练数据集创建的存储桶。
4. 打开IAM 控制台然后删除 IAM 角色。如果您已创建权限策略，也可以将其删除。

   Note

   Docker 容器在运行后会自动关闭。您不需要删除该容器。

调整自己的推理容器

如果没有亚马逊 SageMaker 预构建的推理容器足以满足您的情况，并且您想使用自己的 Docker 容器，请使用SageMaker 推理工具包调整你的容器以使用 SageMaker 托管。调整容器以使其使用 SageMaker 托管，在一个或多个 Python 脚本文件中创建推理代码和 Dockerfile那会导入推理工具包。

推理代码包括推理处理程序、处理程序服务和入口点。在此示例中，它们存储为三个独立的 Python 文件。所有这三个 Python 文件必须与您的 Dockerfile 位于同一目录中。

有关 Jupyter 笔记本的示例，该笔记本显示了使用 SageMaker 推理工具包，请参阅亚马逊 SageMaker 使用自己的算法容器多模型终端节点.

第 1 步：创建推理处理程序

这些区域有： SageMaker 推理工具包构建在多模型服务器 (MMS) 上。MMS 需要使用一个 Python 脚本，它实施一些函数以加载模型、预处理输入数据、从模型中获得结果以及在模型处理程序中处理输出数据。

model_fn 函数

有默认的实现`model_fn`函数，命名为`default_model_fn`，在 SageMaker PyTorch 和 MxNet 推理工具包。默认实现会加载使用 torchscript 保存的模型（格式）`.pt`要么`.pth`. 如果模型需要自定义方法来加载，或者要在加载模型时执行额外的步骤，则必须实现`model_fn`function. 以下简单示例显示了一个实施`model_fn`加载 PyTorch 模型：

```
def model_fn(self, model_dir):
    import torch

    logger.info('model_fn')
    device = "cuda" if torch.cuda.is_available() else "cpu"
    with open(os.path.join(model_dir, 'model.pth'), 'rb') as f:
        model = torch.jit.load(f)
    return model.to(device)
```

input_fn 函数

这些区域有：input_fn函数负责反序列化你的输入数据，以便它可以传递给你的模型。它以输入数据和内容类型作为参数，并返回反序列化数据。SageMaker 推理工具包提供了一个默认实现，可反序列化以下内容类型：

- JSON
- CSV
- Numpy 数组
- NPZ

如果模型需要不同的内容类型，或者希望在将输入数据发送到模型之前对其进行预处理，则必须实现input_fnfunction. 以下示例显示的是一个简单实施的input_fnfunction.

```
from sagemaker_inference import content_types, decoder
def input_fn(self, input_data, content_type):
        """A default input_fn that can handle JSON, CSV and NPZ formats.

        Args:
            input_data: the request payload serialized in the content_type format
            content_type: the request content_type

        Returns: input_data deserialized into torch.FloatTensor or torch.cuda.FloatTensor
 depending if cuda is available.
        """
        return decoder.decode(input_data, content_type)
```

预测_fn 函数

这些区域有：predict_fn函数负责从模型中获取预测。它需要模型和返回的数据input_fn作为参数，然后返回预测。没有默认的实现predict_fn. 你必须自己实施它。以下是一个简单实施的predict_fn函数 PyTorch 模型。

```
def predict_fn(self, data, model):
        """A default predict_fn for PyTorch. Calls a model on data deserialized in
 input_fn.
        Runs prediction on GPU if cuda is available.

        Args:
            data: input data (torch.Tensor) for prediction deserialized by input_fn
            model: PyTorch model loaded in memory by model_fn

        Returns: a prediction
        """
        return model(data)
```

output_fn 函数

这些区域有：output_fn函数负责序列化predict_fn函数作为预测返回。这些区域有： SageMaker 推理工具包实现默认值output_fn序列化 Numpy 数组、JSON 和 CSV 的函数。如果模型输出任何其他内容类型，或者想在将数据发送给用户之前对其进行其他后处理，则必须实施自己的output_fnfunction. 下面显示了一个简单的output_fn函数 PyTorch 模型。

```
from sagemaker_inference import encoder
def output_fn(self, prediction, accept):
        """A default output_fn for PyTorch. Serializes predictions from predict_fn to JSON,
 CSV or NPY format.
```

```
        Args:
            prediction: a prediction result from predict_fn
            accept: type which the output data needs to be serialized

        Returns: output data serialized
        """
        return encoder.encode(prediction, accept)
```

第 2 步：实现处理程序服务

处理程序服务由模型服务器执行。处理程序服务实现initialize和handle方法。这些区域有：initialize方法在模型服务器启动时被调用，而handle方法将针对发送给模型服务器的所有传入推理请求调用。有关更多信息，请参阅 。自定义服务在多模型服务器文档中。以下是一个处理程序服务的示例：PyTorch 模型服务器。

```
from sagemaker_inference.default_handler_service import DefaultHandlerService
from sagemaker_inference.transformer import Transformer
from sagemaker_pytorch_serving_container.default_inference_handler import
 DefaultPytorchInferenceHandler


class HandlerService(DefaultHandlerService):
    """Handler service that is executed by the model server.
    Determines specific default inference handlers to use based on model being used.
    This class extends ``DefaultHandlerService``, which define the following:
        - The ``handle`` method is invoked for all incoming inference requests to the model
 server.
        - The ``initialize`` method is invoked at model server start up.
    Based on: https://github.com/awslabs/mxnet-model-server/blob/master/docs/
custom_service.md
    """
    def __init__(self):
        transformer =
 Transformer(default_inference_handler=DefaultPytorchInferenceHandler())
        super(HandlerService, self).__init__(transformer=transformer)
```

第 3 步：实现入口点

入口点通过调用处理程序服务来启动模型服务器。您可以在 Dockerfile 中指定入口点的位置。以下是入口点的示例。

```
from sagemaker_inference import model_server

model_server.start_model_server(handler_service=HANDLER_SERVICE)
```

第 4 步：编写 Dockerfile

在你的Dockerfile中，从步骤 2 中复制模型处理程序并将上一步中的 Python 文件指定为入口点Dockerfile. 以下是可以添加到 Dockerfile 中以复制模型处理程序并指定入口点的行的示例。有关推理容器的 Dockerfile 的完整示例，请参阅Dockerfile.

```
# Copy the default custom service file to handle incoming data and inference requests
COPY model_handler.py /home/model-server/model_handler.py

# Define an entrypoint script for the docker image
ENTRYPOINT ["python", "/usr/local/bin/entrypoint.py"]
```

第 5 步：构建并注册容器

现在，您可以构建容器并在 Amazon Elastic Container Registry (Amazon ECR) 中注册。示例笔记本中的以下 Shell 脚本构建容器并将其上传至您的 Amazon ECR 存储库。Amazonaccount.

> Note
>
> SageMaker 托管支持使用存储在 Amazon ECR 以外的存储库中的推理容器。有关信息，请参阅对实时推理容器使用私有 Docker 注册表 (p. 2437)。

以下 shell 脚本展示了如何构建和注册容器。

```
%%sh

# The name of our algorithm
algorithm_name=demo-sagemaker-multimodel

cd container

account=$(aws sts get-caller-identity --query Account --output text)

# Get the region defined in the current configuration (default to us-west-2 if none
 defined)
region=$(aws configure get region)
region=${region:-us-west-2}

fullname="${account}.dkr.ecr.${region}.amazonaws.com/${algorithm_name}:latest"

# If the repository doesn't exist in ECR, create it.
aws ecr describe-repositories --repository-names "${algorithm_name}" > /dev/null 2>&1

if [ $? -ne 0 ]
then
    aws ecr create-repository --repository-name "${algorithm_name}" > /dev/null

# Get the login command from ECR and execute it directly
aws ecr get-login-password --region ${region}|docker login --username AWS --password-stdin
 ${fullname}

# Build the docker image locally with the image name and then push it to ECR
# with the full name.

docker build -q -t ${algorithm_name} .
docker tag ${algorithm_name} ${fullname}

docker push ${fullname}
```

现在，您可以使用此容器在 SageMaker 中部署终端节点。有关如何在 SageMaker 中部署终端节点的示例，请参阅亚马逊 SageMaker 使用自己的算法容器多模型终端节点.

使用自己的算法和模型创建容器

如果没有现有 SageMaker 容器满足您的需求，而且您没有自己的现有容器，您可能需要创建一个新的 Docker 容器。以下部分介绍了如何使用训练和推理算法创建 Docker 容器，以便与 SageMaker 一起使用。

主题

- 使用您自己的训练算法 (p. 2429)
- 使用您自己的推理代码 (p. 2435)

使用您自己的训练算法

本节介绍 Amazon SageMaker 如何与运行您的自定义训练算法的 Docker 容器进行交互。使用此信息编写代码并为训练算法创建 Docker 镜像。

主题

- Amazon SageMaker 如何运行您的训练映像 (p. 2429)
- Amazon SageMaker 如何提供训练信息 (p. 2430)
- 使用 EFA 运行训练 (p. 2432)
- Amazon SageMaker 如何发出算法成功和失败信号 (p. 2434)
- Amazon SageMaker 如何处理训练输出 (p. 2434)

Amazon SageMaker 如何运行您的训练映像

要配置 Docker 容器以作为可执行文件运行，请使用 Dockerfile 中的 `ENTRYPOINT` 指令。请注意以下几点：

- 对于模型训练，Amazon SageMaker 会按如下方式运行容器：

```
docker run image train
```

SageMaker 会覆盖任何默认`CMD`通过指定`train`映像名称后的参数。`train` 参数也覆盖您使用 Dockerfile 中的 `CMD` 提供的参数。

- 在你的 Dockerfile 中，使用`exec`的形式`ENTRYPOINT`说明：

```
ENTRYPOINT ["executable", "param1", "param2", ...]
```

例如：

```
ENTRYPOINT ["python", "k-means-algorithm.py"]
```

`exec` 形式的 `ENTRYPOINT` 指令直接启动可执行文件，而不是 `/bin/sh` 的子级。这使其能够接收如下信号`SIGTERM`和`SIGKILL`来自 SageMaker API。请注意以下几点：

- 这些区域有：`CreateTrainingJob`API 有一个停止条件，它指示 SageMaker 在特定的时间后停止模型训练。

- 这些区域有：`StopTrainingJob`API 发出等效于`docker stop`，使用 2 分钟超时命令，以便正常停止指定容器：

```
docker stop -t120
```

该命令尝试通过发送 `SIGTERM` 信号来停止正在运行的容器。超时 2 分钟后，API 会发送`SIGKILL`然后强行停止集装箱。如果容器处理`SIGTERM`不会在收到它后的 120 秒内退出`SIGKILL`发送到。

> **Note**
>
> 如果您希望在 SageMaker 停止训练之后访问中间模型构件，则添加代码以处理在`SIGTERM`处理程序。

- 如果您计划使用 GPU 设备进行模型训练，请确保您的容器与 `nvidia-docker` 兼容。容器只包含 CUDA 工具包；不要将 NVIDIA 驱动程序与映像捆绑。有关 `nvidia-docker` 的更多信息，请参阅 NVIDIA/nvidia-docker。
- 您不能使用`tini`将初始化程序作为 SageMaker 容器中的入口点，因为它会被`train`和`serve`参数。
- `/opt/ml`并且 SageMaker 训练会保留和所有子目录。构建算法的 Docker 镜像时，请确保不要将算法所需的任何数据放置在其下面，因为数据在训练期间可能不再可见。

Amazon SageMaker 如何提供训练信息

本节介绍 SageMaker 如何为您的 Docker 容器提供训练信息，如训练数据、超级参数和其他配置信息。

当你发送`CreateTrainingJob`请求 SageMaker 启动模型训练时，您指定包含训练算法的 Docker 镜像的 Elastic Container Registry (Amazon ECR) 路径。您还指定存储训练数据的 Amazon Simple Storage Service (Amazon S3) 位置以及特定于算法的参数。SageMaker 为 Docker 容器提供此信息，以便您的训练算法可以使用它。本节介绍我们如何向您的 Docker 容器提供此信息。有关创建训练作业的信息，请参阅`CreateTrainingJob`。有关 SageMaker 容器组织信息的更多信息，请参阅使用 SageMaker 训练和推理工具包 (p. 2414).

主题

- 超参数 (p. 2430)
- 环境变量 (p. 2430)
- 输入数据配置 (p. 2430)
- 训练数据 (p. 2431)
- 分布式训练配置 (p. 2432)

超参数

SageMaker 在`CreateTrainingJob`在 Docker 容器中提供请求`/opt/ml/input/config/hyperparameters.json`文件。

环境变量

在容器中设置以下环境变量：

- TRAIN_JOB_NAME — 在`TrainingJobName`的参数`CreateTrainingJob`请求.
- TRINATIN_JOB_ARN — 作为`TrainingJobArn`中的`CreateTrainingJob`响应。
- 中指定的所有环境变量环境中的参数`CreateTrainingJob`请求.

输入数据配置

您在`InputDataConfig`参数`CreateTrainingJob`请求. SageMaker 在`/opt/ml/input/config/inputdataconfig.json`文件在 Docker 容器中。

例如，假设您指定三个数据通道（`train`、`evaluation`, 和`validation`) 在您的请求中。SageMaker 提供以下 JSON：

```
{
  "train" : {"ContentType":  "trainingContentType",
             "TrainingInputMode": "File",
             "S3DistributionType": "FullyReplicated",
             "RecordWrapperType": "None"},
```

```
  "evaluation" : {"ContentType":  "evalContentType",
                  "TrainingInputMode": "File",
                  "S3DistributionType": "FullyReplicated",
                  "RecordWrapperType": "None"},
  "validation" : {"TrainingInputMode": "File",
                  "S3DistributionType": "FullyReplicated",
                  "RecordWrapperType": "None"}
}
```

Note

只向 SageMaker 器提供每个数据通道的相关信息（例如，通道名称和内容类型），如下所示。`S3DistributionType`将被设置为`FullyReplicated`如果指定 EFS 或 fsxLustre 作为输入数据源。

训练数据

这些区域有：`TrainingInputMode`中的参数`AlgorithmSpecification`的`CreateTrainingJob`请求指定了如何提供训练数据集。提供以下输入模式：

- **`File`模式**
  - `TrainingInputMode`写入到的参数`inputdataconfig.json`：“文件”
  - Docker 容器中的数据通道目录：`/opt/ml/input/data/`*`channel_name`*
  - 支持的数据源：Amazon Simple Storage Service (Amazon S3)、Amazon EFS 和 Amazon FSx for Lustre

  为每个通道创建一个目录。例如，如果您有三个频道`training`、`validation`, 和`testing`，SageMaker 在 Docker 容器中创建了三个目录：
  - `/opt/ml/input/data/training`
  - `/opt/ml/input/data/validation`
  - `/opt/ml/input/data/testing`

  Note

  使用 Amazon EFS 和 Amazon FSx 等文件系统数据源的通道必须使用`File`模式。如果指定了文件系统，则通道中提供的目录路径将挂载在 `/opt/ml/input/data/channel_name` 中。
- **`FastFile`模式**
  - `TrainingInputMode`写入到的参数`inputdataconfig.json`：“快速文件”
  - Docker 容器中的数据通道目录：`/opt/ml/input/data/`*`channel_name`*
  - 支持的数据源：Amazon S3

  通道目录以只读方式挂载。

  支持的算法`File`模式可以无缝协作`FastFile`没有代码更改的模式。

  Note

  使用的频道`FastFile`模式必须使用`S3DataType`的 “S3Preix”。
  `FastFile`模式显示了一个使用正斜杠的文件夹视图 (/) 作为将 Amazon S3 对象分组到文件夹中的分隔符。`S3Uri`前缀不能与部分文件夹名称对应。例如，如果 Amazon S3 数据集包含`s3://my-bucket/train-01/data.csv`，那么都没有`s3://my-bucket/train`也不`s3://my-bucket/train-01`允许`S3Uri`前缀。
  建议使用尾随的正斜杠来定义与文件夹对应的频道。例如，`s3://my-bucket/train-01/`的频道`train-01`folder。如果没有尾随的正斜杠，如果存在另一个文件夹，该频道将会模糊不清`s3://my-bucket/train-011/`或者文件`s3://my-bucket/train-01.txt/`.
- **`Pipe`模式**
  - `TrainingInputMode`写入到的参数`inputdataconfig.json`：“管道”

- Docker 容器中的数据通道目录：`/opt/ml/input/data/channel_name_epoch_number`
- 支持的数据源：Amazon S3

你需要为每个频道从单独的管道中读取。例如，如果您有三个频道`training`、`validation`, 和`testing`，你需要从以下管道中读取：

- `/opt/ml/input/data/training_0, /opt/ml/input/data/training_1, ...`
- `/opt/ml/input/data/validation_0, /opt/ml/input/data/validation_1, ...`
- `/opt/ml/input/data/testing_0, /opt/ml/input/data/testing_1, ...`

按顺序读取管道。例如，如果您有一个名为 `training` 的通道，请按以下顺序读取管道：

1. 打开`/opt/ml/input/data/training_0`在读取模式下，并将其读取到文件末尾 (EOF)；如果完成第一个纪元，则尽早关闭该管道文件。
2. 关闭第一个管道文件后，请查找 `/opt/ml/input/data/training_1` 并读取它，直到您完成第二个纪元，以此类推。

如果给定纪元的文件尚不存在，您的代码可能需要重试，直到创建该管道。例如，你可以阅读多个纪元`training`频道只能开始阅读`validation`准备好时频道。或者，如果算法要求，您可以同时读取它们。

有关演示如何在自带容器时使用 Pipe 模式的 Jupyter 笔记本的示例，请参阅将自己的管道模式算法带给 Amazon SageMaker.

分布式训练配置

如果您使用多个容器执行分布式训练，SageMaker 会在`/opt/ml/input/config/resourceconfig.json`文件。

要启用容器间通信，此 JSON 文件应包含所有容器的信息。SageMaker 使此文件可用于两者`File`和`Pipe`模式算法。该文件提供以下信息：

- `current_host`— 容器网络中的当前容器的名称。例如：`algo-1`。您可以随时更改主机值。不要使用此变量的特定值编写代码。
- `hosts`容器网络中的所有容器的名称列表，按字典顺序排列。例如，`["algo-1", "algo-2", "algo-3"]` 用于三节点集群。容器可以使用这些名称来查找容器网络上的其他容器。您可以随时更改主机值。不要使用这些变量的特定值编写代码。
- `network_interface_name`— 向您的容器公开的网络接口的名称。例如，运行消息传递接口 (MPI) 的容器可以使用该信息设置网络接口名称。
- 请勿使用 `/etc/hostname` 或 `/etc/hosts` 中的信息，因为可能不准确。
- 算法容器可能无法立即获得主机名信息。我们建议当节点在集群中可用时，在主机名解析操作上添加重试策略。

下面是三节点集群的节点 1 上的示例文件：

```
{
    "current_host": "algo-1",
    "hosts": ["algo-1","algo-2","algo-3"],
    "network_interface_name":"eth1"
}
```

使用 EFA 运行训练

SageMaker 提供与的集成EFA设备以加快高性能计算 (HPC) 和机器学习应用程序的速度。这种集成允许您在运行分布式培训作业时利用 EFA 设备。您可以将 EFA 集成添加到您带到 SageMaker 的现有 Docker 容器中。以下信息概述了如何配置自己的容器，以便将 EFA 设备用于分布式培训作业。

先决条件

你的容器必须满足SageMaker 训练容器规格.

安装 EFA 和所需的软件包

你的容器必须下载并安装EFA 软件. 这使您的容器能够识别 EFA 设备，并提供兼容版本的 Libfabric 和 Open MPI。

必须在容器内安装和管理 MPI 和 NCCL 等任何工具，才能作为启用 EFA 的培训作业的一部分使用。以下示例说明如何修改启用 EFA 的容器的 Dockerfile 以安装 EFA、MPI、OFI、NCCL 和 NCCL-TEST。

> **Note**
>
> 在容器上使用 PyTorch 和 EFA 时，容器的 NCCL 版本应与 PyTorch 安装的 NCCL 版本匹配。要验证 PyTorch NCCL 版本，请使用以下命令：
>
> ```
> torch.cuda.nccl.version()
> ```

```
ARG OPEN_MPI_PATH=/opt/amazon/openmpi/
ENV NCCL_VERSION=2.7.8
ENV EFA_VERSION=1.11.2
ENV BRANCH_OFI=1.1.1

#################################################
## EFA and MPI SETUP
RUN cd $HOME \
  && curl -O https://s3-us-west-2.amazonaws.com/aws-efa-installer/aws-efa-installer-
${EFA_VERSION}.tar.gz \
  && tar -xf aws-efa-installer-${EFA_VERSION}.tar.gz \
  && cd aws-efa-installer \
  && ./efa_installer.sh -y --skip-kmod -g \

ENV PATH="$OPEN_MPI_PATH/bin:$PATH"
ENV LD_LIBRARY_PATH="$OPEN_MPI_PATH/lib/:$LD_LIBRARY_PATH"

#################################################
## NCCL, OFI, NCCL-TEST SETUP
RUN cd $HOME \
  && git clone https://github.com/NVIDIA/nccl.git -b v${NCCL_VERSION}-1 \
  && cd nccl \
  && make -j64 src.build BUILDDIR=/usr/local

RUN apt-get update && apt-get install -y autoconf
RUN cd $HOME \
  && git clone https://github.com/aws/aws-ofi-nccl.git -b v${BRANCH_OFI} \
  && cd aws-ofi-nccl \
  && ./autogen.sh \
  && ./configure --with-libfabric=/opt/amazon/efa \
       --with-mpi=/opt/amazon/openmpi \
       --with-cuda=/usr/local/cuda \
       --with-nccl=/usr/local --prefix=/usr/local \
  && make && make install

RUN cd $HOME \
  && git clone https://github.com/NVIDIA/nccl-tests \
  && cd nccl-tests \
  && make MPI=1 MPI_HOME=/opt/amazon/openmpi CUDA_HOME=/usr/local/cuda NCCL_HOME=/usr/local
```

创建容器时的注意事项

EFA 设备安装到容器中`/dev/infiniband/uverbs0`在容器可访问的设备列表下。在 p4d 实例上，容器可以访问 4 个 EFA 设备。EFA 设备可以在容器可访问的设备列表中找到，如下所示：

- /dev/infiniband/uverbs0
- /dev/infiniband/uverbs1
- /dev/infiniband/uverbs2
- /dev/infiniband/uverbs3

从中获取有关主机名、对等主机名和网络接口（对于 MPI）的信息resourceconfig.json提供给每个容器实例的文件，请参阅分布式训练配置. 您的容器通过默认的弹性网络接口 (ENI) 处理 Peer 节点之间的常规 TCP 流量，同时处理通过 EFA 设备的 OFI（内核绕过）流量。

验证您的 EFA 设备是否已被识别

要验证 EFA 设备是否已识别，请在容器内运行以下命令。

```
/opt/amazon/efa/bin/fi_info -p efa
```

您的输出应类似于以下内容。

```
provider: efa
    fabric: EFA-fe80::e5:56ff:fe34:56a8
    domain: efa_0-rdm
    version: 2.0
    type: FI_EP_RDM
    protocol: FI_PROTO_EFA
provider: efa
    fabric: EFA-fe80::e5:56ff:fe34:56a8
    domain: efa_0-dgrm
    version: 2.0
    type: FI_EP_DGRAM
    protocol: FI_PROTO_EFA
provider: efa;ofi_rxd
    fabric: EFA-fe80::e5:56ff:fe34:56a8
    domain: efa_0-dgrm
    version: 1.0
    type: FI_EP_RDM
    protocol: FI_PROTO_RXD
```

使用 EFA 运行训练作业

创建启用 EFA 的容器后，您可以使用 SageMaker Estystator 使用 EFA 运行训练作业，就像处理任何其他 Docker 映像一样。有关注册容器并将其用于培训的更多信息，请参阅调整自己的训练容器.

Amazon SageMaker 如何发出算法成功和失败信号

一种指示是否成功使用其进程退出代码的训练算法。

成功训练执行在退出时的退出代码为 0，而不成功训练执行在退出时的退出代码为非零。这些将被转换为Completed和Failed中的TrainingJobStatus返回方DescribeTrainingJob. 此退出代码是标准惯例，对于所有语言均可轻松实施。例如，在 Python 中，您可以使用sys.exit(1)向发送关于失败的退出信号，并且只运行到主例程结束将导致 Python 在退出时的代码为 0。

如果失败，该算法可将关于失败的描述写入到失败文件。有关详细信息，请参阅下一节。

Amazon SageMaker 如何处理训练输出

由于您的算法在容器中运行，它会生成包含训练作业状态以及模型和输出构件的输出。您的算法应将此信息写入到以下文件 (位于容器的)/output目录。Amazon SageMaker 按如下方式处理此目录中包含的信息：

- `/opt/ml/output/failure`如果训练失败，当所有算法输出 (例如日志记录) 完成后，您的算法应将关于失败的描述写入到此文件。在`DescribeTrainingJob`响应，SageMaker 会返回此文件中的前 1024 个字符`FailureReason`.

- `/opt/ml/model`— 您的算法应将所有最终模型项目写入到此目录。SageMaker 将此数据作为单个对象用 tar 压缩格式复制到您在`CreateTrainingJob`请求. 如果在一个训练任务中有多个容器写入到此目录，应确保不`file/directory`名字冲突。SageMaker 会将结果聚合到一个 tar 文件中并上传到 S3。

使用您自己的推理代码

你可以使用亚马逊 SageMaker 要与 Docker 容器进行交互，并通过下列两种方式之一运行您自己的推理代码：

- 要将您自己的推理代码与持久终端节点结合使用以一次获取一个预测，请使用 SageMaker 托管服务。
- 要使用您自己的推理代码以获取整个数据集的预测，请使用 SageMaker 批量转换。

主题

- 将您自己的推理代码用于托管服务 (p. 2435)
- 将您自己的推理代码用于批量转换 (p. 2440)

将您自己的推理代码用于托管服务

此部分介绍 Amazon SageMaker 如何与运行您自己的用于托管服务的推理代码的 Docker 容器进行交互。使用此信息编写推理代码并创建 Docker 镜像。

主题

- SageMaker 如何运行您的推理映像 (p. 2435)
- SageMaker 如何加载您的模型构件 (p. 2436)
- 容器如何使用请求 (p. 2436)
- 容器应如何响应推理请求 (p. 2437)
- 容器应如何响应运行状况检查 (Ping) 请求 (p. 2437)
- 对实时推理容器使用私有 Docker 注册表 (p. 2437)

SageMaker 如何运行您的推理映像

要配置容器以作为可执行文件运行，请使用 Dockerfile 中的 `ENTRYPOINT` 指令。请注意以下几点：

- 对于模型推理，SageMaker 将容器运行为：

```
docker run image serve
```

  SageMaker 会覆盖默认值`CMD`通过指定容器中的语句`serve`映像名后的参数。`serve` 参数覆盖您使用 Dockerfile 中的 `CMD` 命令提供的参数。

- 建议您使用 `exec` 形式的 `ENTRYPOINT` 指令：

```
ENTRYPOINT ["executable", "param1", "param2"]
```

例如：

```
ENTRYPOINT ["python", "k_means_inference.py"]
```

exec 形式的 ENTRYPOINT 指令直接启动可执行文件，而不是 /bin/sh 的子级。这使其能够接收信号，如SIGTERM和SIGKILL来自 SageMaker API 操作，这是一项要求。

例如，当您使用CreateEndpoint为了创建终端节点，SageMaker 会设置您在请求中指定的终端节点配置所需的 ML 计算实例数。SageMaker 在这些实例上运行 Docker 容器。

如果您减少为终端节点提供支持的实例数（通过调用UpdateEndpointWeightsAndCapacitiesAPI）中，SageMaker 将运行命令来在将终止的实例上停止 Docker 容器。此命令发送 SIGTERM 信号，然后在 30 秒后发送 SIGKILL 信号。

如果您更新终端节点（通过调用UpdateEndpointAPI）中，SageMaker 将启动另一组 ML 计算实例，并运行 Docker 容器 (其中包含您的推理代码)。然后，它会运行一条命令来停止以前的 Docker 容器。要停止 Docker 容器，请命令发送SIGTERM信号，然后它发送SIGKILL30 秒后发出信号。

- SageMaker 使用您在CreateModel请求为容器设置环境变量和 DNS 主机名，如下所示：
  - 它使用 ContainerDefinition.Environment 字符串到字符串映射来设置环境变量。
  - 它使用 ContainerDefinition.ContainerHostname 设置 DNS 主机名。
- 如果您计划将 GPU 设备用于模型推理 (通过在CreateEndpointConfig请求），请确保您的容器是nvidia-docker兼容。不要将 NVIDIA 驱动程序与镜像捆绑。有关 nvidia-docker 的更多信息，请参阅 NVIDIA/nvidia-docker。
- 您不能使用tini将初始化程序作为 SageMaker 容器中的入口点，因为它会被train和serve参数。

SageMaker 如何加载您的模型构件

在您的中CreateModel请求，容器定义包括ModelDataUrl参数，此参数标识存储模型构件的 S3 位置。SageMaker 使用此信息来确定从哪里复制模型构件。它会将构件复制到 /opt/ml/model 目录以供推理代码使用。

ModelDataUrl 必须指向 tar.gz 文件。否则，SageMaker 将不会下载该文件。

如果您在 SageMaker 中训练了模型，则模型构件将作为单个压缩的 tar 文件保存在 Amazon S3 中。如果您在 SageMaker 外部训练了模型，则需要创建此单个压缩的 tar 文件并将其保存到 S3 位置。在容器启动之前，SageMaker 将此 tar 文件解压缩到 /opt/ml/model 目录。

容器如何使用请求

容器必须实施在端口 8080 上响应 /invocations 和 /ping 的 Web 服务器。

容器应如何响应推理请求

为了获得推理结果，客户端应用程序将向 SageMaker 终端节点发送 POST 请求。有关更多信息，请参阅 `InvokeEndpoint` API。SageMaker 将请求传递给容器，并将来自容器的推理结果返回给客户端。请注意以下几点：

- SageMaker 将所有`POST`标头除外`InvokeEndpoint`. SageMaker 可能添加其他标头。推理容器必须能够安全地忽略这些额外标头。
- 要接收推理请求，容器必须有一个在端口 8080 上侦听的 Web 服务器，并且必须接受到 `POST` 终端节点的 `/invocations` 请求。
- 客户的模型容器必须在 250 毫秒内接受套接字连接请求。
- 客户的模型容器必须在 60 秒内响应请求。在响应之前，模型本身可能有最多 60 秒的处理时间。`/invocations`. 如果您的模型需要 50 到 60 秒的处理时间，则开发工具包套接字超时应设置为 70 秒。

容器应如何响应运行状况检查 (Ping) 请求

这些区域有：`CreateEndpoint`和`UpdateEndpoint`API 调用导致 SageMaker 启动新的推理容器。在容器启动后不久，SageMaker 开始将定期 GET 请求发送到`/ping`终端节点。

容器上的最简单要求是使用 HTTP 200 状态代码和空白正文进行响应。这告知 SageMaker 容器已准备好接受推理请求。`/invocations`终端节点。

如果容器未开始一致地使用 200s 做出响应来通过运行状况检查，则在启动后的 4 分钟期间内，`CreateEndPoint`将失败，使终端节点处于失败状态，并且请求的更新`UpdateEndpoint`将无法完成。

虽然最低限制供容器用来返回静态 200，但容器开发人员可使用此功能执行更深入的检查。请求超时开启`/ping`尝试为 2 秒。

对实时推理容器使用私有 Docker 注册表

默认情况下，Amazon SageMaker 托管使您能够使用存储在 Amazon ECR 中的图像来构建容器以进行实时推断。或者，您可以构建容器，以便从私有 Docker 注册表中的图像进行实时推断。私有注册表必须可以从账户中的 Amazon VPC 访问。您基于私有 Docker 注册表中存储的映像创建的模型必须配置为连接到可访问私有 Docker 注册表的同一 VPC。有关将您的模型连接到 VPC 的信息，请参阅授予权 SageMaker 托管终端节点对 Amazon VPC 中的资源的访问权限 (p. 2799).

您的 Docker 注册表必须使用来自已知公有证书颁发机构 (CA) 的 TLS 证书来保护。

> **Note**
>
> 您的私有 Docker 注册表必须允许来自您在 VPC 配置中为模型指定的安全组的入站流量，以便 SageMaker 托管能够从注册表中提取模型映像。
> 如果 VPC 内有开放互联网的路径，SageMaker 可以从 DockerHub 中提取模型图像。

主题

- 在亚马逊弹性容器注册表以外的私有 Docker 注册表中存储映像 (p. 2438)
- 使用来自私人 Docker 注册表的图像进行实时推理 (p. 2438)
- 允许 SageMaker 对私有 Docker 注册表进行身份验证 (p. 2439)
- 创建 Lambda 函数 (p. 2439)
- 向 Lambda 授予执行角色权限 (p. 2440)
- 为 Lambda 创建接口 VPC 终端节点 (p. 2440)

在亚马逊弹性容器注册表以外的私有 Docker 注册表中存储映像

要使用私有 Docker 注册表存储映像以供 SageMaker 实时推断，请创建一个可从 Amazon VPC 访问的私有注册表。有关创建 Docker 注册表的信息，请参阅部署注册表服务器在 Docker 文档中。Docker 注册表必须遵守以下要求：

- 注册表必须是HTTP API V2 Docker 注册表注册表。
- Docker 注册表必须可从您在`VpcConfig`在创建模型时指定的参数。

使用来自私人 Docker 注册表的图像进行实时推理

创建模型并将其部署到 SageMaker 托管时，可以指定它使用私有 Docker 注册表中的映像来构建推理容器。在`ImageConfig`中的对象`PrimaryContainer`你传递给调用的参数create_modelfunction.

将存储在私有 Docker 注册表中的图像用于推理容器

1. 创建映像配置对象并指定一个值`Vpc`(对于)`RepositoryAccessMode`字段中返回的子位置类型。

```
image_config = {
                    'RepositoryAccessMode': 'Vpc'
                }
```

2. 如果您的私人 Docker 注册表需要身份验证，请添加`RepositoryAuthConfig`对象转换到映像配置对象。对于`RepositoryCredentialsProviderArn`字段中的`RepositoryAuthConfig`对象中，指定对象的 Amazon 资源名称 (ARN)Amazon Lambda函数，提供凭据，允许 SageMaker 向您的私有 Docker 注册表进行身份验证。有关如何创建 Lambda 函数来提供身份验证的信息，请参阅允许 SageMaker 对私有 Docker 注册表进行身份验证 (p. 2439).

```
image_config = {
                    'RepositoryAccessMode': 'Vpc',
                    'RepositoryAuthConfig': {
                       'RepositoryCredentialsProviderArn':
 'arn:aws:lambda:Region:Acct:function:FunctionName'
                     }
                }
```

3. 创建您要传递到的主容器对象。`create_model`，使用您在上一步中创建的映像配置对象。

   在中提供你的图片摘要形式。如果您使用`:latest`标签，则存在 SageMaker 提取比预期更新版本的图像的风险。使用摘要表格可确保 SageMaker 提取预期的映像版本。

```
primary_container = {
    'ContainerHostname': 'ModelContainer',
    'Image': 'myteam.myorg.com/docker-local/my-inference-image:<IMAGE-TAG>',
    'ImageConfig': image_config
}
```

4. 指定您要传递到的模型名称和执行角色。`create_model`.

```
model_name = 'vpc-model'
execution_role_arn = 'arn:aws:iam::123456789012:role/SageMakerExecutionRole'
```

5. 为您的模型的 VPC 配置指定一个或多个安全组和子网。您的私有 Docker 注册表必须允许来自您指定的安全组的入站流量。您指定的子网必须与私有 Docker 注册表位于同一 VPC 中。

```
vpc_config = {
    'SecurityGroupIds': ['sg-0123456789abcdef0'],
    'Subnets': ['subnet-0123456789abcdef0','subnet-0123456789abcdef1']
```

```
}
```

6. 获取 Boto3 SageMaker 客户端。

```
import boto3
sm = boto3.client('sagemaker')
```

7. 通过调用来创建模型`create_model`，使用您在之前步骤中为`PrimaryContainer`和`VpcConfig`参数。

```
try:
    resp = sm.create_model(
        ModelName=model_name,
        PrimaryContainer=primary_container,
        ExecutionRoleArn=execution_role_arn,
        VpcConfig=vpc_config,
    )
except Exception as e:
    print(f'error calling CreateModel operation: {e}')
else:
    print(resp)
```

8. 最后，打电话创建 _endpoint_config和创建 _终端点使用您在上一步中创建的模型来创建托管终端节点。

```
endpoint_config_name = 'my-endpoint-config'
sm.create_endpoint_config(
    EndpointConfigName=endpoint_config_name,
    ProductionVariants=[
        {
            'VariantName': 'MyVariant',
            'ModelName': model_name,
            'InitialInstanceCount': 1,
            'InstanceType': 'ml.t2.medium'
        },
    ],
)

endpoint_name = 'my-endpoint'
sm.create_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=endpoint_config_name,
)

sm.describe_endpoint(EndpointName=endpoint_name)
```

允许 SageMaker 对私有 Docker 注册表进行身份验证

要从需要身份验证的私有 Docker 注册表中提取推理映像，请创建Amazon Lambda函数提供凭据，并在调用时提供 Lambda 函数的 Amazon 资源名称 (ARN)。create_model. 当 SageMaker 运行时`create_model`，它会调用您指定的 Lambda 函数，以获取凭据以向 Docker 注册表进行身份验证。

创建 Lambda 函数

创建Amazon Lambda返回带有以下形式的响应的函数：

```
def handler(event, context):
   response = {
      "Credentials": {"Username": "username", "Password": "password"}
   }
   return response
```

根据您为私有 Docker 注册表设置身份验证的方式，Lambda 函数返回的凭证可能意味着以下任一内容：

- 如果您将私有 Docker 注册表设置为使用基本身份验证，则这是用户名和密码向注册表进行身份验证。
- 如果您将私有 Docker 注册表设置为使用不记名令牌身份验证，则用户名和密码发送到您的授权服务器，该服务器返回一个持票令牌，然后可用于向私有 Docker 注册表进行身份验证。

向 Lambda 授予执行角色权限

您用来调用的执行角色`create_model`必须具备调用权限Amazon Lambda函数。将以下内容添加到您的执行角色的权限策略中。

```
{
    "Effect": "Allow",
    "Action": [
        "lambda:InvokeFunction"
    ],
    "Resource": [
        "arn:aws:lambda:*:*:function:*myLambdaFunction*"
    ]
}
```

其中mylambDAF 函数是 Lambda 函数的名称。有关编辑角色权限策略的信息，请参阅修改角色权限策略（控制台）中的Amazon Identity and Access Management用户指南.

> **Note**
>
> 具有`AmazonSageMakerFullAccess`与其附加的托管策略有权使用SageMaker以它的名义执行。

为 Lambda 创建接口 VPC 终端节点

创建接口终端节点，以便您的 Amazon VPC 可以与您的Amazon Lambda在不通过 Internet 发送流量的情况下运行。有关如何执行此操作的信息，请参阅为 Lambda 配置接口 VPC 终端节点中的Amazon Lambda开发人员指南.

SageMaker 托管通过您的 VPC 向`lambda.region.amazonaws.com`，来调用 Lambda 函数。如果您在创建接口终端节点时选择私有 DNS 名称，则 Amazon Route 53 会将调用路由到 Lambda 接口终端节点。如果你使用不同的 DNS 提供商，请务必映射`lambda.region.amazonaws.com`将其转到 Lambda 接口终端节点。

将您自己的推理代码用于批量转换

本部分介绍亚马逊如何 SageMaker 与运行您自己的用于批量转换的推理代码的 Docker 容器进行交互。使用此信息编写推理代码并创建 Docker 镜像。

主题

- 如何 SageMaker 运行您的推理映像 (p. 2440)
- 如何 SageMaker 加载您的模型构件 (p. 2441)
- 容器如何使用请求 (p. 2441)
- 容器应如何响应推理请求 (p. 2442)
- 容器应如何响应运行状况检查 (Ping) 请求 (p. 2442)

如何 SageMaker 运行您的推理映像

要配置容器以作为可执行文件运行，请使用 Dockerfile 中的 `ENTRYPOINT` 指令。请注意以下几点：

- 对于批量转换， SageMaker 以下方法运行容器：

```
docker run image serve
```

SageMaker 会覆盖默认值CMD通过在容器中指定语句：serve映像名后的参数。serve 参数覆盖您使用 Dockerfile 中的 CMD 命令提供的参数。

- 建议您使用 exec 形式的 ENTRYPOINT 指令：

```
ENTRYPOINT ["executable", "param1", "param2"]
```

例如：

```
ENTRYPOINT ["python", "k_means_inference.py"]
```

- SageMaker 设置中指定的环境变量CreateModel和CreateTransformJob在容器上。此外，将填充以下环境变量：
  - SAGEMAKER_BATCH始终设置为true当容器在 Batch 变换中运行时。
  - SAGEMAKER_MAX_PAYLOAD_IN_MB设置为通过 HTTP 发送到容器的最大负载。
  - SAGEMAKER_BATCH_STRATEGY设置为SINGLE_RECORD当容器被发送每次调用一条记录时对调用以及MULTI_RECORD当容器获得尽可能多的记录时，有效负载中的记录。
  - SAGEMAKER_MAX_CONCURRENT_TRANSFORMS设置为最大数/invocations可以同时打开的请求。

    Note

    最后三个环境变量来自用户发出的 API 调用。如果用户没有为它们设置值，则不会传递它们。在这种情况下，无论是默认值还是算法请求的值（响应/execution-parameters）被使用。
- 如果您计划将 GPU 设备用于模型推理（通过在您的中指定基于 GPU 的 ML 计算实例）。CreateTransformJob请求），请确保您的容器与 nvidia-docker 兼容。不要将 NVIDIA 驱动程序与镜像捆绑。有关 nvidia-docker 的更多信息，请参阅 NVIDIA/nvidia-docker。

- 不能使用 init 初始化程序作为 SageMaker 容器中的入口点，因为它会被训练参数和服务参数混淆。

如何 SageMaker 加载您的模型构件

在CreateModel请求，容器定义包括ModelDataUrl参数，此参数标识 Amazon S3 中存储模型构件的位置。当您使用 SageMaker 为了运行推理，它使用此信息来确定从何处复制模型构件。它将构件复制到 Docker 容器中的 /opt/ml/model 目录，以供您的推理代码使用。

ModelDataUrl 参数必须指向 tar.gz 文件。否则，SageMaker 将无法下载文件。如果在 SageMaker 中训练模型，则构件以单个 tar 压缩文件形式将构件保存在 Amazon S3 中。如果在另一个框架中训练模型，则需要以 tar 压缩文件形式将模型构件存储在 Amazon S3 中。 SageMaker 解压此 tar 文件并将其保存在/opt/ml/model批处理转换作业开始之前的容器中的目录。

容器如何使用请求

容器必须实施一个 Web 服务器，以响应端口 8080 上的调用和 ping 请求。对于批量转换，您可以选择设置算法以实施执行参数请求，以便为 SageMaker 提供动态运行时配置。SageMaker 使用以下终端节点：

- ping— 用于定期检查容器的运行状况。 SageMaker 等待 HTTP200在发送调用请求前，状态代码和成功 ping 请求的空正文。当调用请求发出后，您可以使用 ping 请求将模型加载到内存中，以生成推理。

- （可选）execution-parameters— 允许算法在运行时为作业提供最佳调整参数。根据容器可用的内存和 CPU，算法选择适当的MaxConcurrentTransforms、BatchStrategy, 和MaxPayloadInMB作业的值。

在调用调用请求之前， SageMaker 尝试调用执行参数请求。在创建批量转换作业时，您可以为MaxConcurrentTransforms、BatchStrategy, 和MaxPayloadInMB参数。 SageMaker 使用以下优先级顺序确定这些参数的值：

1. 在创建CreateTransformJob请求.
2. 在模型容器返回的值。 SageMaker 调用执行参数终端节点 >
3. 默认参数值，如下表所列。

参数	默认值
MaxConcurrentTransforms	1
BatchStrategy	MULTI_RECORD
MaxPayloadInMB	6

响应GET执行参数请求是一个 JSON 对象，其中包含以下密钥：MaxConcurrentTransforms、BatchStrategy, 和MaxPayloadInMB参数。以下是一个有效响应的示例：

```
{
“MaxConcurrentTransforms”: 8,
“BatchStrategy": "MULTI_RECORD",
"MaxPayloadInMB": 6
}
```

容器应如何响应推理请求

为了获得推论，亚马逊 SageMaker 向推理容器发送 POST 请求。POST 请求正文包含来自 Amazon S3 的数据。亚马逊 SageMaker 将请求传递到容器，并从容器返回推理结果，同时将响应中的数据保存到 Amazon S3。

要接收推理请求，容器必须有一个在端口 8080 上侦听的 Web 服务器，并且必须接受到针对 /invocations 终端节点的 POST 请求。推理请求超时和最大重试次数可以通过ModelClientConfig.

容器应如何响应运行状况检查 (Ping) 请求

容器上的最简单要求是使用 HTTP 200 状态代码和空白正文进行响应。这告知 SageMaker 容器已准备好接受推理请求/invocations终端节点。

虽然最低限制供容器用来返回静态 200，但容器开发人员可使用此功能执行更深入的检查。请求超时/ping尝试时间为 2 秒。

示例笔记本：使用您自己的算法或模型

以下 Jupyter 笔记本演示如何从 Amazon SageMaker 笔记本实例使用您自己的算法或预训练模型。有关指向包含 TensorFlow、MXNet、Chainer 和 PyTorch 框架的预构建 Dockerfile 的 GitHub 存储库的链接以及有关使用说明的链接，请参阅Amazon SDK for Python (Boto3)在 SageMaker 学习器上运行您自己的训练算法以及在 SageMaker 托管服务上运行您自己的模型的评估程序用于深度学习的预构建 SageMaker Docker 镜像 (p. 2401)

设置

1. 创建 SageMaker 笔记本实例。有关如何创建和访问 Jupyter笔 记本实例的说明，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147).
2. 打开您创建的笔记本实例。
3. 选择SageMaker 示例选项卡查看所有 SageMaker 示例笔记本的列表。
4. 从高级功能部分或从 GitHub 使用提供的链接从 GitHub 中进行访问。要打开笔记本，请选择其 Use (使用) 选项卡，然后选择 Create copy (创建副本)。

在 Scikit-Learn 中培训的主机模型

要了解如何通过将第一方 k-means 和 XGBoost 容器注入第一方 k-means 和 XGBoost 容器，托管在 SciKit 中训练的模型以在 SageMaker 中进行学习预测，请参阅以下示例笔记本。

- kmeans_bring_your_own_model
- xgboost_bring_your_own_model

封装 TensorFlow 和 Scikit-Learn 模型供 SageMaker 中使用

要了解如何打包在 TensorFlow 中开发的算法和 scikit-learn 框架，以便在 SageMaker 环境中进行训练和部署，请参阅以下笔记本。它们向您展示了如何使用 Dockerfile 构建、注册和部署您自己的 Docker 容器。

- tensorflow_bring_your_own
- scikit_bring_your_own

在 SageMaker 上训练和部署神经网络

要了解如何使用 MXNet 或 TensorFlow 在本地训练神经网络，然后从训练后的模型创建终端节点并将其部署在 SageMaker 上，请参阅以下笔记本。MXNet 模型经过训练，可以识别 MNIST 数据集中的手写数字。TensorFlow 模型经过训练，可以对虹膜进行分类。

- mxnet_mnist_byom
- tensorflow_iris_byom

使用管道模式训练

要了解如何使用 Dockerfile 构建调用`train.py script`并使用管道模式自定义训练算法，请参阅以下笔记本。在管道模式下，输入数据在训练时传输到算法。与使用文件模式相比，这可以减少训练时间。

- pipe_bring_your_own

自带 R 模型

要了解如何通过安装在笔记本中的 R 内核，使用 R 容器来训练和托管模型，请参阅以下笔记本。要充分利用 AmazonSDK for Python (Boto3)，我们使用笔记本中的 Python。您可以通过调用命令行参数在 R 中获得相同的结果。

- r_bring_your_own

扩展预构建的 PyTorch 容器映像

要了解在您具有预构建的 Docker 映像不支持的算法或模型的其他功能要求时如何扩展预构建的 SageMaker PyTorch 容器映像，请参阅以下笔记本。

- pytorch_extending_our_ 容器

在自定义容器上训练和调试培训作业

要了解如何使用 SageMaker 调试器训练和调试培训作业，请参阅以下笔记本。通过本示例提供的训练脚本使用 TensorFlow Keras resNet 50 模型和 CIFAR10 数据集。Docker 自定义容器使用训练脚本构建并推送至 Amazon ECR。在训练作业运行期间，调试器会收集张量输出并识别调试问题。与`smdebug`客户端库工具，你可以设置`smdebug`试用对象，该对象调用培训作业和调试信息，检查训练和调试器规则状态，并检索 Amazon S3 存储桶中保存的张量以分析训练问题。

- build_YOUR_own_容器_WIT_WIT_调试器

排除 Docker 容器故障排除

将 Docker 容器与 SageMaker 结合使用时，可能会遇到以下常见错误。每个错误之后都是错误的解决方案。

- Error: SageMaker 已经丢失 Docker 守护程序。

  要修复此错误，请使用以下命令重新启动 Docker。

  ```
  sudo service docker restart
  ```

- Error: 这些区域有：**`/tmp`**您 Docker 容器的目录已经用尽了空间。

  使用 Docker 容器`/tmp`和用于存储代码的分区。在本地模式下使用大型代码模块时，这些分区可以很容易填满。SageMaker Python SDK 支持为本地模式根目录指定自定义临时目录以避免此问题。

  要在 EBS 卷存储中指定自定义临时目录，请在以下路径中创建一个文件`~/.sagemaker/config.yaml`然后添加以下配置。您指定为的目录`container_root`必须已存在。SageMaker Python SDK 不会尝试创建它。

  ```
  local:
  container_root: /home/ec2-user/SageMaker/temp
  ```

  使用此配置，本地模式使用`/temp`目录而不是默认`/tmp`目录。

SageMaker 工作流

您可以管理亚马逊 SageMaker 使用亚马逊的训练和推理工作流 SageMaker 工作室和亚马逊 SageMaker Python 开发工具包。使用可用的工具，你可以简化你的 SageMaker 处理并将其集成到现有项目中。

支持以下工作流程技术。

- 亚马逊 SageMaker 模型构建管道 (p. 2445)：SageMaker 用于构建和管理的工具 end-to-end ML 管道。
- 气流工作流： SageMaker 用于导出用于创建和管理 Airflow 工作流的配置的 API。
- Kubernetes 编排 (p. 2543)： SageMaker Kubernetes 集群的自定义运算符以及 Kubeflow Pipeline 的自定义组件。
- Amazon Step Functions：在 Python 中创建多步骤机器学习工作流程，以编排 SageMaker 基础设施，而不必分别预置资源。

有关管理的更多信息 SageMaker 请参阅训练和推理Amazon SageMaker Python SDK 工作流.

主题

- 亚马逊 SageMaker 模型构建管道 (p. 2445)
- 使用实现自动化 mLOP SageMaker 项目 (p. 2507)
- 亚马逊 SageMaker ML 世系跟踪 (p. 2527)
- Kubernetes 编排 (p. 2543)
- 亚马逊 SageMaker 常见问题 (p. 2593)

亚马逊 SageMaker 模型构建管道

亚马逊 SageMaker 模型构建管道是利用直接利用机器学习管道构建机器学习管道的工具 SageMaker 集成。由于这种集成，您可以使用为自己处理大部分步骤创建和管理的工具来创建管道并设置 SageMaker Projects 以进行编排。 SageMaker 与其他管道相比，管道具备下述优点Amazon产品流程：

SageMaker 集成

SageMaker Pipeline 直接与 SageMaker 集成，因此您无需与任何其他人交互Amazon服务。您也不需要管理任何资源，因为 SageMaker Pipeline 是一项完全托管的服务，这意味着它为您创建和管理资源。

SageMaker Python SDK 集成

由于 SageMaker 管道与 SageMaker Python SDK，您可以使用可能已经熟悉的高级 Python 界面以编程方式创建管道。查看 SageMaker 要获得 Python SDK 文档，请参阅管道.

SageMaker Studio 集成

SageMaker Studio 提供了一个环境来管理 end-to-end SageMaker 管道体验。使用 Studio，你可以绕过Amazon用于整个工作流管理的控制台。有关管理的更多信息 SageMaker 来自的管道 SageMaker 工作室，请参阅查看、跟踪和执行 SageMaker 中的管道 SageMaker 工作室 (p. 2493).

数据谱系跟踪

与 SageMaker 管道您可以在管道执行过程中跟踪数据的历史记录。借助 Amazon SageMaker ML 世系追踪，您可以分析数据的来源、将其用作输入的位置以及从中生成的输出。例如，您可以查看从单个数据集

创建的模型，也可以查看创建单个模型的数据集。有关更多信息，请参阅 亚马逊 SageMaker ML 世系跟踪 (p. 2527)。

步骤重用

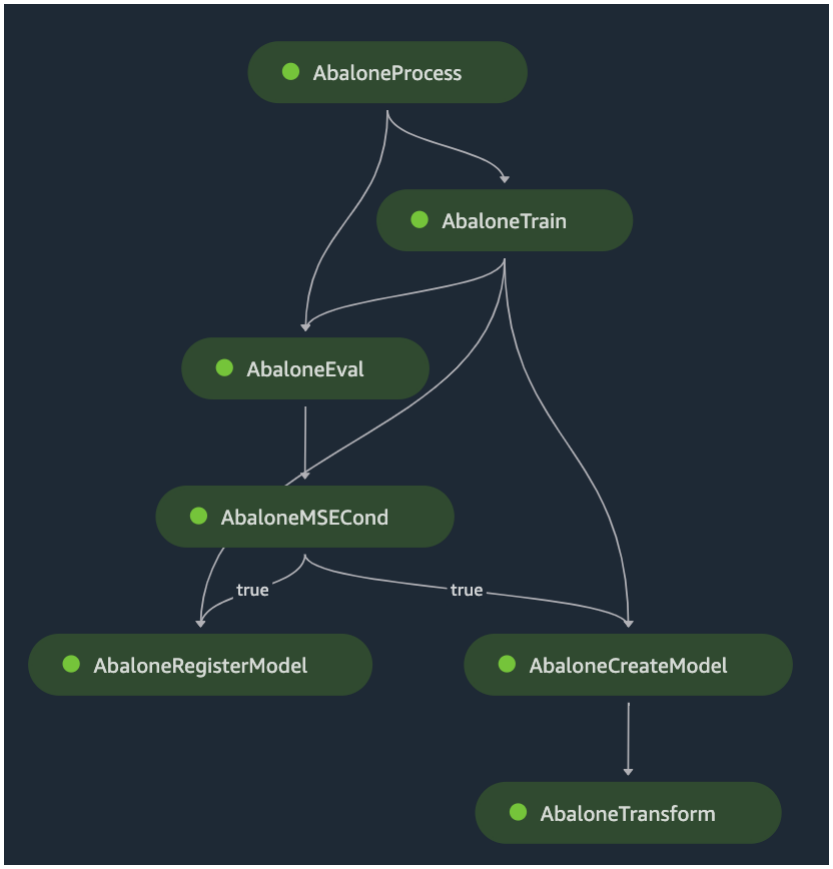
与 SageMaker 管道，您可以指定缓存步骤。缓存步骤时，如果再次执行同一步骤，则会对其进行索引以供稍后重复使用。因此，您可以在同一管道中重复使用同一步骤上一步执行的输出，而无需再次运行该步骤。有关步骤缓存的更多信息，请参阅缓存管道步骤 (p. 2466).

主题

- SageMaker 管道概述 (p. 2446)
- 创建和管理 SageMaker 管道 (p. 2478)

SageMaker 管道概述

Amazon SageMaker 模型构建管道管道是由 JSON 管道定义定义的一系列相互关联的步骤。此管道定义使用有向无环图 (DAG) 对管道进行编码。此 DAG 提供了有关管道每个步骤的要求和关系的信息。管道的 DAG 结构取决于步骤之间的数据依赖关系。这些数据依赖关系是在步骤输出的属性作为输入传递给另一个步骤时创建的。下图是管道 DAG 的示例：



以下主题介绍基本主题： SageMaker 管道概念。有关描述这些概念实现的教程，请参阅创建和管理 SageMaker 管道 (p. 2478).

主题

- 管道结构和执行 (p. 2447)
- IAM 访问管理 (p. 2448)

- 管道参数 (p. 2449)
- 管道步骤 (p. 2450)
- 属性文件和JsonGet (p. 2465)
- 缓存管道步骤 (p. 2466)
- 管道步骤的重试策略 (p. 2467)
- 基线计算、漂移检测和生命周期 ClarifyCheck 和 QualityCheck 亚马逊中的步骤 SageMaker 模型构建管道 (p. 2469)
- 亚马逊 EventBridge 集成 (p. 2473)
- 亚马逊 SageMaker 实验集成 (p. 2474)
- Amazon 故障排除 SageMaker 模型构建管道 (p. 2477)

管道结构和执行

主题

- 管道结构 (p. 2447)
- 使用并行度配置执行管道 (p. 2447)

管道结构

Amazon SageMaker 模型构建管道实例由name、parameters, 和steps. 管道名称必须唯一(account, region)对。步骤定义中使用的所有参数都必须在管道中定义。列出的流水线步骤通过彼此的数据依赖关系自动确定它们的执行顺序。这些区域有： SageMaker Pipeline 服务解决数据依赖关系 DAG 中步骤之间的关系，以创建执行完成的一系列步骤。以下是管道结构的示例。

```
from sagemaker.workflow.pipeline import Pipeline

  pipeline_name = f"AbalonePipeline"
  pipeline = Pipeline(
      name=pipeline_name,
      parameters=[
          processing_instance_type,
          processing_instance_count,
          training_instance_type,
          model_approval_status,
          input_data,
          batch_data,
      ],
      steps=[step_process, step_train, step_eval, step_cond],
  )
```

使用并行度配置执行管道

默认情况下，管道执行可 parallel 行运行的所有步骤。您可以使用ParallelismConfiguration属性在创建或更新管道时，以及启动或重试管道执行时。

并行度配置将在每次执行中应用。例如，如果启动了两个执行，它们每个执行最多可以同时运行 50 个步骤，共计 100 个并发运行步骤。另请参阅，ParallelismConfiguration启动、重试或更新执行时指定的 (s) 优先于管道中定义的并行度配置。

Example 使用创建管道执行**ParallelismConfiguration**

```
pipeline = Pipeline(
        name="myPipeline",
        steps=[step_process, step_train]
```

```
    )

pipeline.create(role, parallelism_config={"MaxParallelExecutionSteps": 50})
```

IAM 访问管理

以下部分介绍了Amazon Identity and Access Management(IAM) 亚马逊的要求 SageMaker 模型构建管道。有关如何实现这些权限的示例，请参阅。先决条件 (p. 2479).

主题

- 管道角色权限 (p. 2448)
- 管道步骤权限 (p. 2448)
- 使用管道进行服务控制策略 (p. 2448)

管道角色权限

您的管道需要一个传递给的 IAM 管道执行角色 SageMaker 创建管道时的管道。 的角色 SageMaker 正在创建管道的实例必须具有`iam:PassRole`为了传递管道执行角色的权限。 有关 IAM 角色的更多信息，请参阅IAM 角色.

您的管道执行角色需要以下权限：

- 要将任何角色传递给 SageMaker 在管道内的工作，`iam:PassRole`对正在传递的角色的权限。
- `Create`和`Describe`管道中每种作业类型的权限。
- Amazon S3 有权使用`JsonGet`function. 您使用基于资源的策略和基于身份的策略控制对 Amazon S3 资源的访问。基于资源的策略将应用于 Amazon S3 存储桶并授予。 SageMaker 管道访问存储桶。基于身份的策略使您的管道能够从您的账户拨打 Amazon S3 调用。有关基于资源的策略和基于身份的策略的更多信息，请参阅。基于身份的策略和基于资源的策略.

```
{
    "Action": [
        "s3:GetObject"
    ],
    "Resource": "arn:aws:s3:::<your-bucket-name>/*",
    "Effect": "Allow"
}
```

管道步骤权限

SageMaker 管道包括运行的步骤 SageMaker 个作业。为了使管道步骤运行这些作业，它们需要在您的账户中拥有 IAM 角色以提供对所需资源的访问权限。此角色被传递给 SageMaker 由你的管道提供服务主体。有关 IAM 角色的更多信息，请参阅IAM 角色.

默认情况下，每个步骤都采用管道执行角色。您可以选择将不同的角色传递给管道中的任何步骤。这可确保每个步骤中的代码无法影响在其他步骤中使用的资源，除非在管道定义中指定的两个步骤之间存在直接关系。在为步骤定义处理器或估算器时，您可以传递这些角色。有关如何在这些定义中包含这些角色的示例，请参阅SageMaker Python SDK 文档.

使用管道进行服务控制策略

服务控制策略 (SCP) 是一种组织策略，可用于管理组织中的权限。SCP 为您组织中的所有账户提供对最大可用权限的集中控制。通过使用 SageMaker 组织内的管道，您可以确保数据科学家管理您的管道执行，而无需与Amazon控制台。

如果您将 VPC 与 SCP 一起使用限制对 Amazon S3 的访问，则需要采取措施允许管道访问其他 Amazon S3 资源。

允许 SageMaker 使用管道访问 VPC 外的 Amazon S3`JsonGet`函数，更新组织的 SCP 以确保角色使用 SageMaker 管道可以访问 Amazon S3。要执行此操作，请为正在使用的角色创建例外。 SageMaker 使用委托人标签和条件键通过管道执行角色执行管道执行器。

允许 SageMaker 用于访问您的 VPC 之外的 Amazon S3 的管道

1. 按照中的步骤为管道执行角色创建唯一标签标记 IAM 用户和角色.
2. 使用在 SCP 中授予例外`Aws:PrincipalTag IAM`您创建的标签的条件密钥。有关更多信息，请参阅 。创建、更新和删除服务控制策略.

管道参数

您可以使用参数将变量引入管道定义中。您可以引用在整个管道定义中定义的参数。参数具有默认值，您可以通过在启动管道执行时指定参数值来覆盖该值。默认值必须是与参数类型匹配的实例。 步骤定义中使用的所有参数都必须在管道定义中定义。亚马逊 SageMaker 模型构建管道支持以下参数类型：

- `ParameterString`— 表示字符串参数。
- `ParameterInteger`— 表示整数参数。
- `ParameterFloat`— 表示浮点参数。
- `ParameterBoolean`— 表示布尔 Python 类型。

参数采用以下格式：

```
<parameter> = <parameter_type>(
    name="<parameter_name>",
    default_value=<default_value>
)
```

以下示例演示一个示例参数实现。

```
from sagemaker.workflow.parameters import (
    ParameterInteger,
    ParameterString,
    ParameterFloat,
    ParameterBoolean
)

processing_instance_count = ParameterInteger(
    name="ProcessingInstanceCount",
    default_value=1
)
```

如以下示例所示，您可以在创建管道时传递该参数。

```
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[
        processing_instance_count
    ],
    steps=[step_process]
)
```

您还可以将与默认值不同的参数值传递到管道执行，如以下示例所示。

```
execution = pipeline.start(
    parameters=dict(
        ProcessingInstanceType="ml.c5.xlarge",
        ModelApprovalStatus="Approved"
    )
)
```

你可以使用 SageMaker Python 开发工具包的功能 `sagemaker.workflow.functions.Join`. 有关参数的更多信息，请参阅 SageMaker 管道参数.

管道步骤

SageMaker 管道由步骤组成。这些步骤定义了管道执行的操作以及使用属性的步骤之间的关系。

主题

- 步骤类型 (p. 2450)
- 步骤属性 (p. 2463)
- 步进并行度 (p. 2463)
- 步骤之间的数据依赖 (p. 2463)
- 步骤间的自定义依赖 (p. 2464)
- 在步骤中使用自定义映像 (p. 2465)

步骤类型

下面介绍了每个步骤类型的要求，并提供了该步骤的示例实施。这些不是功能性的实现，因为它们不提供所需的资源和投入。有关实现这些步骤的教程，请参阅创建和管理 SageMaker 管道 (p. 2478).

亚马逊 SageMaker 模型构建管道支持以下步骤类型：

- 处理 (p. 2450)
- 培训 (p. 2452)
- 优化 (p. 2452)
- `CreateModel` (p. 2453)
- `RegisterModel` (p. 2453)
- 转换 (p. 2454)
- Condition (p. 2455)
- `Callback` (p. 2455)
- Lambda (p. 2457)
- `ClarifyCheck` (p. 2458)
- `QualityCheck` (p. 2460)
- EMR (p. 2461)
- Fail (p. 2462)

处理步骤

您可以使用处理步骤创建用于数据处理的处理作业。有关处理作业的更多信息，请参阅处理数据和评估模型.

处理步骤需要处理器、定义处理代码的 Python 脚本、用于处理的输出以及作业参数。以下示例显示了如何创建`ProcessingStep`定义。有关处理步骤要求的更多信息，请参阅SageMaker.workflow .steps.处理步骤文档中）。

```
from sagemaker.sklearn.processing import SKLearnProcessor

sklearn_processor = SKLearnProcessor(framework_version='0.20.0',
                                     role=<role>,
                                     instance_type='ml.m5.xlarge',
                                     instance_count=1)
```

```
from sagemaker.processing import ProcessingInput, ProcessingOutput
from sagemaker.workflow.steps import ProcessingStep

step_process = ProcessingStep(
    name="AbaloneProcess",
    processor=sklearn_processor,
    inputs=[
      ProcessingInput(source=<input_data>, destination="/opt/ml/processing/input"),
    ],
    outputs=[
        ProcessingOutput(output_name="train", source="/opt/ml/processing/train"),
        ProcessingOutput(output_name="validation", source="/opt/ml/processing/validation"),
        ProcessingOutput(output_name="test", source="/opt/ml/processing/test")
    ],
    code="abalone/preprocessing.py"
)
```

传递运行时参数

您可以使用将运行时参数传递给处理步骤get_run_args的方法亚马逊 SageMaker Python 开发工具包. 这允许你除此之外使用处理器SKLearnProcessor之外的压缩算法（例如PySparkProcessor和SparkJarProcessor.

以下示例显示如何从中传入运行时参数： PySpark 处理器到ProcessingStep.

```
from sagemaker.spark.processing import PySparkProcessor

pyspark_processor = PySparkProcessor(framework_version='2.4',
                                     role=<role>,
                                     instance_type='ml.m5.xlarge',
                                     instance_count=1)
```

```
from sagemaker.processing import ProcessingInput, ProcessingOutput

run_args = pyspark_processor.get_run_args(
    "preprocess.py",
    inputs=[
      ProcessingInput(source=<input_data>, destination="/opt/ml/processing/input"),
    ],
    outputs=[
        ProcessingOutput(output_name="train", source="/opt/ml/processing/train"),
        ProcessingOutput(output_name="validation", source="/opt/ml/processing/validation"),
        ProcessingOutput(output_name="test", source="/opt/ml/processing/test")
    ],
    arguments=None
)
```

```
from sagemaker.workflow.steps import ProcessingStep

step_process = ProcessingStep(
    name="AbaloneProcess",
    processor=pyspark_processor,
    inputs=run_args.inputs,
```

```
    outputs=run_args.outputs,
    job_arguments=run_args.arguments,
    code=run_args.code
)
```

训练步骤

您可以使用训练步骤创建训练作业来训练模型。有关训练作业的更多信息，请参阅使用 Amazon SageMaker 训练模型.

训练步骤需要估算器以及训练和验证数据输入。以下示例显示了如何创建`TrainingStep`定义。有关培训练步骤要求的更多信息，请参阅SageMaker.workflow .steps.训练步骤文档中）。

```
from sagemaker.inputs import TrainingInput
from sagemaker.workflow.steps import TrainingStep

step_train = TrainingStep(
    name="TrainAbaloneModel",
    estimator=xgb_train,
    inputs={
        "train": TrainingInput(
            s3_data=step_process.properties.ProcessingOutputConfig.Outputs[
                "train"
            ].S3Output.S3Uri,
            content_type="text/csv"
        ),
        "validation": TrainingInput(
            s3_data=step_process.properties.ProcessingOutputConfig.Outputs[
                "validation"
            ].S3Output.S3Uri,
            content_type="text/csv"
        )
    }
)
```

优化步骤

您可以使用调整步骤创建超参数调整作业，也称为超参数优化 (HPO)。一个超参数调整作业运行多个训练作业，每个作业都生成一个模型版本。有关超参数优化的更多信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939).

调整作业与 SageMaker 为管道进行试验，将培训工作作作为试验创造。有关更多信息，请参阅 实验集成 (p. 2474)。

调整步骤需要HyperparameterTuner以及培训投入。您可以通过指定`warm_start_config`的参数`HyperparameterTuner`. 有关超参数调整和热启动的详细信息，请参阅运行热启动超参数优化作业 (p. 1959).

您将get_top_model_s3_uri的方法SageMaker.workflow .steps.TuningStep类来从性能最佳的模型版本之一获取模型工件。对于展示如何使用调整步骤的笔记本 SageMaker 管道，请参阅sagemaker-管道-调整步骤 .ipynb.

> **Important**
>
> 亚马逊引入了调整步骤 SageMaker Python SDK v2.48.0 和亚马逊 SageMaker 工作室 v3.8.0。在使用调整步骤之前，必须更新 Studio，否则管道 DAG 不显示。要更新 Studio，请参阅关闭并更新 SageMaker 工作室 (p. 129).

以下示例显示了如何创建`TuningStep`定义。

```
from sagemaker.tuner import HyperparameterTuner
```

```
from sagemaker.inputs import TrainingInput
from sagemaker.workflow.steps import TuningStep

step_tuning = TuningStep(
    name = "HPTuning",
    tuner = HyperparameterTuner(...),
    inputs = TrainingInput(s3_data=input_path)
)
```

获取最佳模型版本

以下示例说明如何使用`get_top_model_s3_uri`方法。最多可以根据以下条件排名排名前 50 的表演版本超参数调整 Jobbject. 这些区域有：`top_k`参数是版本的索引，其中`top_k=0`是性能最佳的版本`top_k=49`是性能最差的版本。

```
best_model = Model(
    image_uri=image_uri,
    model_data=step_tuning.get_top_model_s3_uri(
        top_k=0,
        s3_bucket=sagemaker_session.default_bucket()
    ),
    ...
)
```

有关调整步骤要求的更多信息，请参阅SageMaker.workflow .steps.TuningStep文档中）。

CreateModel 步骤

你使用`CreateModel`创建一个步骤 SageMaker 模型。有关 SageMaker 模型，请参阅使用 Amazon SageMaker 训练模型.

创建模型步骤需要模型工件和有关 SageMaker 创建模型时需要使用的实例类型。以下示例显示了如何创建`CreateModel`步骤定义。有关`CreateModel`步骤要求，请参阅SageMaker.workflow .steps.创建模型步骤文档中）。

```
from sagemaker.workflow.steps import CreateModelStep

step_create_model = CreateModelStep(
    name="AbaloneCreateModel",
    model=best_model,
    inputs=inputs
)
```

注册Model Step

你使用`RegisterModel`步骤注册SageMaker.Model。模型或者SageMaker.Pipeline。Pipeline 模型使用 Amazon SageMaker 模型注册表。一个`PipelineModel`表示推理管道，这是由处理推理请求的容器的线性序列组成的模型。

有关如何注册模型的更多信息，请参阅使用模型注册表注册并部署模型 (p. 2252). 有关`RegisterModel`步骤要求，请参阅SageMaker.workflow .step_Collections。注册模型文档中）。

以下示例显示了如何创建`RegisterModel`注册 a 的步骤`PipelineModel`.

```
import time
from sagemaker.sklearn import SKLearnModel
from sagemaker.xgboost import XGBoostModel

code_location = 's3://{0}/{1}/code'.format(bucket_name, prefix)
```

```
sklearn_model =
 SKLearnModel(model_data=processing_step.properties.ProcessingOutputConfig.Outputs['model'].S3Output.S3
 entry_point='inference.py',
 source_dir='sklearn_source_dir/',
 code_location=code_location,
 framework_version='0.20.0',
 role=role,
 sagemaker_session=sagemaker_session,
 py_version='py3')

xgboost_model =
 XGBoostModel(model_data=training_step.properties.ModelArtifacts.S3ModelArtifacts,
 entry_point='inference.py',
 source_dir='xgboost_source_dir/',
 code_location=code_location,
 framework_version='0.90-2',
 py_version='py3',
 sagemaker_session=sagemaker_session,
 role=role)

from sagemaker.workflow.step_collections import RegisterModel
from sagemaker import PipelineModel
pipeline_model =
 PipelineModel(models=[sklearn_model,xgboost_model],role=role,sagemaker_session=sagemaker_session)

step_register = RegisterModel(
 name="AbaloneRegisterModel",
 model=pipeline_model,
 content_types=["application/json"],
 response_types=["application/json"],
 inference_instances=["ml.t2.medium", "ml.m5.xlarge"],
 transform_instances=["ml.m5.xlarge"],
 model_package_group_name='sipgroup',
)
```

如果model未提供，则注册模型步骤需要估计器，如以下示例所示。

```
from sagemaker.workflow.step_collections import RegisterModel

step_register = RegisterModel(
    name="AbaloneRegisterModel",
    estimator=xgb_train,
    model_data=step_train.properties.ModelArtifacts.S3ModelArtifacts,
    content_types=["text/csv"],
    response_types=["text/csv"],
    inference_instances=["ml.t2.medium", "ml.m5.xlarge"],
    transform_instances=["ml.m5.xlarge"],
    model_package_group_name=model_package_group_name,
    approval_status=model_approval_status,
    model_metrics=model_metrics
)
```

转换步骤

您可以使用转换步骤进行批处理转换，以便对整个数据集运行推理。有关批量转换的更多信息，请参阅运行使用推理管道进行 Batch 转换 (p. 2137).

转换步骤需要转换器和运行批处理转换的数据。以下示例显示了如何创建Transform步骤定义。有关Transform步骤要求，请参阅SageMaker.workflow .steps.TransStep文档中）。

```
from sagemaker.inputs import TransformInput
```

```
from sagemaker.workflow.steps import TransformStep

step_transform = TransformStep(
    name="AbaloneTransform",
    transformer=transformer,
    inputs=TransformInput(data=batch_data)
)
```

条件步骤

您可以使用条件步骤来评估步骤属性的条件，以评估下一步应在管道中执行哪些操作。

条件步骤需要条件列表，如果条件的计算结果为，则要运行的步骤列表`true`，以及条件计算结果为时要运行的步骤列表`false`. 以下示例显示了如何创建`Condition`步骤定义。有关`Condition`步骤要求，请参阅SageMaker.workflow。条件 _Step. 条件步骤文档中）。

限制

- SageMaker Pipeline 不支持使用嵌套条件步骤。您不能通过条件步骤作为另一个条件步骤的输入。
- 条件步骤不能在两个分支中使用相同的步骤。如果您在两个分支中都需要相同的步骤功能，请复制该步骤并为其指定不同的名称。

```
from sagemaker.workflow.conditions import ConditionLessThanOrEqualTo
from sagemaker.workflow.condition_step import (
    ConditionStep,
    JsonGet
)

cond_lte = ConditionLessThanOrEqualTo(
    left=JsonGet(
        step_name=step_eval.name,
        property_file=evaluation_report,
        json_path="regression_metrics.mse.value"
    ),
    right=6.0
)

step_cond = ConditionStep(
    name="AbaloneMSECond",
    conditions=[cond_lte],
    if_steps=[step_register, step_create_model, step_transform],
    else_steps=[]
)
```

回调步骤

你使用`Callback`步骤纳入其他流程和Amazon亚马逊不直接提供的服务纳入工作流程 SageMaker 模型构建管道。当您时`Callback`步骤运行时，将发生以下过程：

- SageMaker Pipeline 会将消息发送到客户指定的 Amazon Simple Queue Service (Amazon SQS) 队列。邮件包含 SageMaker Pipeline-生成的令牌和客户提供的输入参数列表。发送消息后， SageMaker 管道等待客户的响应。
- 客户从 Amazon SQS 队列检索消息并启动其自定义流程。
- 流程完成后，客户会调用以下 API 之一并提交 SageMaker Pipeline — 生成的令牌：
  - 发送 Pipeline 执行步骤成功，以及输出参数列表
  - 发送 Pipeline 执行步骤失败，以及失败原因
- API 调用导致 SageMaker 管道要么继续管道进程，要么使该流程失败。

有关Callback步骤要求，请参阅sagemaker.workflow .callback_Step. 回调步骤文档中）。有关完整的解决方案，请参阅延长 SageMaker 使用回调步骤包含自定义步骤的管道.

> Important
>
> CallbackAmazon 中引入了步骤 SageMaker Python SDK v2.45.0 和亚马逊 SageMaker 工作室 v3.6.2。您必须先更新 Studio 才能使用Callback步骤或管道 DAG 不显示。要更新 Studio，请参阅关闭并更新 SageMaker 工作室 (p. 129).

下面的示例演示了上述过程的实现。

```
from sagemaker.workflow.callback_step import CallbackStep

step_callback = CallbackStep(
    name="MyCallbackStep",
    sqs_queue_url="https://sqs.us-east-2.amazonaws.com/012345678901/MyCallbackQueue",
    inputs={...},
    outputs=[...]
)

callback_handler_code = '
    import boto3
    import json

    def handler(event, context):
        sagemaker_client=boto3.client("sagemaker")

        for record in event["Records"]:
            payload=json.loads(record["body"])
            token=payload["token"]

            # Custom processing

            # Call SageMaker to complete the step
            sagemaker_client.send_pipeline_execution_step_success(
                CallbackToken=token,
                OutputParameters={...}
            )
'
```

停止行为

管道过程不会停止Callback步骤正在运行。

当你打电话StopPipelineExecution在正在运行的管道进程中Callback步骤， SageMaker 管道向指定的 SQS 队列发送额外的 Amazon SQS 消息。SQS 消息的正文包含状态字段，设置为Stopping. 以下显示了示例 SQS 消息正文。

```
{
  "token": "26vcYbeWsZ",
  "pipelineExecutionArn": "arn:aws:sagemaker:us-east-2:012345678901:pipeline/callback-
pipeline/execution/7pinimwddh3a",
  "arguments": {
    "number": 5,
    "stringArg": "some-arg",
    "inputData": "s3://sagemaker-us-west-2-012345678901/abalone/abalone-dataset.csv"
  },
  "status": "Stopping"
}
```

您应该向 Amazon SQS 消息使用者添加逻辑，以便在收到消息后采取任何必要的操作（例如，资源清理），然后致电SendPipelineExecutionStepSuccess要么SendPipelineExecutionStepFailure.

只有当 SageMaker 管道会收到其中一个调用，它会停止管道进程吗？

Lambda 步骤

您可以使用 Lambda 步骤来运行Amazon Lambdafunction. 您可以运行现有的 Lambda 函数，或者 SageMaker 可以创建和运行新的 Lambda 函数。对于显示如何在中使用 Lambda 步骤的笔记本 SageMaker 管道，请参阅sagemaker-管道-lambda Step.ipynb.

> **Important**
>
> 亚马逊推出了 Lambda 步骤 SageMaker Python SDK v2.51.0 和亚马逊 SageMaker 工作室 v3.9.1。在使用 Lambda 步骤之前，必须更新 Studio，否则管道 DAG 不显示。要更新 Studio，请参阅关闭并更新 SageMaker 工作室 (p. 129).

SageMaker 提供sagemaker.lambda_helper.lambda类来创建、更新、调用和删除 Lambda 函数。`Lambda`具有以下签名。

```
Lambda(
    function_arn,       # Only required argument to invoke an existing Lambda function

    # The following arguments are required to create a Lambda function:
    function_name,
    execution_role_arn,
    zipped_code_dir,    # Specify either zipped_code_dir and s3_bucket, OR script
    s3_bucket,          # S3 bucket where zipped_code_dir is uploaded
    script,             # Path of Lambda function script
    handler,            # Lambda handler specified as "lambda_script.lambda_handler"
    timeout,            # Maximum time the Lambda function can run before the lambda step
 fails
    ...
)
```

这些区域有：sagemaker.workflow .lambda_step.lambdaStep班级有`lambda_func`参数类型`Lambda`. 要调用现有的 Lambda 函数，唯一的要求是向以下条件提供函数的 Amazon Resource Name (ARN)：`function_arn`. 如果您没有为提供值`function_arn`，您必须指定`handler`以及以下操作之一：

- `zipped_code_dir`— 压缩的 Lambda 函数的路径

  `s3_bucket`— 其中 Amazon S3 存储桶`zipped_code_dir`是要上传
- `script`— Lambda 函数脚本文件的路径

以下示例显示了如何创建`Lambda`调用现有 Lambda 函数的步骤定义。

```
from sagemaker.workflow.lambda_step import LambdaStep
from sagemaker.lambda_helper import Lambda

step_lambda = LambdaStep(
    name="ProcessingLambda",
    lambda_func=Lambda(
        function_arn="arn:aws:lambda:us-west-2:012345678910:function:split-dataset-lambda"
    ),
    inputs={
        s3_bucket = s3_bucket,
        data_file = data_file
    },
    outputs=[
        "train_file", "test_file"
    ]
)
```

以下示例显示了如何创建Lambda使用 Lambda 函数脚本创建和调用 Lambda 函数的步骤定义。

```
from sagemaker.workflow.lambda_step import LambdaStep
from sagemaker.lambda_helper import Lambda

step_lambda = LambdaStep(
    name="ProcessingLambda",
    lambda_func=Lambda(
      function_name="split-dataset-lambda",
      execution_role_arn=execution_role_arn,
      script="lambda_script.py",
      handler="lambda_script.lambda_handler",
      ...
    ),
    inputs={
        s3_bucket = s3_bucket,
        data_file = data_file
    },
    outputs=[
        "train_file", "test_file"
    ]
)
```

输入和输出

如果您的Lambda函数有输入或输出，这些也必须在Lambda步骤。

在定义Lambda步骤，inputs必须是键/值对的字典。的每个值inputs字典必须是原始类型（字符串、整数或浮点型）。不支持嵌套对象。如果保持未定义，则inputs值默认为None.

这些区域有：outputs值必须是键列表。这些键指的是在输出中定义的字典Lambdafunction. LIKEinputs，这些键必须是原始类型，并且不支持嵌套对象。

超时和停止行为

这些区域有：Lambda班级有timeout参数，指定 Lambda 函数可以运行的最长时间。默认值为 120 秒，最大值为 10 分钟。如果达到超时时 Lambda 函数正在运行，则 Lambda 步骤将失败；但是，Lambda 函数将继续运行。

在 Lambda 步骤运行期间无法停止管道进程，因为 Lambda 步骤调用的 Lambda 函数无法停止。如果您尝试在 Lambda 函数运行期间停止该进程，则管道将等待 Lambda 函数完成或直到达到超时（以先发生者为准），然后停止。如果 Lambda 函数完成，则管道进程状态为Stopped. 如果达到超时，则管道进程状态为Failed.

CLARIFY 检查步骤

您可以使用ClarifyCheck步骤对照以前的基线进行基线漂移检查，以便进行偏差分析和模型可解释性。通过这样做，你可以生成和注册基准对于漂移检查RegisterModel (p. 2453)以及在模型构建阶段通过管道训练的模型。亚马逊可以使用这些漂移检查基准 SageMaker 模型监视器适用于模型终端节点，这样您就不需要执行基线分别提供建议。这些区域有：ClarifyCheckstep 还可以从模型注册表中提取用于漂移检查的基线。这些区域有：ClarifyCheckstep 利用了亚马逊 SageMaker 要澄清预建容器，提供一系列模型监控功能，包括针对给定基准的约束验证。有关更多信息，请参阅 。入门 SageMaker 澄清容器.

配置 ClarifyCheck 步

您可以配置ClarifyCheck步骤来每次在管道中使用以下检查类型时只执行其中一种检查类型。

- 检查数据偏差
- 检查模型偏差

- 模型可解释性检查

要实现此目的，需要将clarify_check_config具有以下检查类型值之一的参数：

- DataBiasCheckConfig
- ModelBiasCheckConfig
- ModelExplainabilityCheckConfig

这些区域有：ClarifyCheckstep 启动一个处理作业，该作业运行 SageMaker 澄清预构建的容器并需要专用支票和处理作业的配置.ClarifyCheckConfig和CheckJobConfig是这些配置的帮助函数，它们与 SageMaker 澄清处理作业计算，以检查模型偏差、数据偏差或模型可解释性。有关更多信息，请参阅 运行 SageMaker 澄清处理工作以进行偏见分析和可解释性 (p. 2055)。

控制漂移检查的步骤行为

这些区域有：ClarifyCheckstep 需要以下两个布尔标志来控制其行为：

- skip_check：此参数表示是否跳过与之前基线的漂移检查。如果设置为False，必须可用已配置的检查类型的先前基准。
- register_new_baseline：此参数表示是否可以通过 step 属性访问新计算的基线BaselineUsedForDriftCheckConstraints. 如果设置为False，配置的检查类型的先前基准也必须可用。可通过BaselineUsedForDriftCheckConstraints财产。

有关更多信息，请参阅 基线计算、漂移检测和生命周期 ClarifyCheck 和 QualityCheck 亚马逊中的步骤 SageMaker 模型构建管道 (p. 2469)。

使用基准

您也可以指定model_package_group_name找到现有的基线和ClarifyCheck一步拉DriftCheckBaselines在模型包组中最新批准的模型包上。或者，您可以通过supplied_baseline_constraints参数。如果指定了model_package_group_name和supplied_baseline_constraints，ClarifyCheck步骤使用由supplied_baseline_constraints参数。

有关使用的更多信息ClarifyCheck步骤要求，请参阅SageMaker.workflow .steps.Clarify 检查步骤中的亚马逊 SageMaker 适用于 Python 的 SageMaker SDK. 对于 Amazon SageMaker 演示如何使用的工作室笔记本ClarifyCheck单步进入 SageMaker 管道，请参阅sagemaker-Pipeline 模型监视器-澄清-steps.ipynb.

Example 创建**ClarifyCheck**数据偏差检查步骤

```
from sagemaker.workflow.check_job_config import CheckJobConfig
from sagemaker.workflow.clarify_check_step import DataBiasCheckConfig, ClarifyCheckStep
from sagemaker.workflow.execution_variables import ExecutionVariables

check_job_config = CheckJobConfig(
    role=role,
    instance_count=1,
    instance_type="ml.c5.xlarge",
    volume_size_in_gb=120,
    sagemaker_session=sagemaker_session,
)

data_bias_data_config = DataConfig(

 s3_data_input_path=step_process.properties.ProcessingOutputConfig.Outputs["train"].S3Output.S3Uri,
    s3_output_path=Join(on='/', values=['s3:/', your_bucket, base_job_prefix,
 ExecutionVariables.PIPELINE_EXECUTION_ID, 'databiascheckstep']),
    label=0,
```

```
    dataset_type="text/csv",
    s3_analysis_config_output_path=data_bias_analysis_cfg_output_path,
)

data_bias_config = BiasConfig(
    label_values_or_threshold=[15.0], facet_name=[8], facet_values_or_threshold=[[0.5]]
)

data_bias_check_config = DataBiasCheckConfig(
    data_config=data_bias_data_config,
    data_bias_config=data_bias_config,
)h

data_bias_check_step = ClarifyCheckStep(
    name="DataBiasCheckStep",
    clarify_check_config=data_bias_check_config,
    check_job_config=check_job_config,
    skip_check=False,
    register_new_baseline=False
   supplied_baseline_constraints="s3://sagemaker-us-west-2-111122223333/baseline/
analysis.json",
    model_package_group_name="MyModelPackageGroup"
)
```

质量检查步骤

您可以使用QualityCheck步骤进行基准建议以及对管道中的数据质量或模型质量的先前基线进行偏移检查。通过这样做，你可以生成和注册基准对于漂移检查注册Model Step (p. 2453)与模型构建阶段通过管道训练的模型一起步进行培训。Model Monitor 可以使用这些基线对模型端点进行漂移检查，这样您就不需要单独执行基线建议。这些区域有：QualityCheckstep 还可以从模型注册表中提取用于漂移检查的基线。这些区域有：QualityCheckstep 利用了亚马逊 SageMaker Model Monitor 预构建容器，其具有一系列模型监控功能，包括约束建议、统计数据生成和基准约束验证。有关更多信息，请参阅 亚马逊 SageMaker 模型监控器预构建容器 (p. 2192)。

配置 QualityCheck 步

您可以配置QualityCheck步骤来每次在管道中使用以下检查类型时只执行其中一种检查类型。

- 数据质量检查
- 检查模型质量

要实现此目的，需要将quality_check_config具有以下检查类型值之一的参数：

- DataQualityCheckConfig
- ModelQualityCheckConfig

这些区域有：QualityCheckstep 启动一个处理作业，该作业运行 Model Monitor 预构建的容器，并且需要为检查和处理作业进行专用配置。这些区域有：QualityCheckConfig和CheckJobConfig是这些配置的帮助函数，它们与模型监视器为模型质量或数据质量监控创建基线的方式保持一致。有关模型监视器基准建议的更多信息，请参阅创建基准 (p. 2169)和创建模型质量基准 (p. 2174).

控制漂移检查的步骤行为

这些区域有：QualityCheckstep 需要以下两个布尔标志来控制其行为：

- skip_check：此参数表示是否跳过与之前基线的漂移检查。如果设置为False，必须可用已配置的检查类型的先前基准。
- register_new_baseline：此参数表示是否可以通过步骤属性访问新计算的基线BaselineUsedForDriftCheckConstraints和BaselineUsedForDriftCheckStatistics.

如果设置为`False`，配置的检查类型的先前基准也必须可用。可通过访问这些`BaselineUsedForDriftCheckConstraints`和`BaselineUsedForDriftCheckStatistics`属性。

有关更多信息，请参阅 基线计算、漂移检测和生命周期 ClarifyCheck 和 QualityCheck 亚马逊中的步骤 SageMaker 模型构建管道 (p. 2469)。

使用基准

您可以直接通过`supplied_baseline_statistics`和`supplied_baseline_constraints`参数，或者你可以简单地指定`model_package_group_name`和`QualityCheck`一步拉`DriftCheckBaselines`在模型包组中最新批准的模型包上。当你指定`model_package_group_name`，`supplied_baseline_constraints`，和`supplied_baseline_statistics`，`QualityCheck`step 使用指定的基线`supplied_baseline_constraints`和`supplied_baseline_statistics`在的支票类型`QualityCheck`你正在运行的步骤。

有关使用的更多信息`QualityCheck`步骤要求，请参阅sagemaker.workflow .steps.质量检查步骤中的亚马逊 SageMaker 适用于 Python 的 SageMaker SDK. 对于 Amazon SageMaker 演示如何使用的工作室笔记本`QualityCheck`单步进入 SageMaker 管道，请参阅sagemaker-Pipeline 模型监视器-澄清-steps.ipynb.

Example 创建**`QualityCheck`**数据质量检查步骤

```
from sagemaker.workflow.check_job_config import CheckJobConfig
from sagemaker.workflow.quality_check_step import DataQualityCheckConfig, QualityCheckStep
from sagemaker.workflow.execution_variables import ExecutionVariables

check_job_config = CheckJobConfig(
    role=role,
    instance_count=1,
    instance_type="ml.c5.xlarge",
    volume_size_in_gb=120,
    sagemaker_session=sagemaker_session,
)

data_quality_check_config = DataQualityCheckConfig(

 baseline_dataset=step_process.properties.ProcessingOutputConfig.Outputs["train"].S3Output.S3Uri,
    dataset_format=DatasetFormat.csv(header=False, output_columns_position="START"),
    output_s3_uri=Join(on='/', values=['s3:/', your_bucket, base_job_prefix,
 ExecutionVariables.PIPELINE_EXECUTION_ID, 'dataqualitycheckstep'])
)

data_quality_check_step = QualityCheckStep(
    name="DataQualityCheckStep",
    skip_check=False,
    register_new_baseline=False,
    quality_check_config=data_quality_check_config,
    check_job_config=check_job_config,
    supplied_baseline_statistics="s3://sagemaker-us-west-2-555555555555/baseline/
statistics.json",
    supplied_baseline_constraints="s3://sagemaker-us-west-2-555555555555/baseline/
constraints.json",
    model_package_group_name="MyModelPackageGroup"
)
```

Amazon EMR 步骤

您可以使用 Amazon SageMaker 模型构建管道Amazon EMR步骤来处理EMR 步骤到运行中的 Amazon EMR 集群。有关更多信息，请参阅 。开始使用 Amazon EMR.

亚马逊 EMR 步骤需要`EMRStepConfig`具有将由 Amazon EMR 集群使用的 JAR 的 Amazon S3 位置以及要传递的任何参数以及 Amazon EMR 群集 ID。

Note

- Amazon EMR 步骤要求传递给管道的角色具有额外的权限。您应该附上Amazon托管策略：AmazonSageMakerPipelinesIntegrations转到管道角色，或者确保该角色包含该策略中的权限。
- 不支持 Amazon EMR（EKS）。
- 只能在处于以下某种状态的集群上运行 EMR 步骤：`STARTING`、`BOOTSTRAPPING`、`RUNNING`，或者`WAITING`.
- 您最多可拥有 256 个 EMR 步骤`PENDING`在 EMR 群集上；提交超出该限制的 EMR 步骤会导致管道执行失败。您可以考虑使用管道步骤的重试策略 (p. 2467).

Example 创建 Amazon EMR 步骤定义，在 EMR 群集上启动新作业。

```
from sagemaker.workflow.emr_step import EMRStep, EMRStepConfig

emr_config = EMRStepConfig(
    jar="s3://path/to/jar/MyJar.jar", # required, S3 path to jar
    args=["--verbose", "--force"], # optional list of arguments to pass to the jar
    main_class="com.my.Main1", # optional main class, this can be omitted if jar above has
 a manifest
    properties=[ # optional list of Java properties that are set when the step runs
    {
        "key": "mapred.tasktracker.map.tasks.maximum",
        "value": "2"
    },
    {
        "key": "mapreduce.map.sort.spill.percent",
        "value": "0.90"
   },
   {
       "key": "mapreduce.tasktracker.reduce.tasks.maximum",
       "value": "5"
    }
  ]
)

step_emr = EMRStep (
    name="EMRSampleStep", # required
    cluster_id="j-1YONHTCP3YZKC", # required
    step_config=emr_config, # required
    display_name="My EMR Step",
    description="Pipeline step to execute EMR job"
)
```

失败步骤

你使用`FailStep`阻止亚马逊 SageMaker 在未达到所需条件或状态时执行模型构建管道，并将该管道的执行标记为失败。这些区域有：`FailStep`还允许您输入自定义错误消息，指示管道执行失败的原因。

Note

当您时`FailStep`而其他管道步骤同时执行，则在所有并发步骤完成之前，管道才会终止。

使用的限制`FailStep`

- 您无法添加`FailStep`到`DependsOn`其他步骤列表。有关更多信息，请参阅 步骤间的自定义依赖 (p. 2464)。
- 其他步骤无法引用`FailStep`. 这是总是是管道执行的最后一步。

- 您无法重试以结尾的管道执行FailStep.

您可以创建FailStep ErrorMessage采用静态文本字符串的形式。或者，您也可以使用管道参数一个加入操作或其他步骤属性以创建更具信息性的错误消息。

Example

以下示例代码片段使用FailStep用一个ErrorMessage配置了管道参数和Joinoperation.

```
from sagemaker.workflow.fail_step import FailStep
from sagemaker.workflow.functions import Join
from sagemaker.workflow.parameters import ParameterInteger

mse_threshold_param = ParameterInteger(name="MseThreshold", default_value=5)
step_fail = FailStep(
    name="AbaloneMSEFail",
    error_message=Join(
        on=" ", values=["Execution failed due to MSE >", mse_threshold_param]
    ),
)
```

步骤属性

这些区域有：properties属性用于在管道中的步骤之间添加数据依赖关系。然后将使用这些数据依赖关系 SageMaker 用于根据管道定义构造 DAG 的管道。这些属性可以作为占位符值引用，并在运行时解析。

这些区域有：properties属性 SageMaker Pipeline 步骤与返回的对象匹配Describe呼叫相应的 SageMaker 任务类型。对于每种作业类型，Describe调用将返回以下响应对象：

- ProcessingStep–描述处理作业
- TrainingStep–DescribeTrainingJob
- TransformStep–描述转换Job

步进并行度

当步骤不依赖于任何其他步骤时，它会在管道执行后立即运行。但是，parallel 行执行太多的管道步骤可以快速耗尽可用资源。使用控制管道执行的并发步骤数ParallelismConfiguration.

下面的示例使用ParallelismConfiguration将并发步长限制设置为 5。

```
pipeline.create(
    parallelism_config=ParallelismConfiguration(5),
)
```

步骤之间的数据依赖

您可以通过指定步骤之间的数据关系来定义 DAG 的结构。要在步骤之间创建数据依赖关系，请将一个步骤的属性作为输入传递给管道中的另一个步骤。在提供输入的步骤完成运行之后，接收输入的步骤才会开始。

数据依赖关系使用 JsonPath 采用以下格式的符号。此格式遍历 JSON 属性文件，这意味着你可以追加尽可能多的<property>实例以访问文件中所需的嵌套属性。有关 JsonPath 符号，请参阅JSONPath 存储库.

```
<step_name>.properties.<property>.<property>
```

以下内容介绍了如何使用指定 Amazon S3 存储桶。ProcessingOutputConfig处理步骤的属性。

```
step_process.properties.ProcessingOutputConfig.Outputs["train_data"].S3Output.S3Uri
```

要创建数据依赖关系，请按如下方式将存储桶传递给训练步骤。

```
step_train = TrainingStep(
    name="CensusTrain",
    estimator=sklearn_train,
    inputs=TrainingInput(
        s3_data=step_process.properties.ProcessingOutputConfig.Outputs[
            "train_data"].S3Output.S3Uri
    )
)
```

步骤间的自定义依赖

指定数据依赖关系时， SageMaker Pipeline 提供了步骤之间的数据连接。或者，一个步骤可以访问上一步中的数据，而无需直接使用 SageMaker 管道。在这种情况下，您可以创建一个自定义依赖项，告诉 SageMaker 在另一个步骤完成运行之前，管道不要开始一个步骤。您可以通过指定步骤来创建自定义依赖项`DependsOn`属性。

例如，下面定义了一个步骤`C`只有在两个步骤之后才开始`A`单步进入`B`完成跑步。

```
{
  'Steps': [
    {'Name':'A', ...},
    {'Name':'B', ...},
    {'Name':'C', 'DependsOn': ['A', 'B']}
  ]
}
```

如果依赖项会创建循环依赖关系，则 SageMaker Pipeline 会抛出验证异常。

以下示例创建了一个训练步骤，该步骤在处理步骤完成运行后开始。

```
processing_step = ProcessingStep(...)
training_step = TrainingStep(...)

training_step.add_depends_on([processing_step])
```

以下示例创建了一个训练步骤，该步骤在两个不同的处理步骤完成运行之前才开始。

```
processing_step_1 = ProcessingStep(...)
processing_step_2 = ProcessingStep(...)

training_step = TrainingStep(...)

training_step.add_depends_on([processing_step_1, processing_step_2])
```

下面提供了创建自定义依赖关系的另一种方法。

```
training_step.add_depends_on([processing_step_1])
training_step.add_depends_on([processing_step_2])
```

以下示例创建了一个训练步骤，该步骤接收来自一个处理步骤的输入，并等待其他处理步骤完成运行。

```
processing_step_1 = ProcessingStep(...)
```

```
processing_step_2 = ProcessingStep(...)

training_step = TrainingStep(
    ...,
    inputs=TrainingInput(
        s3_data=processing_step_1.properties.ProcessingOutputConfig.Outputs[
            "train_data"
        ].S3Output.S3Uri
    )

training_step.add_depends_on([processing_step_2])
```

以下示例说明如何检索某个步骤的自定义依赖项的字符串列表。

```
custom_dependencies = training_step.depends_on
```

在步骤中使用自定义映像

您可以使用任何可用的 SageMaker深度学习容器映像当您在管道中创建步骤时。

您也可以使用创建步骤 SageMaker Amazon S3 应用程序。一个 SageMaker Amazon S3 应用程序是一个 tar.gz 捆绑包，其中包含一个或多个可以在该捆绑包中运行的 Python 脚本。有关应用程序包捆绑的更多信息，请参阅直接从模型项目部署.

您还可以将自己的容器与管道步骤一起使用。因为你无法在亚马逊内创建图片 SageMaker Studio，您必须先使用其他方法创建图片，然后才能与亚马逊一起使用 SageMaker 模型构建管道。

要在为管道创建步骤时使用自己的容器，请在估计器定义中包含图像 URI。有关将您的容器与 SageMaker 搭配使用的更多信息，请参阅在 SageMaker 中使用 Docker 容器.

属性文件和JsonGet

您可以使用属性文件存储来自处理步骤输出的信息。在分析处理步骤的结果以决定如何执行条件步骤时，这特别有用。这些区域有：JsonGet函数处理属性文件并使您能够使用 JsonPath 表示法来查询属性 JSON 文件。有关 JsonPath 符号，请参阅JSONPath 存储库.

要存储属性文件以供以后使用，必须先创建PropertyFile实例采用以下格式。这些区域有：path参数是要将属性文件保存到的 JSON 文件的名称。output_name必须与output_name的ProcessingOutput您在处理步骤中定义的。这使属性文件能够捕获ProcessingOutput在步骤中。

```
from sagemaker.workflow.properties import PropertyFile

<property_file_instance> = PropertyFile(
    name="<property_file_name>",
    output_name="<processingoutput_output_name>",
    path="<path_to_json_file>"
)
```

当您创建ProcessingStep实例，请添加property_files参数来列出亚马逊的所有参数文件 SageMaker 模型构建管道服务必须索引。此操作将保存属性文件以备后用。

```
property_files=[<property_file_instance>]
```

要在条件步骤中使用属性文件，请添加property_file转到您传递给条件步骤的条件，如以下示例所示，以便使用json_path参数。

```
cond_lte = ConditionLessThanOrEqualTo(
```

```
    left=JsonGet(
        step_name=step_eval.name,
        property_file=<property_file_instance>,
        json_path="mse"
    ),
    right=6.0
)
```

缓存管道步骤

当你使用步骤签名缓存时，之前 SageMaker 管道执行一个步骤，它试图找到以前用相同参数调用的步骤的执行情况。SageMaker Pipeline 会检查呼叫签名是否相同。Pipeline 不会检查参数所指向的数据或代码是否已更改。如果 Pipeline 找到之前的执行，它会创建缓存命中。然后，Pipeline 在执行过程中传播缓存命中的值，而不是重新计算步骤。

步骤缓存只考虑成功执行，因此它永远不会重复使用失败的执行。如果在超时期限内存在多个成功执行，Pipeline 将使用结果来执行最近一次成功执行。如果在超时期内没有成功执行匹配，则 Pipeline 不会重复使用任何步骤。 如果执行者找到仍在进行的上一步执行的缓存命中，则两个步骤都将继续执行并更新缓存（如果成功）。

你必须选择加入分步缓存，否则默认情况下它处于关闭状态。启用步骤缓存时，还必须定义超时。此超时定义了以前的执行时间可以考虑重复使用的时间。

步骤缓存仅限于单个管道的范围，因此您无法重复使用其他管道中的步骤。即使另一个管道中存在步骤签名匹配，也不会重复使用该步骤。

步骤缓存可用于以下步骤类型：

- 处理 (p. 2450)
- 培训 (p. 2452)
- 优化 (p. 2452)
- 转换 (p. 2454)
- `ClarifyCheck` (p. 2458)
- `QualityCheck` (p. 2460)
- EMR (p. 2461)

启用步骤缓存

要启用步骤缓存，必须添加`CacheConfig`属性到步骤定义中。

`CacheConfig`属性在管道定义文件中使用以下格式。

```
{
    "CacheConfig": {
        "Enabled": false,
        "ExpireAfter": "<time>"
    }
}
```

这些区域有：`Enabled`字段可能为 true 或 false。`ExpireAfter`是定义超时时间段的字符串。任何 ISO 8601 持续时间字符串都是有效的`ExpireAfter`值。这些区域有：`ExpireAfter`持续时间可包含年、月、周、日、小时和分钟值。每个值都由一个数字组成，后跟一个字母表示它所用的持续时间单位。 例如：

- “30d” = 三十天
- “5y” = 五年

- “t16m” = 16 分钟
- “30dt5h” = 30 天零 5 个小时。

以下示例说明如何使用 Amazon 对训练步骤启用缓存。 SageMaker Python 开发工具包。

```
from sagemaker.workflow.steps import CacheConfig

cache_config = CacheConfig(enable_caching=True, expire_after="PT1H")

step_train = TrainingStep(
    name="TrainAbaloneModel",
    estimator=xgb_train,
    inputs=inputs,
    cache_config=cache_config
)
```

管道步骤的重试策略

重试策略可以帮助你自动重试 SageMaker 错误发生后，管道将步骤操作。任何管道步骤都可能遇到异常，出于各种原因出现异常。在某些情况下，重试可以解决这些问题。对于管道步骤的重试策略，您可以选择是否重试特定管道步骤。

重试策略仅支持以下管道步骤：

- 处理步骤 (p. 2450)
- 训练步骤 (p. 2452)
- 优化步骤 (p. 2452)

  Note

  这些区域有： SageMaker 超参数调整作业已在内部执行重试，因此它不会重试`SageMaker.JOB_INTERNAL_ERROR`例外类型，即使配置了重试策略也是如此。如果你真的想重试，你可以自己编程重试策略使用 SageMaker API。
- CreateModel 步骤 (p. 2453)
- 注册Model Step (p. 2453)
- 转换步骤 (p. 2454)

重试策略支持的异常类型

管道步骤的重试策略支持以下异常类型：

- `Step.SERVICE_FAULT`：当调用下游服务时发生内部服务器错误或暂时错误时，会发生这些例外情况。SageMaker 管道会自动重试这种类型的错误。使用重试策略，您可以覆盖此例外类型的默认重试操作。
- `Step.THROTTLING`：调用下游服务时可能会发生限制异常。 SageMaker 管道会自动重试这种类型的错误。使用重试策略，您可以覆盖此例外类型的默认重试操作。
- `SageMaker.JOB_INTERNAL_ERROR`：这些例外发生在 SageMaker 工作回报`InternalServerError`. 在这种情况下，开始新工作可能会解决暂时性问题。
- `SageMaker.CAPACITY_ERROR`: 该 SageMaker 工作可能会遇到 Amazon EC2`InsufficientCapacityErrors`，这导致 SageMaker 工作失败。你可以通过开始一个新的重试 SageMaker 工作以避免这个问题。
- `SageMaker.RESOURCE_LIMIT`：运行时，您可以超过资源限制配额 SageMaker 任务。你可以等然后重试运行 SageMaker 在短时间之后工作，看看资源是否被释放。

重试策略的 JSON 模式

对于 Pipeline 的重试策略有以下 JSON 架构：

```
"RetryPolicy": {
   "ExceptionType": [String]
   "IntervalSeconds": Integer
   "BackoffRate": Double
   "MaxAttempts": Integer
   "ExpireAfterMin": Integer
}
```

- `ExceptionType`：此字段需要以字符串数组格式提供以下异常类型。
  - `Step.SERVICE_FAULT`
  - `Step.THROTTLING`
  - `SageMaker.JOB_INTERNAL_ERROR`
  - `SageMaker.CAPACITY_ERROR`
  - `SageMaker.RESOURCE_LIMIT`
- `IntervalSeconds`（可选）：第一次重试尝试之前的秒数（默认值为 1）。`IntervalSeconds`最大值为 43200 秒（12 小时）。
- `BackoffRate`（可选）：每次尝试期间重试时间间隔增长的倍数 (默认值为 2.0)。
- `MaxAttempts`（可选）：一个正整数，表示重试的最大次数 (默认值为 5 次)。如果错误再次发生的次数超过`MaxAttempts`指定，停止重试并恢复正常错误处理。值为 0 时，指定永不重试错误。`MaxAttempts`最大值为 20。
- `ExpireAfterMin`（可选）：一个正整数，表示重试的最大时间跨度。如果在之后再次出现错误`ExpireAfterMin`执行步骤计数的分钟数，停止重试并恢复正常错误处理。值为 0 时，指定永不重试错误。`ExpireAfterMin` 最大值为 14,400 分钟（10 天）。

  Note

  只有其中一个`MaxAttempts`要么`ExpireAfterMin`可以给出，但不能两者兼而有；如果两者都是不已指定，`MaxAttempts`成为默认设置。如果在一个策略中识别了两个属性，则重试策略将生成验证错误。

配置重试策略

以下是具有重试策略的训练步骤的示例。

```
{
    "Steps": [
        {
            "Name": "MyTrainingStep",
            "Type": "Training",
            "RetryPolicies": [
                {
                    "ExceptionType": [
                        "SageMaker.JOB_INTERNAL_ERROR",
                        "SageMaker.CAPACITY_ERROR"
                    ],
                    "IntervalSeconds": 1,
                    "BackoffRate": 2,
                    "MaxAttempts": 5
                }
            ]
        }
    ]
}
```

以下示例说明了如何构建TrainingStep在 SDK for Python (Boto3) 中使用重试策略。

```
from sagemaker.workflow.retry import (
    StepRetryPolicy,
    StepExceptionTypeEnum,
    SageMakerJobExceptionTypeEnum,
    SageMakerJobStepRetryPolicy
)

step_train = TrainingStep(
    name="MyTrainingStep",
    xxx,
    retry_policies=[
        // override the default
        StepRetryPolicy(
            exception_types=[
                StepExceptionTypeEnum.SERVICE_FAULT,
                StepExceptionTypeEnum.THROTTLING
            ],
            expire_after_min=5,
            interval_seconds=10,
            backoff_rate=2.0
        ),
        // retry when resource limit quota gets exceeded
        SageMakerJobStepRetryPolicy(
            exception_types=[SageMakerJobExceptionTypeEnum.RESOURCE_LIMIT]
            expire_after_min=120,
            interval_seconds=60,
            backoff_rate=2.0
        ),
        // retry when job failed due to transient error or EC2 ICE.
        SageMakerJobStepRetryPolicy(
            failure_reason_types=[
                SageMakerJobExceptionTypeEnum.JOB_INTERNAL_ERROR,
                SageMakerJobExceptionTypeEnum.JOB_CAPACITY_ERROR,
            ]
            max_attempts=10,
            interval_seconds=30,
            backoff_rate=2.0
        )
    ]
)
```

基线计算、漂移检测和生命周期 ClarifyCheck 和 QualityCheck 亚马逊中的步骤 SageMaker 模型构建管道

以下主题讨论了亚马逊基准和模型版本如何演变 SageMaker 使用时对构建管道进行模型ClarifyCheck (p. 2458)和QualityCheck (p. 2460)步骤。

对于ClarifyCheckstep，基线是位于步骤属性中的单个文件，后缀为后缀constraints. 对于QualityCheckstep，基线是驻留在步骤属性中的两个文件的组合：一个带有后缀statistics另一个带有后缀constraints. 在以下主题中，我们将以前缀讨论这些属性，该前缀描述了它们的使用方式，从而影响这两个管道步骤中的基准行为和生命周期。例如，ClarifyCheckstep 始终在CalculatedBaselineConstraints财产和QualityCheckstep在CalculatedBaselineConstraints和CalculatedBaselineStatistics属性。

基线计算和注册 ClarifyCheck 和 QualityCheck 步骤

无论您的ClarifyCheck和QualityCheck步骤始终根据通过底层处理作业运行的步骤输入来计算新的基线。这些新计算的基线可以通过带前缀的属性访问CalculatedBaseline. 您可以将这些属性记录

为ModelMetrics中的模型包RegisterModel (p. 2453)步骤。该模型包可以使用 5 种不同的基线注册。您可以为每种支票类型注册一个：数据偏差、模型偏差和模型可解释性ClarifyCheck步骤和模型质量，以及运行QualityCheck步骤。这些区域有：register_new_baseline参数指示了在属性中设置的带有前缀的值BaselineUsedForDriftCheck在一步运行之后。

下表列出了潜在用例的表格，显示了可以为ClarifyCheck和QualityCheck步骤：

选择此配置时可能会考虑的使用案例	**skip_check** / **register_new_ba**	步骤会进行漂移检查吗？	Step 属性的价值**CalculatedBase**	Step 属性的价值**BaselineUsedForDriftChe**
你正在定期进行再训练，并启用检查以获得新的模型版本，但是你想结转以前的基线作为DriftCheckBaselines在新模型版本的模型注册表中。	False/ False	漂移检查将根据现有基线运行	通过运行步骤计算的新基线	模型注册表中最新批准的模型或作为步骤参数提供的基线
你正在定期进行再训练，并启用检查以获得新的模型版本，但是你想要刷新*DriftCheckBaselines*在模型注册表中使用新计算的基线适用于您的新模型版本。	False/ True	漂移检查将根据现有基线运行	通过运行步骤计算的新基线	通过运行步骤（属性值）新计算的基线CalculatedBaseline）
您正在启动管道以重新训练新模型版本，因为亚马逊检测到了违规行为 SageMaker 对于特定类型的检查的终端上的模型监视器，并且您想对照之前的基线跳过此类检查，但将之前的基线作为*DriftCheckBaselines*在模型注册表中适用于您的新模型版本。	True/ False	无偏差检查	通过运行计算的新基准	来自模型注册表中最新批准的模型或作为步骤参数提供的基线
这发生在以下情况下： • 您正在开始管道的初始运行、构建第一个模型版本并生成初始基线。 • 您正在启动管道以重新训练新模型版本，因为模型监视器在终端上检测	True/ True	无偏差检查	通过运行步骤计算的新基线	通过运行步骤（属性值）新计算的基线CalculatedBaseline）

选择此配置时可能会考虑的使用案例	**skip_check** / **register_new_ba**	步骤会进行漂移检查吗？	Step 属性的价值**CalculatedBase**	Step 属性的价值**BaselineUsedForDriftChe**
到特定类型检查的违规。如果要跳过对照之前的基线检查并刷新*DriftCheckBaselines*使用新计算的基线直接在模型注册表中。				

在RegisterModel步骤，您可以注册BaselineUsedForDriftCheck属性为DriftCheckBaselines. 然后，模型监视器可以使用这些基准文件进行模型和数据质量检查。此外，这些基线也可以用于 ClarifyCheckStep 和QualityCheck步骤将新训练的模型与在模型注册表中注册的现有模型进行比较，以便 future 的管道运行。

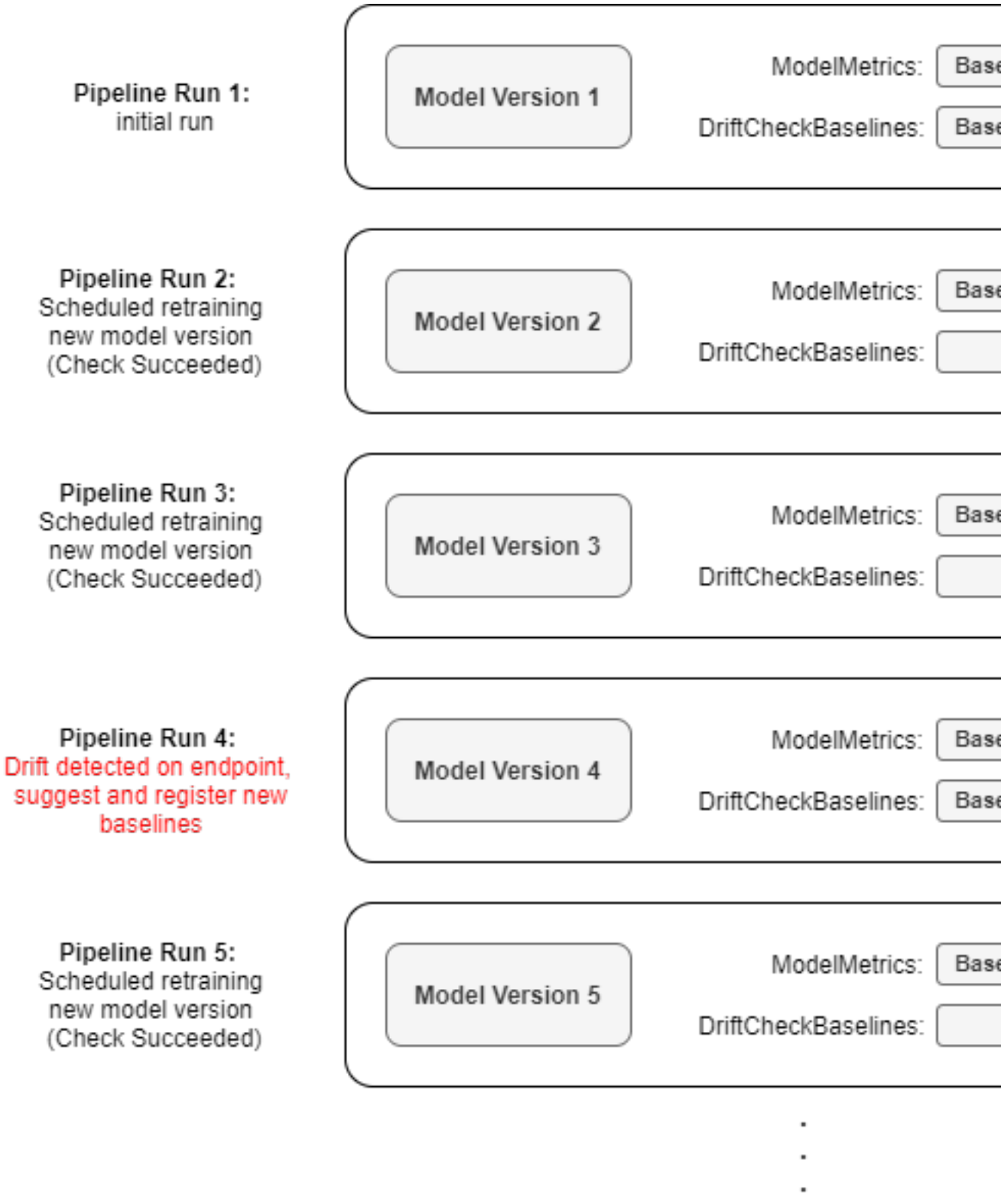
根据以前的基线进行漂移检测 SageMaker 管道

在这种情况下QualityCheckstep，当您启动定期再训练以获得新模型版本的管道时，如果数据质量和数据偏差存在，则可能不想运行训练步骤??? (p. 2172)根据您之前批准的模型版本的基线。如果模型质量、模型偏差或模型可解释性违反了之前批准的模型版本的注册基线，则可能也不想注册新训练的模型版本ClarifyCheck步骤。在这些情况下，您可以通过设置skip_check设置为的相应检查步骤的属性False，导致ClarifyCheck和QualityCheck如果检测到违反之前的基准，则步骤失败。然后管道进程不会继续，因此不会注册偏离基线的模型。ClarifyCheck和QualityCheck步骤能够得到DriftCheckBaselines要比较的给定模型包组的最新批准的模型版本。以前的基准也可以直接通过supplied_baseline_constraints（除了supplied_baseline_statistics如果是QualityCheckstep），并且始终优先于从模型包组中提取的任何基线。

基线和模型版本生命周期和演进 SageMaker 管道

通过设置register_new_baseline您的ClarifyCheck和QualityCheck单步进入False，您之前的基线可以通过 step 属性前缀访问BaselineUsedForDriftCheck. 然后，您可以将这些基准注册为DriftCheckBaselines在新模型版本中RegisterModel步骤。在模型注册表中批准此新模型版本后，DriftCheckBaseline在此模型版本中可用于ClarifyCheck和QualityCheck下一个管道流程中的步骤。如果要为 future 模型版本刷新某种检查类型的基线，则可以设置register_new_baseline到True以便带有前缀的属性BaselineUsedForDriftCheck成为新计算的基线。通过这些方式，您可以为 future 训练的模型保留首选基准，或者在需要时刷新基准以进行漂移检查，从而在模型训练迭代中管理基线演变和生命周期。

下图阐明了 model-version-centric 基线演变和生命周期的视图。


Pipeline Run 1:
initial run
Model Version 1
ModelMetrics:
DriftCheckBaselines:
Pipeline Run 2:
Scheduled retraining
new model version
(Check Succeeded)
Model Version 2
ModelMetrics:
DriftCheckBaselines:
Pipeline Run 3:
Scheduled retraining
new model version
(Check Succeeded)
Model Version 3
ModelMetrics:
DriftCheckBaselines:
Pipeline Run 4:
Drift detected on endpoint,
suggest and register new
baselines
Model Version 4
ModelMetrics:
DriftCheckBaselines:
Pipeline Run 5:
Scheduled retraining
new model version
(Check Succeeded)
Model Version 5
ModelMetrics:
DriftCheckBaselines:

亚马逊 EventBridge 集成

你可以安排你的亚马逊 SageMaker 模型构建管道处决使用Amazon EventBridge. 亚马逊 SageMaker 在中支持模型构建管道作为目标Amazon EventBridge. 这允许您根据事件总线中的任何事件启动模型构建管道的执行。借助 EventBridge，您可以自动执行管道，并自动响应培训作业或终端状态更改等事件。事件包括正在上传到 Amazon S3 存储桶的新文件、亚马逊状态的变化 SageMaker 由于漂移造成的端点，以及Amazon Simple Notification Service(SNS) 主题。

以下 SageMaker 可以自动启动管道操作：

- `StartPipelineExecution`

有关日程安排的更多信息 SageMaker 工作，请参阅使用 实现自动化 SageMaker 使用 Amazon EventBridge。

使用 Amazon EventBridge 安排管道

通过亚马逊启动管道执行 CloudWatch 活动，你必须创建 EventBridge规则. 在为事件创建规则时，您可以指定要在以下时间执行的目标操作。 EventBridge 接收与规则匹配的事件。当事件符合规则时， EventBridge 将事件发送到指定的目标并启动规则中定义的操作。

以下教程演示如何使用计划管道执行。 EventBridge 使用 EventBridge 控制台或Amazon CLI.

先决条件

- 一个角色 EventBridge 可以假设`SageMaker::StartPipelineExecution`允许。如果您从 EventBridge 控制台创建规则，则可以自动创建此角色；否则，您需要自己创建此角色。 有关创建 SageMaker 角色，请参阅SageMaker 角色.
- Amazon SageMaker 待安排的管道。要创建亚马逊 SageMaker 管道，请参阅定义管道.

创建 EventBridge 使用 EventBridge 控制台的规则

以下过程演示如何创建 EventBridge 规则使用 EventBridge 控制台。

1. 导航到EventBridge 控制台.
2. SelectRule在左侧。
3. Select `Create Rule`.
4. 为您的规则输入名称和描述。
5. 选择您希望如何启动此规则。您的规则有以下选择：
   - 事件模式：当发生与模式匹配的事件时，您的规则将启动。您可以选择与特定类型的事件匹配的预定义模式，也可以创建自定义模式。如果选择了预定义的模式，则可以编辑模式以对其进行自定义。有关事件模式的更多信息，请参阅中的事件模式 CloudWatch 事件.
   - Schedule：您的规则将按照指定的时间表定期启动。您可以使用固定费率计划，定期启动指定的分钟、小时或周数。您还可以使用Cron 表达式以创建更精细的时间表，例如 “每个月的第一个星期一上午 8 点”。自定义事件总线或合作伙伴事件总线不支持时间表。
6. 选择你想要的活动总线。
7. 选择当事件与您的事件模式匹配或启动时间表时要调用的目标。每个规则最多可以添加 5 个目标。Select`SageMaker Pipeline`在目标下拉列表中。
8. 从管道下拉列表中选择要启动的管道。
9. 使用名称和值对添加要传递给管道执行的参数。参数值可以是静态的或动态的。有关 Amazon 的更多信息 SageMaker 管道参数，请参阅AWS::Events::Rule SageMakerPipeline 参数.

- 每次启动管道时，静态值都会传递给管道执行。例如，如果`{"Name": "Instance_type", "Value": "ml.4xlarge"}`在参数列表中指定，然后它作为参数传入`StartPipelineExecutionRequest`每次 EventBridge 启动管道。
- 动态值是使用 JSON 路径指定的。 EventBridge 从事件负载中解析值，然后将其传递给管道执行。例如：*`$.detail.param.value`*。

10. 选择用于此规则的角色。您可以使用现有角色或创建一个新角色。
11. （可选）添加标签。
12. Select`Create`以最终确定你的规则。

您的规则现已生效，可以启动管道执行。

创建 EventBridge 规则使用Amazon CLI

以下过程演示如何创建 EventBridge 规则使用Amazon CLI.

1. 创建要启动的规则。创建时 EventBridge 规则使用Amazon CLI，您有两种选择来确定规则的启动方式、事件模式和时间表。

   - 事件模式：当发生与模式匹配的事件时，您的规则将启动。您可以选择与特定类型的事件匹配的预定义模式，也可以创建自定义模式。如果选择了预定义的模式，则可以编辑模式以对其进行自定义。 您可以使用以下命令创建具有事件模式的规则：

     ```
     aws events put-rule --name <RULE_NAME> ----event-pattern <YOUR_EVENT_PATTERN> --
     description <RULE_DESCRIPTION> --role-arn <ROLE_TO_EXECUTE_PIPELINE> --tags <TAGS>
     ```

   - Schedule：您的规则将按照指定的时间表定期启动。您可以使用固定费率计划，定期启动指定的分钟、小时或周数。您还可以使用 cron 表达式来创建更精细的时间表，例如“每月第一个星期一上午 8 点”。自定义事件总线或合作伙伴事件总线不支持时间表。您可以使用以下命令创建具有计划的规则：

     ```
     aws events put-rule --name <RULE_NAME> --schedule-expression <YOUR_CRON_EXPRESSION>
      --description <RULE_DESCRIPTION> --role-arn <ROLE_TO_EXECUTE_PIPELINE> --tags <TAGS>
     ```

2. 添加目标，以便在事件与您的事件模式匹配或时间表启动时调用。每个规则最多可以添加 5 个目标。对于每个目标，您必须指定：

   - ARN：管道的资源 ARN。
   - 角色 ARN：角色的 ARN EventBridge 应该假设执行管道。
   - 参数： 亚马逊 SageMaker 要传递的管道参数。

3. 运行以下命令以传递 Amazon SageMaker 使用管道作为规则的目标放置目标：

   ```
   aws events put-targets --rule <RULE_NAME> --event-bus-name <EVENT_BUS_NAME>
    --targets "[{\"Id\": <ID>, \"Arn\": <RESOURCE_ARN>, \"RoleArn\": <ROLE_ARN>,
    \"SageMakerPipelineParameter\": { \"SageMakerParameterList\": [{\"Name\": <NAME>,
    \"Value\": <VALUE>}]} }]"]
   ```

亚马逊 SageMaker 实验集成

亚马逊 SageMaker 模型构建管道与亚马逊紧密集成 SageMaker 实验。默认情况下，何时 SageMaker Pipeline 创建并执行管道，如下 SageMaker 如果实验实体不存在，则会创建实验实体：

- 管道的实验
- 每次执行管道的试用
- 为每个项添加到试用中的试用组件 SageMaker 在管道执行步骤中创建的作业

您可以在多个管道执行中比较模型训练准确度等指标，就像在多个试验中比较这些指标一样，就像在 SageMaker 模型训练实验。

以下示例显示了管道在中的类亚马逊 SageMaker Python 开发工具包.

```
Pipeline(
    name="MyPipeline",
    parameters=[...],
    pipeline_experiment_config=PipelineExperimentConfig(
      ExecutionVariables.PIPELINE_NAME,
      ExecutionVariables.PIPELINE_EXECUTION_ID
    ),
    steps=[...]
)
```

如果你不想为管道创建实验和试验，请设置`pipeline_experiment_config`到`None`.

> **Note**
>
> 亚马逊引入了实验集成 SageMaker Python 软件开发工具包 v2.41.0。

以下命名规则基于您为`ExperimentName`和`TrialName`的参数`pipeline_experiment_config`：

- 如果您没有指定`ExperimentName`，管道`name`用于实验名称。

  如果你确实指定`ExperimentName`，它用于实验名称。如果存在具有该名称的实验，则管道创建的试验将添加到现有实验中。如果不存在具有该名称的实验，则会创建一个新的实验。
- 如果您没有指定`TrialName`，管道执行 ID 用于试用名称。

  如果你确实指定`TrialName`，它用于试用名称。如果存在具有该名称的试验，则管道创建的试验组件将添加到现有试验中。如果不存在具有该名称的试用版，则会创建一个新的试用版。

> **Note**
>
> 当创建实体的管道被删除时，实验实体不会被删除。您可以使用 SageMaker 实验 API 来删除实体。有关更多信息，请参阅 清理 Amazon SageMaker 实验资源 (p. 1776)。

有关如何查看的信息，请参阅 SageMaker 实验与管道关联的实体，请参阅查看创建者的实验实体 SageMaker 管道 (p. 2499). 有关 SageMaker 请参阅实验使用亚马逊管理 Machine Learning SageMaker 实验 (p. 1758).

以下各节展示了先前规则的示例以及它们在管道定义文件中的表示方式。有关管道定义文件的更多信息，请参阅。SageMaker 管道概述 (p. 2446).

主题

- 默认行为 (p. 2475)
- 禁用实验集成 (p. 2476)
- 指定自定义实验名称 (p. 2476)
- 指定自定义试用名称 (p. 2477)

默认行为

创建管道

这些区域有：`pipeline_experiment_config`被省略。`ExperimentName`默认为管道`name`.`TrialName`默认为执行ID。

```
pipeline_name = f"MyPipeline"
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[...],
    steps=[step_train]
)
```

管道定义文件

```
{
  "Version": "2020-12-01",
  "Parameters": [
    {
      "Name": "InputDataSource"
    },
    {
      "Name": "InstanceCount",
      "Type": "Integer",
      "DefaultValue": 1
    }
  ],
  "PipelineExperimentConfig": {
    "ExperimentName": {"Get": "Execution.PipelineName"},
    "TrialName": {"Get": "Execution.PipelineExecutionId"}
  },
  "Steps": [...]
}
```

禁用实验集成

创建管道

`pipeline_experiment_config` 设置为 `None`。

```
pipeline_name = f"MyPipeline"
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[...],
    pipeline_experiment_config=None,
    steps=[step_train]
)
```

管道定义文件

这与前面的默认示例相同，没有`PipelineExperimentConfig`.

指定自定义实验名称

使用自定义实验名称。试用名称设置为执行 ID，与默认行为一样。

创建管道

```
pipeline_name = f"MyPipeline"
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[...],
    pipeline_experiment_config=PipelineExperimentConfig(
      "CustomExperimentName",
      ExecutionVariables.PIPELINE_EXECUTION_ID
```

```
    ),
    steps=[step_train]
)
```

管道定义文件

```
{
  ...,
  "PipelineExperimentConfig": {
    "ExperimentName": "CustomExperimentName",
    "TrialName": {"Get": "Execution.PipelineExecutionId"}
  },
  "Steps": [...]
}
```

指定自定义试用名称

使用自定义试用名称并附加执行 ID。与默认行为一样，实验名称设置为管道名称。

创建管道

```
pipeline_name = f"MyPipeline"
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[...],
    pipeline_experiment_config=PipelineExperimentConfig(
      ExecutionVariables.PIPELINE_NAME,
      Join(on="-", values=["CustomTrialName", ExecutionVariables.PIPELINE_EXECUTION_ID])
    ),
    steps=[step_train]
)
```

管道定义文件

```
{
  ...,
  "PipelineExperimentConfig": {
    "ExperimentName": {"Get": "Execution.PipelineName"},
    "TrialName": {
      "On": "-",
      "Values": [
         "CustomTrialName",
         {"Get": "Execution.PipelineExecutionId"}
       ]
    }
  },
  "Steps": [...]
}
```

Amazon 故障排除 SageMaker 模型构建管道

使用亚马逊时 SageMaker 建模构建管道，您可能会因各种原因而遇到问题。本主题提供有关常见错误以及如何解决这些错误的信息。

管道定义问题

管道定义的格式可能不正确。这可能会导致执行失败或者你的工作不准确。在创建管道或执行时可能会发现这些错误。如果你的定义没有验证， SageMaker Pipeline 返回一条错误消息，标识 JSON 文件格式错误的字符。要解决此问题，请查看使用 SageMaker 为了准确起见 Python SDK。

您只能在管道定义中包含一次步骤。因此，步骤不能作为条件步骤的一部分存在和同一管道中的管道。

检查管道日志

您可以使用以下命令查看步骤的状态：

```
execution.list_steps()
```

每个步骤包含以下信息：

- 管道启动的实体的 ARN，例如 SageMaker 作业 ARN、模型 ARN 或模型包 ARN。
- 失败原因包括对步骤失败的简要说明。
- 如果步骤是条件步骤，则包括条件是评估为 true 还是 false。
- 如果执行重复使用之前的任务执行，则`CacheHit`列出了源代码执行。

您还可以查看错误消息并在 Amazon 中查看错误消息和日志。 SageMaker Studio 界面。有关如何在 Studio 中查看日志的信息，请参阅查看管道执行 (p. 2497).

缺失权限

创建管道执行的角色以及在管道执行中创建每个作业的步骤都需要正确的权限。如果没有这些权限，您可能无法提交管道执行或运行 SageMaker 按预期工作。要确保正确设置权限，请参阅IAM 访问管理 (p. 2448).

Job 执行错误

执行步骤时可能会遇到问题，因为脚本中定义了你的功能的问题 SageMaker 个作业。每个作业都有一组 CloudWatch 日志。 要从 Studio 查看这些日志，请参阅查看管道执行 (p. 2497). 有关使用的信息 CloudWatch 使用 SageMaker 的日志，请参阅使用 Amazon CloudWatch 记录 Amazon SageMaker 事件 (p. 2820).

属性文件错误

使用管道错误地实现属性文件时，可能会遇到问题。要确保属性文件的实施按预期工作，请参阅属性文件和`JsonGet` (p. 2465).

创建和管理 SageMaker 管道

你可以使用亚马逊 SageMaker 为创建管道模型构建管道 end-to-end 管理和部署的工作流 SageMaker 个作业。 SageMaker 附带管道 SageMaker Python SDK 集成，因此您可以使用基于 Python 的界面构建管道的每一步。

部署管道后，您可以查看管道的定向非循环图 (DAG) 并使用亚马逊管理您的执行 SageMaker Studio。 使用 SageMaker Studio，您可以获取有关当前和历史管道的信息、比较执行情况、查看执行情况的 DAG、获取元数据信息等。要了解如何从 SageMaker Studio 查看管道，请参阅查看、跟踪和执行 SageMaker 中的管道 SageMaker 工作室 (p. 2493).

主题

- 定义管道 (p. 2478)
- 运行管道 (p. 2490)
- 查看、跟踪和执行 SageMaker 中的管道 SageMaker 工作室 (p. 2493)

定义管道

与亚马逊协调您的工作流程 SageMaker 模型构建管道，您需要以 JSON 管道定义的形式生成定向非循环图 (DAG)。 下图是您在本教程中创建的管道 DAG 的表示形式：

您可以使用生成 JSON 管道定义 SageMaker Python 开发工具包。 以下教程展示了如何为管道生成管道定义，该管道可以解决回归问题，以根据鲍鱼的物理测量结果来确定鲍鱼的年龄。有关包含本教程中可以运行的内容的 Jupyter 笔记本，请参阅与亚马逊协调工作 SageMaker 模型构建管道.

主题

- 先决条件 (p. 2479)
- 创建管道 (p. 2480)

先决条件

要运行以下教程，您必须执行以下操作：

- 按照中的概述设置笔记本实例创建笔记本实例. 这使您的角色能够在 SageMaker 中读取和写入 Amazon S3，以及在 SageMaker 中创建培训、批量转换和处理作业的权限。
- 授予笔记本权限以获取和传递自己的角色，如中所示修改角色权限策略. 添加以下 JSON 片段以将此策略附加到您的角色。Replace<your-role-arn>使用 ARN 用于创建笔记本实例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole"
            ],
```

```
            "Resource": "<your-role-arn>"
        }
    ]
}
```

- 相信 SageMaker 按照中的步骤执行服务主体修改角色信任策略. 将以下语句片段添加到角色的信任关系中：

```
{
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
```

设置环境

创建新的 SageMaker 会话使用以下代码块。这将返回会话的角色 ARN。 此角色 ARN 应该是您作为先决条件设置的执行角色 ARN。

```
import boto3
import sagemaker
import sagemaker.session


region = boto3.Session().region_name
sagemaker_session = sagemaker.session.Session()
role = sagemaker.get_execution_role()
default_bucket = sagemaker_session.default_bucket()
model_package_group_name = f"AbaloneModelPackageGroupName"
```

创建管道

运行以下步骤 SageMaker notebook 实例来创建管道，包括预处理、训练、评估、条件评估和模型注册的步骤。

第 1 步：下载数据集

本笔记本使用 UCI Machine Learning 鲍鱼数据集。数据集包含以下功能：

- `length`— 鲍鱼最长的贝壳测量。
- `diameter`— 鲍鱼的直径垂直于它的长度。
- `height`— 贝壳里有肉的鲍鱼的高度。
- `whole_weight`— 整个鲍鱼的重量。
- `shucked_weight`— 从鲍鱼中取出的肉的重量。
- `viscera_weight`— 出血后鲍鱼内脏的重量。
- `shell_weight`— 肉类去除和干燥后鲍鱼壳的重量。
- `sex`— 鲍鱼的性别。“M”、“F” 或 “我” 之一，其中 “我” 是婴儿鲍鱼。
- `rings`— 鲍鱼壳中的环形数。

使用公式，鲍鱼壳中的戒指数量对于其年龄来说是一个很好的近似值`age=rings + 1.5`. 但是，获取这个数字是一项耗时的任务。你必须通过锥体切割外壳，弄脏部分，然后通过显微镜计算戒指的数量。但是，其他物理测量值更容易确定。本笔记本使用数据集使用其他物理测量值来构建可变环的预测模型。

下载数据集

1. 将数据集下载到您账户的默认 Amazon S3 存储桶中。

```
!mkdir -p data
local_path = "data/abalone-dataset.csv"

s3 = boto3.resource("s3")
s3.Bucket(f"sagemaker-servicecatalog-seedcode-{region}").download_file(
    "dataset/abalone-dataset.csv",
    local_path
)

base_uri = f"s3://{default_bucket}/abalone"
input_data_uri = sagemaker.s3.S3Uploader.upload(
    local_path=local_path,
    desired_s3_uri=base_uri,
)
print(input_data_uri)
```

2. 创建模型后，下载第二个数据集以进行批量转换。

```
local_path = "data/abalone-dataset-batch"

s3 = boto3.resource("s3")
s3.Bucket(f"sagemaker-servicecatalog-seedcode-{region}").download_file(
    "dataset/abalone-dataset-batch",
    local_path
)

base_uri = f"s3://{default_bucket}/abalone"
batch_data_uri = sagemaker.s3.S3Uploader.upload(
    local_path=local_path,
    desired_s3_uri=base_uri,
)
print(batch_data_uri)
```

第 2 步：定义管道参数

此代码块定义了管道的以下参数：

- `processing_instance_type`— 该`ml.*`处理作业的实例类型。
- `processing_instance_count`— 处理作业的实例计数。
- `training_instance_type`— 该`ml.*`培训作业的实例类型。
- `input_data`— 输入数据的 Amazon S3 位置。
- `batch_data`— 用于批处理转换的输入数据的 Amazon S3 位置。
- `model_approval_status`— 为 CI/CD 注册训练模型的批准状态。有关更多信息，请参阅 使用实现自动化 mLOP SageMaker 项目 (p. 2507)。

```
from sagemaker.workflow.parameters import (
    ParameterInteger,
    ParameterString,
)


processing_instance_count = ParameterInteger(
    name="ProcessingInstanceCount",
    default_value=1
```

```
)
processing_instance_type = ParameterString(
    name="ProcessingInstanceType",
    default_value="ml.m5.xlarge"
)
training_instance_type = ParameterString(
    name="TrainingInstanceType",
    default_value="ml.m5.xlarge"
)
model_approval_status = ParameterString(
    name="ModelApprovalStatus",
    default_value="PendingManualApproval"
)
input_data = ParameterString(
    name="InputData",
    default_value=input_data_uri,
)
batch_data = ParameterString(
    name="BatchData",
    default_value=batch_data_uri,
)
```

第 3 步：为要素工程定义处理步骤

本部分将介绍如何创建处理步骤，以便为训练准备数据集中的数据。

创建处理步骤

1. 为处理脚本创建一个目录。

   ```
   !mkdir -p abalone
   ```

2. 在中创建文件/abalone名为的目录preprocessing.py包含以下内容。此预处理脚本将传递到处理步骤，以便对输入数据执行。然后，训练步骤使用预处理的训练要素和标签来训练模型，评估步骤使用训练模型和预处理的测试要素和标签来评估模型。该脚本使用scikit-learn执行以下操作：

   - 填写丢失sex分类数据并对其进行编码，以便适合训练。
   - 缩放和标准化除外的所有数字字段rings和sex.
   - 将数据拆分为训练、测试和验证数据集。

   ```
   %%writefile abalone/preprocessing.py
   import argparse
   import os
   import requests
   import tempfile
   import numpy as np
   import pandas as pd


   from sklearn.compose import ColumnTransformer
   from sklearn.impute import SimpleImputer
   from sklearn.pipeline import Pipeline
   from sklearn.preprocessing import StandardScaler, OneHotEncoder


   # Because this is a headerless CSV file, specify the column names here.
   feature_columns_names = [
       "sex",
       "length",
       "diameter",
       "height",
   ```

```
    "whole_weight",
    "shucked_weight",
    "viscera_weight",
    "shell_weight",
]
label_column = "rings"

feature_columns_dtype = {
    "sex": str,
    "length": np.float64,
    "diameter": np.float64,
    "height": np.float64,
    "whole_weight": np.float64,
    "shucked_weight": np.float64,
    "viscera_weight": np.float64,
    "shell_weight": np.float64
}
label_column_dtype = {"rings": np.float64}


def merge_two_dicts(x, y):
    z = x.copy()
    z.update(y)
    return z


if __name__ == "__main__":
    base_dir = "/opt/ml/processing"

    df = pd.read_csv(
        f"{base_dir}/input/abalone-dataset.csv",
        header=None,
        names=feature_columns_names + [label_column],
        dtype=merge_two_dicts(feature_columns_dtype, label_column_dtype)
    )
    numeric_features = list(feature_columns_names)
    numeric_features.remove("sex")
    numeric_transformer = Pipeline(
        steps=[
            ("imputer", SimpleImputer(strategy="median")),
            ("scaler", StandardScaler())
        ]
    )

    categorical_features = ["sex"]
    categorical_transformer = Pipeline(
        steps=[
            ("imputer", SimpleImputer(strategy="constant", fill_value="missing")),
            ("onehot", OneHotEncoder(handle_unknown="ignore"))
        ]
    )

    preprocess = ColumnTransformer(
        transformers=[
            ("num", numeric_transformer, numeric_features),
            ("cat", categorical_transformer, categorical_features)
        ]
    )

    y = df.pop("rings")
    X_pre = preprocess.fit_transform(df)
    y_pre = y.to_numpy().reshape(len(y), 1)

    X = np.concatenate((y_pre, X_pre), axis=1)

    np.random.shuffle(X)
```

```
    train, validation, test = np.split(X, [int(.7*len(X)), int(.85*len(X))])


    pd.DataFrame(train).to_csv(f"{base_dir}/train/train.csv", header=False,
 index=False)
    pd.DataFrame(validation).to_csv(f"{base_dir}/validation/validation.csv",
 header=False, index=False)
    pd.DataFrame(test).to_csv(f"{base_dir}/test/test.csv", header=False, index=False)
```

3. 创建的实例SKLearnProcessor进入处理步骤。

```
from sagemaker.sklearn.processing import SKLearnProcessor


framework_version = "0.23-1"

sklearn_processor = SKLearnProcessor(
    framework_version=framework_version,
    instance_type=processing_instance_type,
    instance_count=processing_instance_count,
    base_job_name="sklearn-abalone-process",
    role=role,
)
```

4. 创建处理步骤。这个步骤是在SKLearnProcessor、输入和输出通道以及preprocessing.py你创建的脚本。这与处理器实例非常相似run中的方法 SageMaker Python 开发工具包。这些区域有：input_data传递到的参数ProcessingStep是步骤本身的输入数据。处理器实例在运行时会使用此输入数据。

   注意"train、"validation, 和"test"在处理作业的输出配置中指定的命名频道。步骤Properties例如它们可以在后续步骤中使用，并在执行时解析为它们的运行时值。

```
from sagemaker.processing import ProcessingInput, ProcessingOutput
from sagemaker.workflow.steps import ProcessingStep


step_process = ProcessingStep(
    name="AbaloneProcess",
    processor=sklearn_processor,
    inputs=[
      ProcessingInput(source=input_data, destination="/opt/ml/processing/input"),
    ],
    outputs=[
        ProcessingOutput(output_name="train", source="/opt/ml/processing/train"),
        ProcessingOutput(output_name="validation", source="/opt/ml/processing/
validation"),
        ProcessingOutput(output_name="test", source="/opt/ml/processing/test")
    ],
    code="abalone/preprocessing.py",
)
```

第 4 步：定义训练步骤

本节演示如何使用 SageMaker XGBoost 算法以便根据处理步骤的训练数据输出来训练逻辑回归模型。

定义训练步骤

1. 指定您要在训练中保存模型的模型路径。

```
model_path = f"s3://{default_bucket}/AbaloneTrain"
```

2. 为 xgBoost 算法和输入数据集配置估计器。这些区域有：`training_instance_type`传递到估算程序中。典型的训练脚本从输入通道加载数据、使用超参数配置训练、训练模型并将模型保存到`model_dir`以便稍后可以托管它。 SageMaker 以一种形式将模型上传到 Amazon S3`model.tar.gz`在训练作业结束时。

```
from sagemaker.estimator import Estimator


image_uri = sagemaker.image_uris.retrieve(
    framework="xgboost",
    region=region,
    version="1.0-1",
    py_version="py3",
    instance_type=training_instance_type,
)
xgb_train = Estimator(
    image_uri=image_uri,
    instance_type=training_instance_type,
    instance_count=1,
    output_path=model_path,
    role=role,
)
xgb_train.set_hyperparameters(
    objective="reg:linear",
    num_round=50,
    max_depth=5,
    eta=0.2,
    gamma=4,
    min_child_weight=6,
    subsample=0.7,
    silent=0
)
```

3. 创建`TrainingStep`使用估计器实例和`ProcessingStep`。 特别是，请传入`S3Uri`的`"train"`和`"validation"`输出频道到`TrainingStep`.

```
from sagemaker.inputs import TrainingInput
from sagemaker.workflow.steps import TrainingStep


step_train = TrainingStep(
    name="AbaloneTrain",
    estimator=xgb_train,
    inputs={
        "train": TrainingInput(
            s3_data=step_process.properties.ProcessingOutputConfig.Outputs[
                "train"
            ].S3Output.S3Uri,
            content_type="text/csv"
        ),
        "validation": TrainingInput(
            s3_data=step_process.properties.ProcessingOutputConfig.Outputs[
                "validation"
            ].S3Output.S3Uri,
            content_type="text/csv"
        )
    },
)
```

第 5 步：定义模型评估的处理步骤

本部分将介绍如何创建处理步骤以评估模型的准确性。此模型评估的结果用于条件步骤，以确定要采取的执行路径。

定义模型评估的处理步骤

1. 在中创建文件/abalone名为的目录evaluation.py. 此脚本在处理步骤中用于执行模型评估。它接受训练的模型和测试数据集作为输入，然后生成包含分类评估指标的 JSON 文件。

```
%%writefile abalone/evaluation.py
import json
import pathlib
import pickle
import tarfile
import joblib
import numpy as np
import pandas as pd
import xgboost


from sklearn.metrics import mean_squared_error


if __name__ == "__main__":
    model_path = f"/opt/ml/processing/model/model.tar.gz"
    with tarfile.open(model_path) as tar:
        tar.extractall(path=".")

    model = pickle.load(open("xgboost-model", "rb"))

    test_path = "/opt/ml/processing/test/test.csv"
    df = pd.read_csv(test_path, header=None)

    y_test = df.iloc[:, 0].to_numpy()
    df.drop(df.columns[0], axis=1, inplace=True)

    X_test = xgboost.DMatrix(df.values)

    predictions = model.predict(X_test)

    mse = mean_squared_error(y_test, predictions)
    std = np.std(y_test - predictions)
    report_dict = {
        "regression_metrics": {
            "mse": {
                "value": mse,
                "standard_deviation": std
            },
        },
    }

    output_dir = "/opt/ml/processing/evaluation"
    pathlib.Path(output_dir).mkdir(parents=True, exist_ok=True)

    evaluation_path = f"{output_dir}/evaluation.json"
    with open(evaluation_path, "w") as f:
        f.write(json.dumps(report_dict))
```

2. 创建实例ScriptProcessor用于创建ProcessingStep.

```
from sagemaker.processing import ScriptProcessor
```

```
script_eval = ScriptProcessor(
    image_uri=image_uri,
    command=["python3"],
    instance_type=processing_instance_type,
    instance_count=1,
    base_job_name="script-abalone-eval",
    role=role,
)
```

3. 创建ProcessingStep使用处理器实例、输入和输出通道以及evaluation.py脚本。 特别是，请传入S3ModelArtifacts属性来自step_train训练步骤，以及S3Uri的"test"的输出通道step_process处理步骤。 这与处理器实例非常相似run中的方法 SageMaker Python 开发工具包。

```
from sagemaker.workflow.properties import PropertyFile


evaluation_report = PropertyFile(
    name="EvaluationReport",
    output_name="evaluation",
    path="evaluation.json"
)
step_eval = ProcessingStep(
    name="AbaloneEval",
    processor=script_eval,
    inputs=[
        ProcessingInput(
            source=step_train.properties.ModelArtifacts.S3ModelArtifacts,
            destination="/opt/ml/processing/model"
        ),
        ProcessingInput(
            source=step_process.properties.ProcessingOutputConfig.Outputs[
                "test"
            ].S3Output.S3Uri,
            destination="/opt/ml/processing/test"
        )
    ],
    outputs=[
        ProcessingOutput(output_name="evaluation", source="/opt/ml/processing/
evaluation"),
    ],
    code="abalone/evaluation.py",
    property_files=[evaluation_report],
)
```

第 6 步：定义 CreateModelStep 适用于 Batch 转换

本节演示如何创建 SageMaker 来自训练步骤输出的模型。此模型用于对新数据集进行批量转换。此步骤将传递到条件步骤中，并且只有在条件步骤的计算结果为时才执行true.

定义 CreateModelStep 用于批量转换

1. 创建 SageMaker 模型。传递S3ModelArtifacts属性来自step_train训练步骤。

```
from sagemaker.model import Model


model = Model(
    image_uri=image_uri,
    model_data=step_train.properties.ModelArtifacts.S3ModelArtifacts,
    sagemaker_session=sagemaker_session,
    role=role,
```

```
)
```

2. 为你的定义模型输入 SageMaker 模型。

```
from sagemaker.inputs import CreateModelInput


inputs = CreateModelInput(
    instance_type="ml.m5.large",
    accelerator_type="ml.eia1.medium",
)
```

3. 创建您的`CreateModelStep`使用`CreateModelInput`和 SageMaker 您定义的模型实例。

```
from sagemaker.workflow.steps import CreateModelStep


step_create_model = CreateModelStep(
    name="AbaloneCreateModel",
    model=model,
    inputs=inputs,
)
```

步骤 7：定义 TransformStep 要执行 Batch 转换

本节演示如何创建`TransformStep`在训练模型后对数据集执行批处理转换。此步骤将传递到条件步骤中，并且只有在条件步骤的计算结果为时才执行`true`.

定义 TransformStep 执行批处理转换

1. 使用适当的计算实例类型、实例计数和所需输出 Amazon S3 存储桶 URI 创建变压器实例。传递`ModelName`属性来自`step_create_model` `CreateModel`步骤。

```
from sagemaker.transformer import Transformer


transformer = Transformer(
    model_name=step_create_model.properties.ModelName,
    instance_type="ml.m5.xlarge",
    instance_count=1,
    output_path=f"s3://{default_bucket}/AbaloneTransform"
)
```

2. 创建`TransformStep`使用你定义的变压器实例和`batch_data`管道参数。

```
from sagemaker.inputs import TransformInput
from sagemaker.workflow.steps import TransformStep


step_transform = TransformStep(
    name="AbaloneTransform",
    transformer=transformer,
    inputs=TransformInput(data=batch_data)
)
```

步骤 8：定义 RegisterModel 创建模型 Package 的步骤

本节介绍如何构建`RegisterModel`. 执行的结果`RegisterModel`在管道中是模型包。模型包是一种可重复使用的模型工件抽象，它包含推断所需的所有要素。它包含一个推理规范，该规范定义了要使用的推理图像

以及可选的模型权重位置。模型包组是模型包的集合。您可以使用ModelPackageGroup为了 SageMaker Pipeline 为每个管道执行添加新版本和模型包到组中。有关模型注册表的更多信息，请参阅使用模型注册表注册并部署模型 (p. 2252).

此步骤将传递到条件步骤中，并且只有在条件步骤的计算结果为时才执行true.

定义 RegisterModel 创建模型包

- 构建RegisterModel步骤使用您用于训练步骤的估计器实例。传递S3ModelArtifacts属性来自step_train训练步骤并指定ModelPackageGroup. SageMaker 管道创造了这个ModelPackageGroup为您而言。

```
from sagemaker.model_metrics import MetricsSource, ModelMetrics
from sagemaker.workflow.step_collections import RegisterModel


model_metrics = ModelMetrics(
    model_statistics=MetricsSource(
        s3_uri="{}/evaluation.json".format(
            step_eval.arguments["ProcessingOutputConfig"]["Outputs"][0]["S3Output"]
["S3Uri"]
        ),
        content_type="application/json"
    )
)
step_register = RegisterModel(
    name="AbaloneRegisterModel",
    estimator=xgb_train,
    model_data=step_train.properties.ModelArtifacts.S3ModelArtifacts,
    content_types=["text/csv"],
    response_types=["text/csv"],
    inference_instances=["ml.t2.medium", "ml.m5.xlarge"],
    transform_instances=["ml.m5.xlarge"],
    model_package_group_name=model_package_group_name,
    approval_status=model_approval_status,
    model_metrics=model_metrics
)
```

步骤 9：定义条件步骤以验证模型准确性

一个ConditionStep允许 SageMaker 用于根据步骤属性的条件在管道 DAG 中支持条件执行的管道。在这种情况下，只有在模型评估步骤确定的模型的准确性超过所需值时，才想注册模型包。如果精度超过所需值，则管道还会创建 SageMaker 对数据集进行建模并运行批量转换。此节演示如何定义条件步骤。

定义条件步骤以验证模型准确性

1. 定义ConditionLessThanOrEqualTo条件使用模型评估处理步骤输出中找到的精度值，step_eval. 使用您在处理步骤中索引的属性文件以及平方误差值的相应 JSONPath 获取此输出，"mse".

```
from sagemaker.workflow.conditions import ConditionLessThanOrEqualTo
from sagemaker.workflow.condition_step import ConditionStep
from sagemaker.workflow.functions import JsonGet


cond_lte = ConditionLessThanOrEqualTo(
    left=JsonGet(
        step_name=step_eval.name,
        property_file=evaluation_report,
        json_path="regression_metrics.mse.value"
    ),
    right=6.0
```

```
)
```

2. 构建ConditionStep. 传递ConditionEquals条件中，然后将模型包注册和批量转换步骤设置为条件通过后续步骤。

```
step_cond = ConditionStep(
    name="AbaloneMSECond",
    conditions=[cond_lte],
    if_steps=[step_register, step_create_model, step_transform],
    else_steps=[],
)
```

步骤 10：创建管道

既然您已经创建了所有步骤，请将它们合并到管道中。

创建管道

1. 为管道定义以下内容：name、parameters, 和steps. 名称在(account, region)对。

   Note

   步骤只能在管道的步骤列表或条件步骤的 if/else 步骤列表中出现一次。它不能出现在两者中。

```
from sagemaker.workflow.pipeline import Pipeline


pipeline_name = f"AbalonePipeline"
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[
        processing_instance_type,
        processing_instance_count,
        training_instance_type,
        model_approval_status,
        input_data,
        batch_data,
    ],
    steps=[step_process, step_train, step_eval, step_cond],
)
```

2. （可选）检查 JSON 管道定义以确保其格式正确。

```
import json

json.loads(pipeline.definition())
```

此管道定义已准备好提交给 SageMaker 了。在下一个教程中，您将此管道提交到 SageMaker 然后启动处决。

下一步: 运行管道 (p. 2490)

运行管道

使用创建管道定义后 SageMaker Python SDK，你可以将其提交到 SageMaker 开始执行。以下教程演示了如何提交管道、开始执行、检查该执行的结果以及删除管道。

主题

- 先决条件 (p. 2491)

- 第 1 步：启动管道 (p. 2491)
- 第 2 步：检查管道执行 (p. 2491)
- 第 3 步：覆盖管道执行的默认参数 (p. 2492)
- 第 4 步：停止和删除管道执行 (p. 2493)

先决条件

本教程要求以下项目：

- 一个 SageMaker 笔记本实例。
- 一个 SageMaker 管道管道定义。本教程假定使用的是通过完成创建的管道定义。定义管道 (p. 2478)教程。

第 1 步：启动管道

首先，您需要启动管道。

启动管道

1. 检查 JSON 管道定义以确保其格式良好。

```
import json

json.loads(pipeline.definition())
```

2. 将管道定义提交到 SageMaker Pipeline 服务用于在管道不存在的情况下创建管道，或者如果存在，则更新管道。 传入的角色由 SageMaker 用于创建步骤中定义的所有作业的管道。

```
pipeline.upsert(role_arn=role)
```

3. 启动管道执行。

```
execution = pipeline.start()
```

第 2 步：检查管道执行

接下来，你需要检查管道的执行情况。

检查管道执行

1. 描述管道执行状态以确保管道已成功创建并启动。

```
execution.describe()
```

2. 请等待执行完成。

```
execution.wait()
```

3. 列出执行步骤及其状态。

```
execution.list_steps()
```

您的输出应与以下内容类似：

```
[{'StepName': 'AbaloneTransform',
  'StartTime': datetime.datetime(2020, 11, 21, 2, 41, 27, 870000, tzinfo=tzlocal()),
  'EndTime': datetime.datetime(2020, 11, 21, 2, 45, 50, 492000, tzinfo=tzlocal()),
  'StepStatus': 'Succeeded',
  'CacheHitResult': {'SourcePipelineExecutionArn': ''},
  'Metadata': {'TransformJob': {'Arn': 'arn:aws:sagemaker:us-
east-2:111122223333:transform-job/pipelines-cfvy1tjuxdq8-abalonetransform-
ptyjoef3jy'}}},
 {'StepName': 'AbaloneRegisterModel',
  'StartTime': datetime.datetime(2020, 11, 21, 2, 41, 26, 929000, tzinfo=tzlocal()),
  'EndTime': datetime.datetime(2020, 11, 21, 2, 41, 28, 15000, tzinfo=tzlocal()),
  'StepStatus': 'Succeeded',
  'CacheHitResult': {'SourcePipelineExecutionArn': ''},
  'Metadata': {'RegisterModel': {'Arn': 'arn:aws:sagemaker:us-
east-2:111122223333:model-package/abalonemodelpackagegroupname/1'}}},
 {'StepName': 'AbaloneCreateModel',
  'StartTime': datetime.datetime(2020, 11, 21, 2, 41, 26, 895000, tzinfo=tzlocal()),
  'EndTime': datetime.datetime(2020, 11, 21, 2, 41, 27, 708000, tzinfo=tzlocal()),
  'StepStatus': 'Succeeded',
  'CacheHitResult': {'SourcePipelineExecutionArn': ''},
  'Metadata': {'Model': {'Arn': 'arn:aws:sagemaker:us-east-2:111122223333:model/
pipelines-cfvy1tjuxdq8-abalonecreatemodel-jl94rai0ra'}}},
 {'StepName': 'AbaloneMSECond',
  'StartTime': datetime.datetime(2020, 11, 21, 2, 41, 25, 558000, tzinfo=tzlocal()),
  'EndTime': datetime.datetime(2020, 11, 21, 2, 41, 26, 329000, tzinfo=tzlocal()),
  'StepStatus': 'Succeeded',
  'CacheHitResult': {'SourcePipelineExecutionArn': ''},
  'Metadata': {'Condition': {'Outcome': 'True'}}},
 {'StepName': 'AbaloneEval',
  'StartTime': datetime.datetime(2020, 11, 21, 2, 37, 34, 767000, tzinfo=tzlocal()),
  'EndTime': datetime.datetime(2020, 11, 21, 2, 41, 18, 80000, tzinfo=tzlocal()),
  'StepStatus': 'Succeeded',
  'CacheHitResult': {'SourcePipelineExecutionArn': ''},
  'Metadata': {'ProcessingJob': {'Arn': 'arn:aws:sagemaker:us-
east-2:111122223333:processing-job/pipelines-cfvy1tjuxdq8-abaloneeval-zfraozhmny'}}},
 {'StepName': 'AbaloneTrain',
  'StartTime': datetime.datetime(2020, 11, 21, 2, 34, 55, 867000, tzinfo=tzlocal()),
  'EndTime': datetime.datetime(2020, 11, 21, 2, 37, 34, 34000, tzinfo=tzlocal()),
  'StepStatus': 'Succeeded',
  'CacheHitResult': {'SourcePipelineExecutionArn': ''},
  'Metadata': {'TrainingJob': {'Arn': 'arn:aws:sagemaker:us-
east-2:111122223333:training-job/pipelines-cfvy1tjuxdq8-abalonetrain-tavd6f3wdf'}}},
 {'StepName': 'AbaloneProcess',
  'StartTime': datetime.datetime(2020, 11, 21, 2, 30, 27, 160000, tzinfo=tzlocal()),
  'EndTime': datetime.datetime(2020, 11, 21, 2, 34, 48, 390000, tzinfo=tzlocal()),
  'StepStatus': 'Succeeded',
  'CacheHitResult': {'SourcePipelineExecutionArn': ''},
  'Metadata': {'ProcessingJob': {'Arn': 'arn:aws:sagemaker:us-
east-2:111122223333:processing-job/pipelines-cfvy1tjuxdq8-abaloneprocess-
mgqyfdujcj'}}}]
```

4. 管道执行完成后，下载结果`evaluation.json`从 Amazon S3 向其提供文件以检查报告。

```
evaluation_json = sagemaker.s3.S3Downloader.read_file("{}/evaluation.json".format(
    step_eval.arguments["ProcessingOutputConfig"]["Outputs"][0]["S3Output"]["S3Uri"]
))
json.loads(evaluation_json)
```

第 3 步：覆盖管道执行的默认参数

您可以通过指定不同的管道参数来覆盖默认值来运行管道的其他执行。

覆盖默认参数

1. 创建管道执行。这将在计算优化的实例类型上启动另一个管道执行，并将模型批准状态设置为 “已批准”。这意味着由RegisterModelstep 已自动准备好通过 CI/CD 管道进行部署，例如 SageMaker 项目。有关更多信息，请参阅 使用实现自动化 mLOP SageMaker 项目 (p. 2507)。

```
execution = pipeline.start(
    parameters=dict(
        ProcessingInstanceType="ml.c5.xlarge",
        ModelApprovalStatus="Approved",
    )
)
```

2. 请等待执行完成。

```
execution.wait()
```

3. 列出执行步骤及其状态。

```
execution.list_steps()
```

4. 管道执行完成后，下载结果evaluation.json从 Amazon S3 向其提供文件以检查报告。

```
evaluation_json = sagemaker.s3.S3Downloader.read_file("{}/evaluation.json".format(
    step_eval.arguments["ProcessingOutputConfig"]["Outputs"][0]["S3Output"]["S3Uri"]
))
json.loads(evaluation_json)
```

第 4 步：停止和删除管道执行

完成使用管道后，您可以停止任何正在进行的执行并删除管道。

停止和删除管道执行

1. 停止管道执行。

```
execution.stop()
```

2. 删除管道。

```
pipeline.delete()
```

查看、跟踪和执行 SageMaker 中的管道 SageMaker 工作室

查看、跟踪和执行亚马逊 SageMaker 亚马逊中的管道 SageMaker 您必须登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。

主题

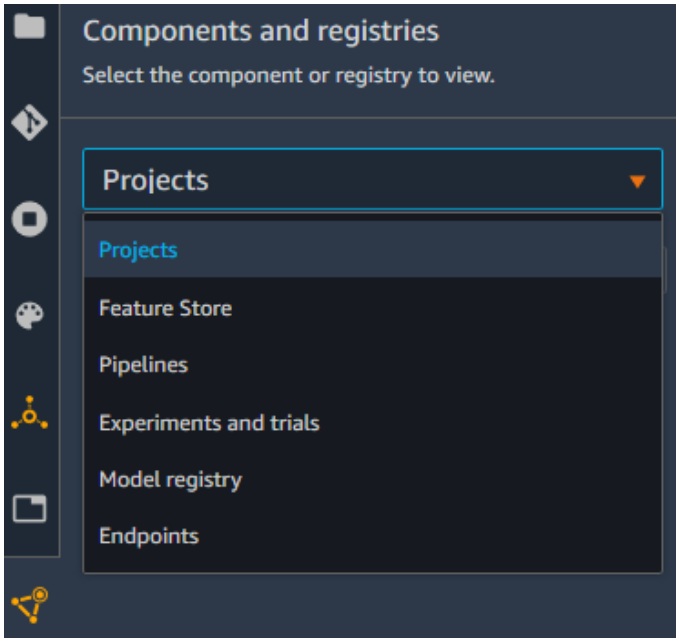
- 查看管道 (p. 2494)
- 查看管道执行 (p. 2497)
- 查看创建者的实验实体 SageMaker 管道 (p. 2499)
- 执行管道 (p. 2501)
- 跟踪的谱系 SageMaker ML 管道 (p. 2503)

查看管道

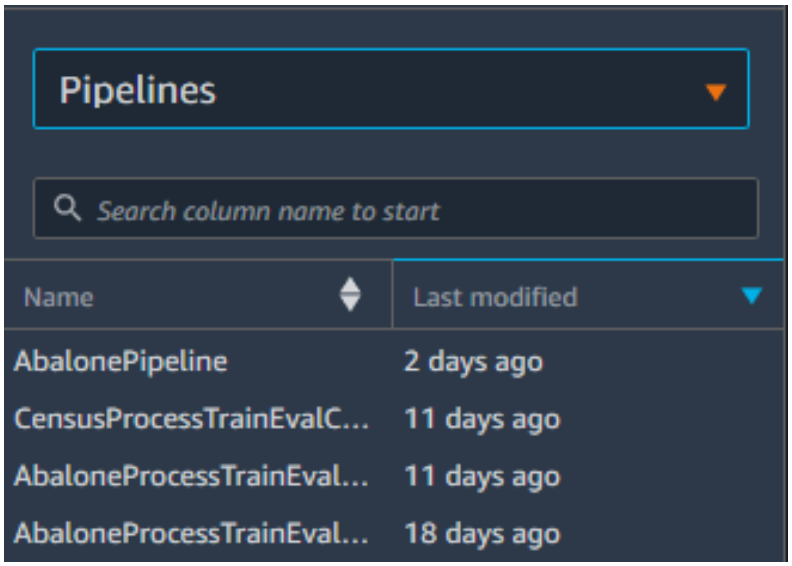
此过程向您演示如何直接查找管道并查看其详细信息页面。您还可以找到属于项目详细信息页面中列出的项目的一部分的管道。有关查找属于项目一部分的管道的信息，请参阅使用实现自动化 mLOP SageMaker 项目 (p. 2507).

查看管道列表

1. 在 Studio 的左侧边栏中，选择SageMaker 资源（图标）。



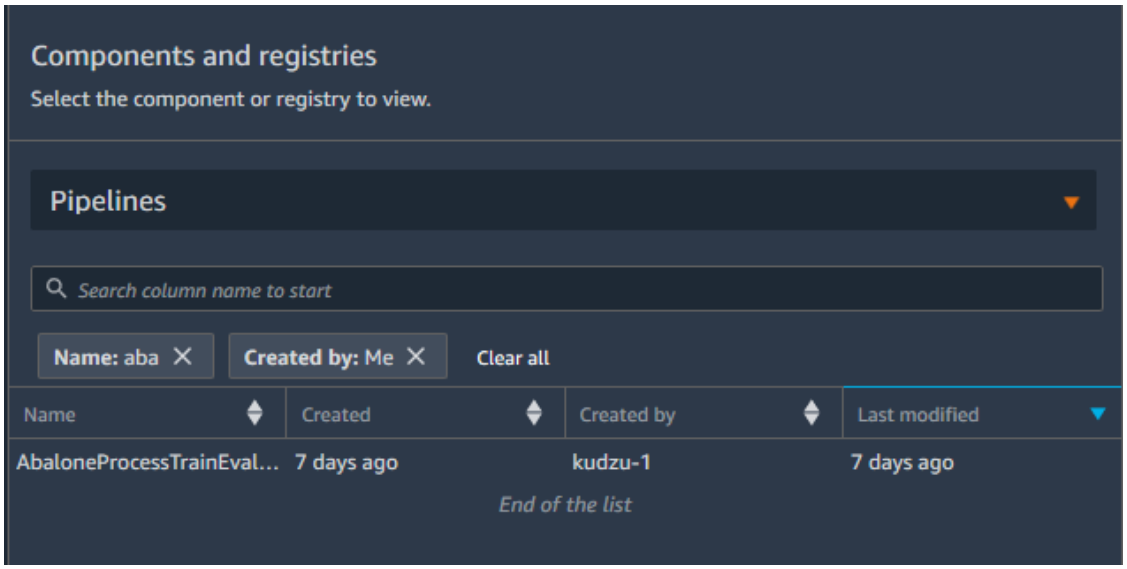
2. Select管道从下拉列表中选择。



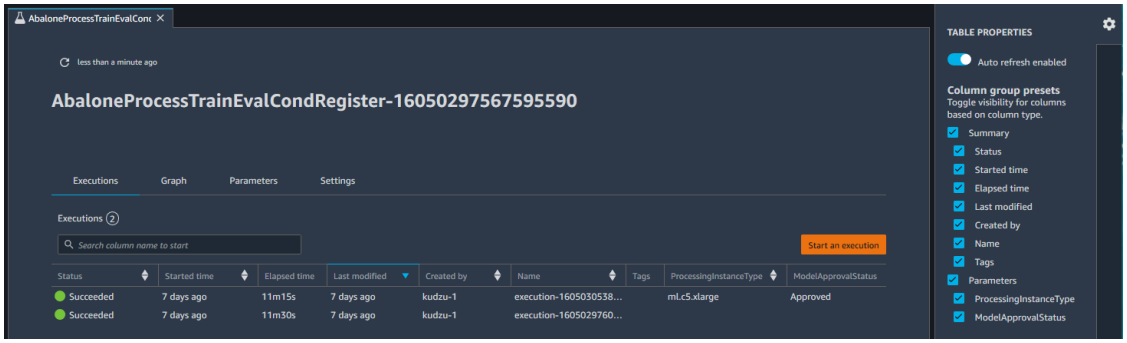
3. 拖动SageMaker 资源窗格向右查看所有列。使用搜索功能缩小管道的范围。

   搜索包括两个步骤。首先，输入与要搜索的列名称匹配的字符。其次，您输入字符以匹配该列中的项目。

您可以拥有多个搜索过滤器。举个例子，以下屏幕截图将显示的管道限制为名称以开头的管道。aba并由创建Me. 有关在 Studio 中进行搜索的更多信息，请参阅使用亚马逊搜索实验 SageMaker 工作室 (p. 1772).

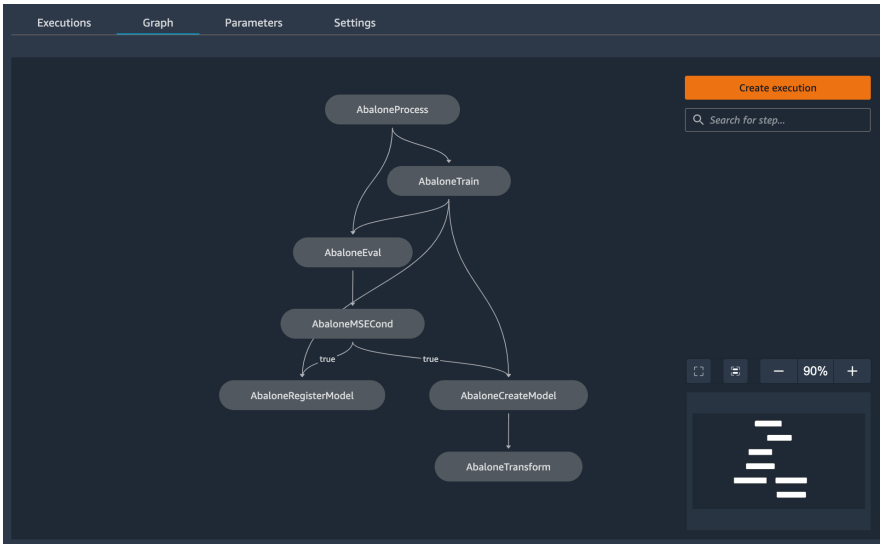


4. 打开管道以查看有关管道的详细信息。管道详细信息选项卡打开，并显示管道执行的列表。您可以开始执行，也可以选择其他选项卡之一以获取有关管道的更多信息。使用设置（图标）来选择要显示哪些列。

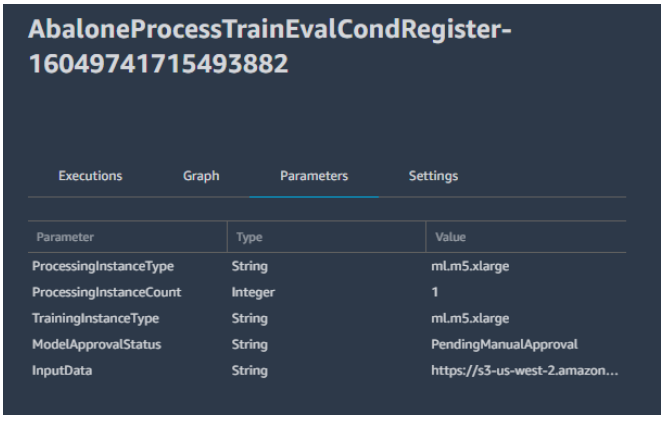


5. 在管道详细信息页面中，选择以下选项卡之一以查看管道的详细信息：

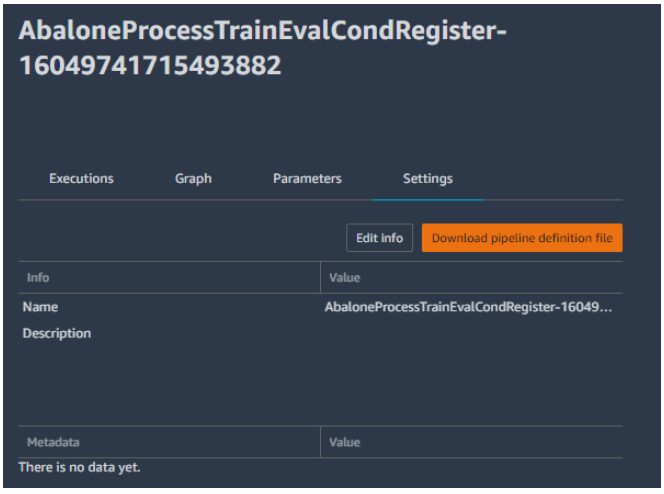
   - 执行— 有关执行的详细信息。你可以从此选项卡或图表选项卡。
   - 图表— 管道的 DAG。



- 参数— 包括模型批准状态。



- 设置— 与管道相关联的元数据。您可以下载管道定义文件并从此选项卡中编辑管道名称和描述。

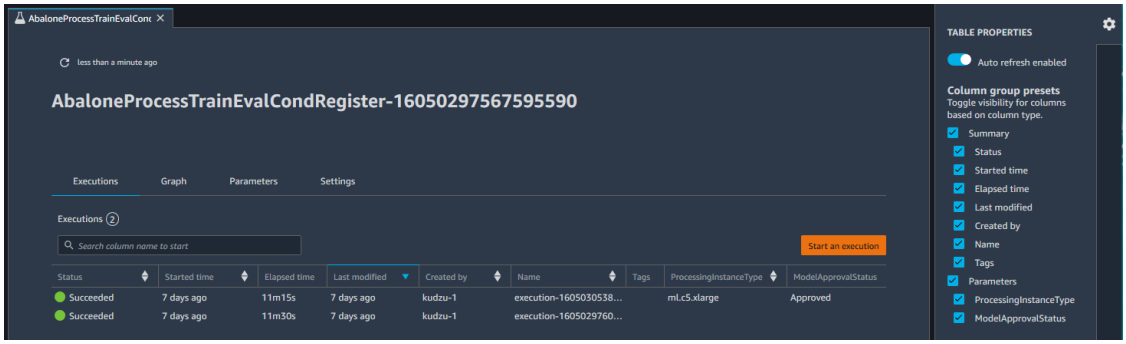


查看管道执行

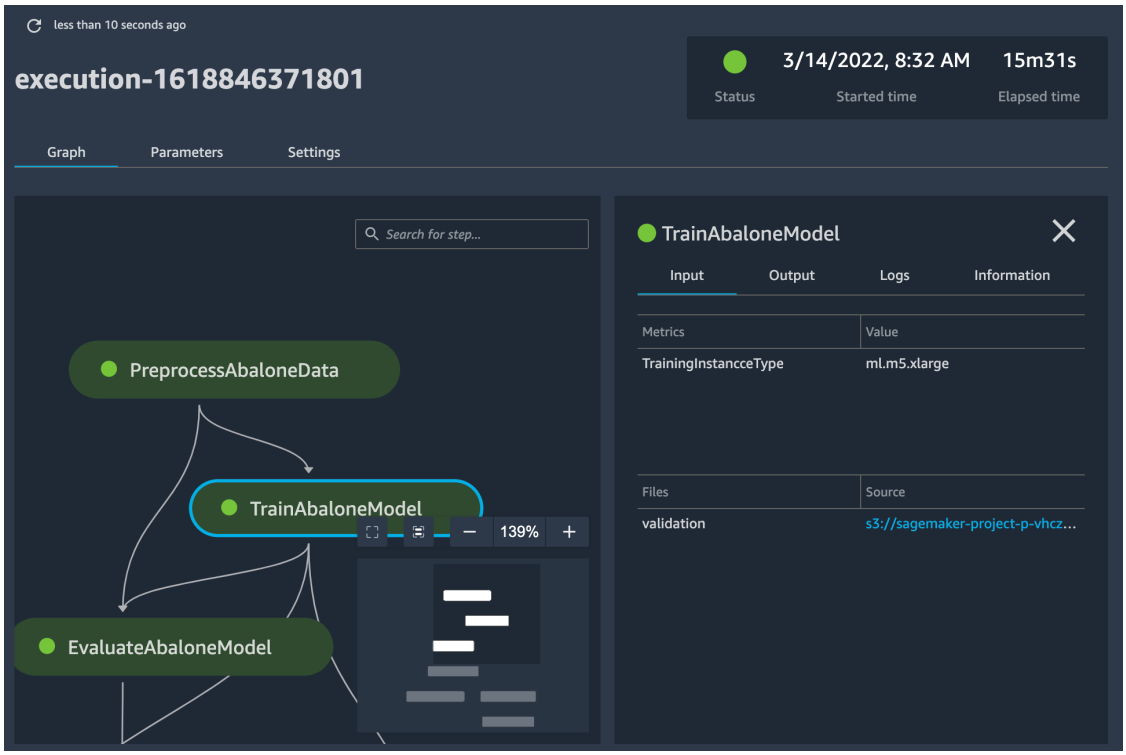
此过程演示如何查看管道执行。有关如何查看管道执行列表以及如何使用的信息。 SageMaker 搜索缩小列表中的执行范围，请参阅查看管道 (p. 2494).

要查看管道执行的详细信息

1. 在执行列表中，打开执行以查看有关执行的详细信息。执行详细信息选项卡随即打开，并显示管道中步骤的图表。您可以选择其他选项卡之一来查看有关管道执行的信息，这与查看管道详细信息时显示的信息类似。

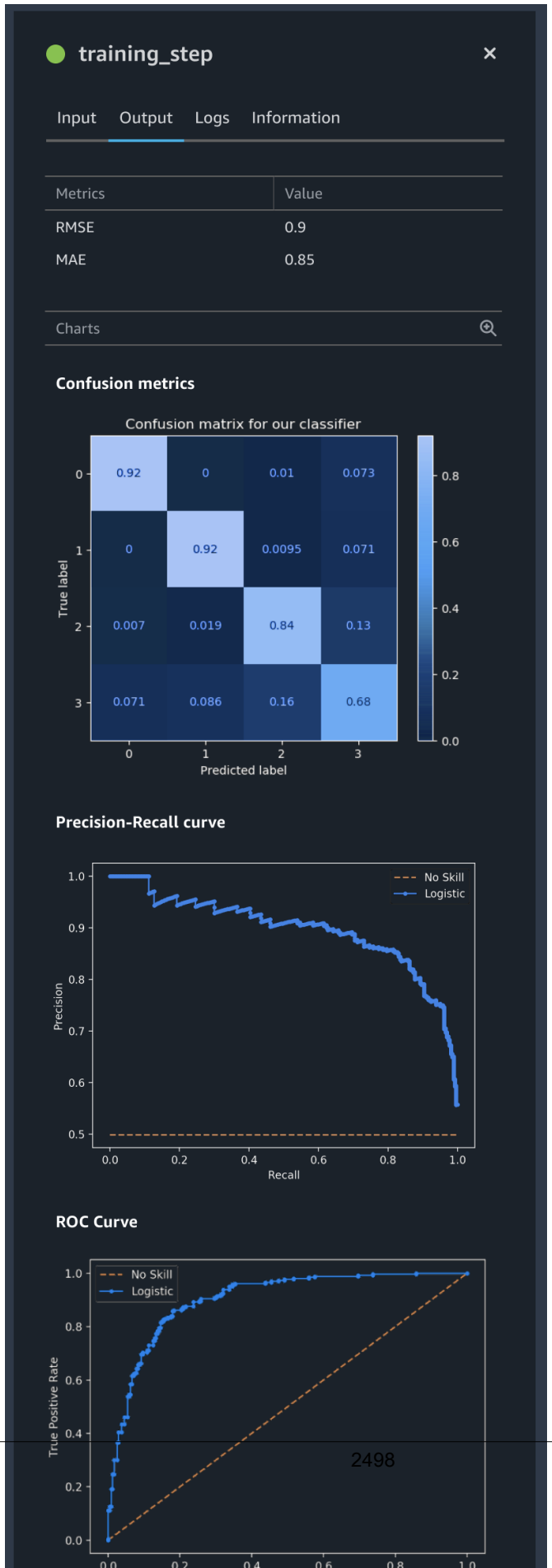


2. 使用搜索在图表中查找步骤。键入与步骤名称匹配的字符。您可以四处拖动图表，也可以使用图表左下角的调整大小图标。图表右下角的插图将显示您在图表中的位置。



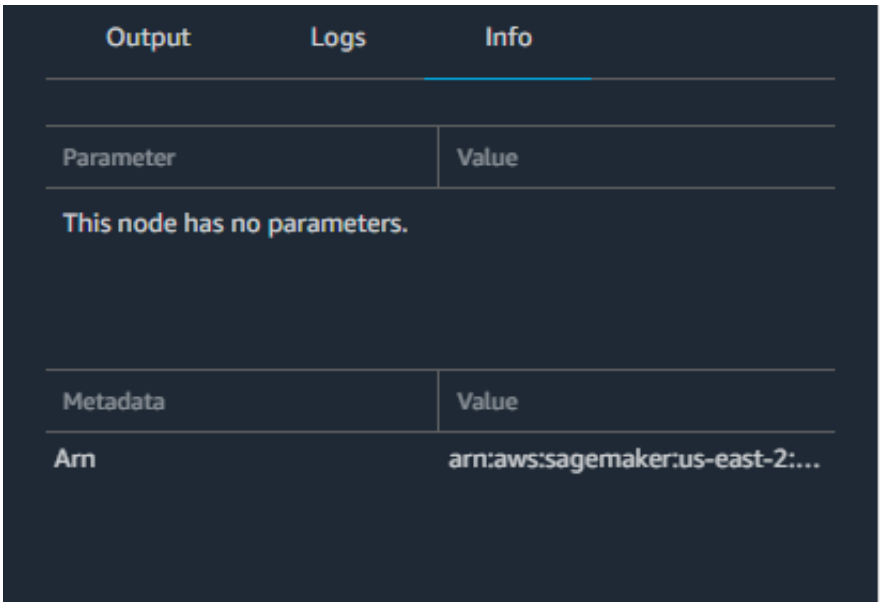
3. 选择图中的步骤之一以查看有关步骤的详细信息。在上面的屏幕截图中，选择了一个训练步骤并显示以下选项卡：

   - 输入— 培训投入。如果输入源来自 Amazon Simple Storage Service (Amazon S3)，则选择链接以在 Amazon S3 控制台中查看文件。
   - 输出— 培训输出，例如指标、图表、文件和评估结果。这些图表是使用跟踪API。


training_step
Input
Output
Logs
Information
Metrics
Value
RMSE
0.9
MAE
0.85
Charts
Confusion metrics
Confusion matrix for our classifier
True label
Predicted label
Precision-Recall curve
No Skill
Logistic
Precision
Recall
ROC Curve
No Skill
Logistic
True Positive Rate

- 日志— Amazon CloudWatch 步骤生成的日志。

- Info— 与该步关联的参数和元数据。



查看创建者的实验实体 SageMaker 管道

当您创建管道并指定pipeline_实验_config、 SageMaker 管道创建以下内容 SageMaker 默认情况下，如果实体不存在，则实验它们：

- 管道的实验
- 每次执行管道的试用
- 每个组件的试用组件 SageMaker 在管道步骤中创建的作业

有关如何与管道集成实验的信息，请参阅亚马逊 SageMaker 实验集成 (p. 2474). 有关 SageMaker 请参阅实验使用亚马逊管理 Machine Learning SageMaker 实验 (p. 1758).

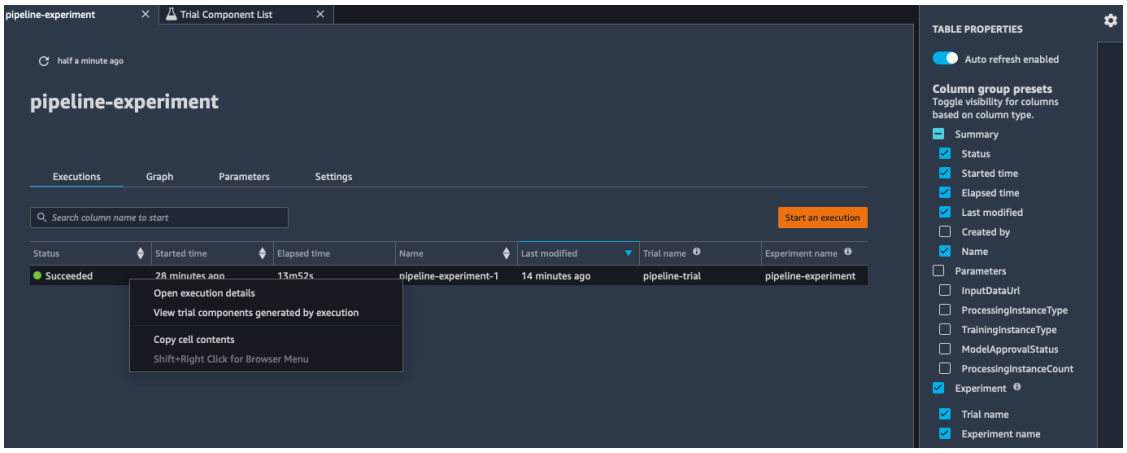
您可以从管道执行列表或实验列表中访问与管道关联的试用组件列表。

从管道执行列表中查看试用组件列表

1. 要查看管道执行列表，请执行中的前四个步骤查看管道 (p. 2494).
2. 在屏幕右上角，选择设置（图标）以打开表属性.
3. Select试验. 如果在创建管道时没有禁用实验集成，则实验名称和试验名称将显示在执行列表中。

   Note

   实验集成在 v2.41.0 中引入亚马逊 SageMaker Python 开发工具包. 默认情况下，使用早期版本的 SDK 创建的管道不与实验集成。
4. 选择一个或多个执行，打开（右键单击）选择，然后选择查看执行生成的试用组件.



从实验列表中查看试验组件列表

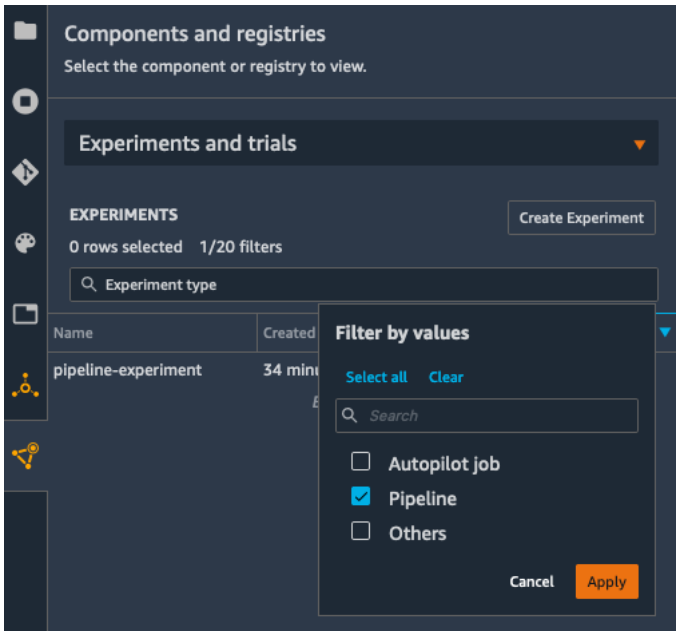
1. 在 Studio 的左侧边栏中，选择SageMaker 资源（图标）。
2. Select试验和试验从下拉列表中选择。
3. 使用搜索功能将列表筛选为管道创建的实验。

   a. 拖动SageMaker 资源向右窗格直到你能看到实验类型column.

   b. 在搜索框中，输入ex然后选择实验类型.

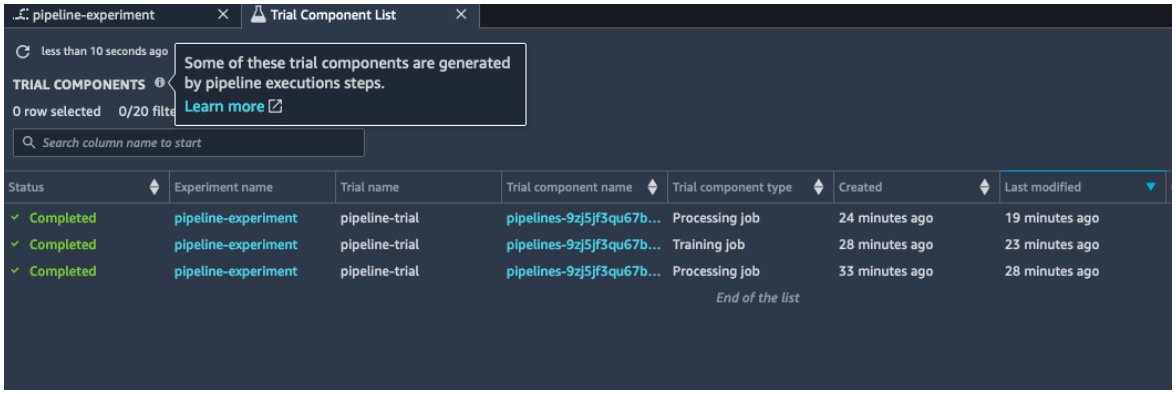
   c. In按值筛选，选择管道选择，然后选择Apply.

   要进一步缩小搜索范围，您可以添加其他筛选器。有关更多信息，请参阅 使用亚马逊搜索实验 SageMaker 工作室 (p. 1772)。
4. 打开（右键单击）实验名称，然后选择在试用组件列表中打开.



试用组件列表

下面显示了管道创建的试用组件的列表。

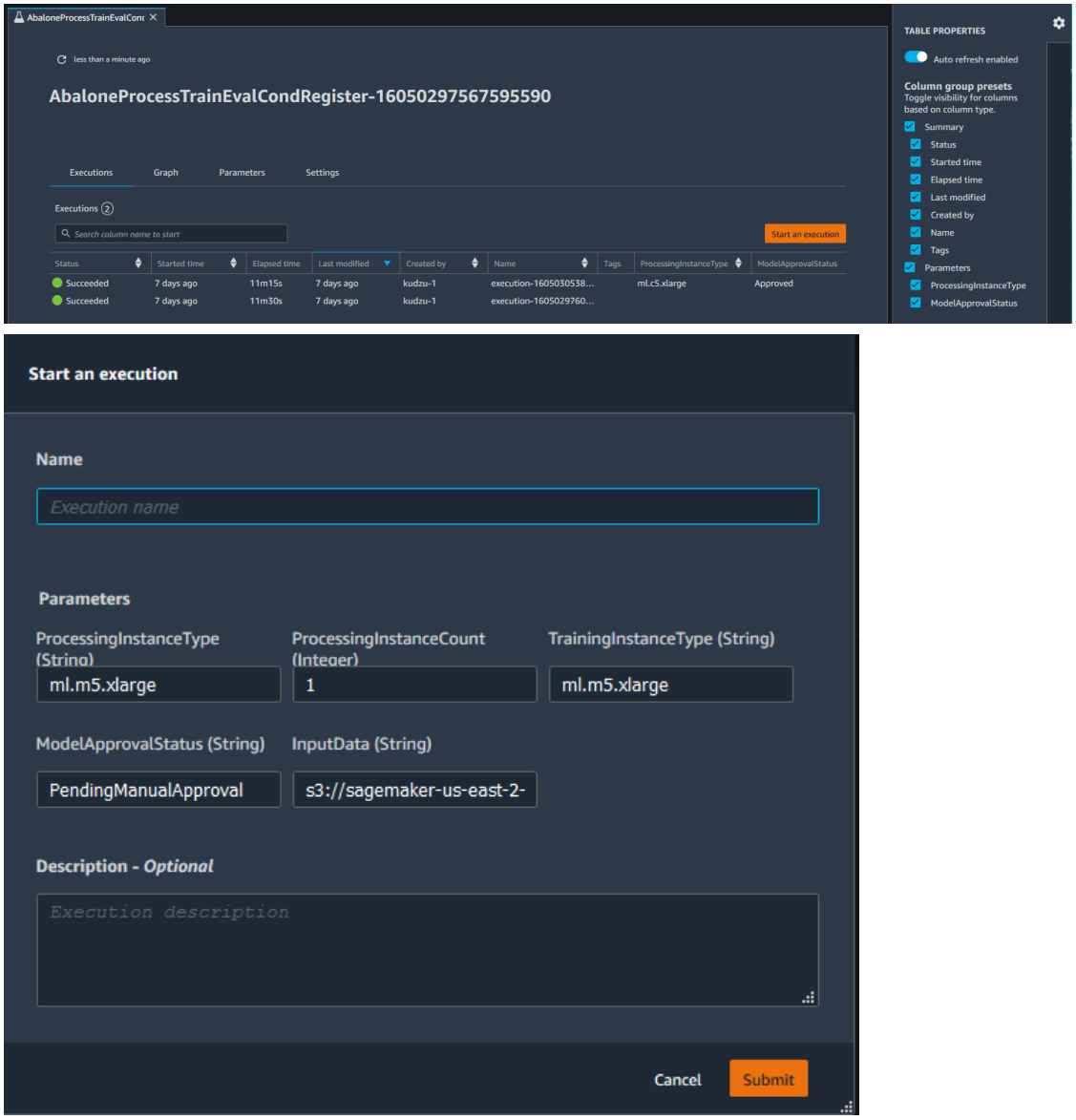


执行管道

此过程演示如何执行管道。有关如何查看管道执行列表的信息，请参阅。查看管道 (p. 2494).

要启动管道执行

1. 从执行要么图表在执行列表中选择选项卡开始执行.



2. 输入或更新以下必填信息：
   - 名称— 必须是您账户的唯一名称Amazon区域。
   - 处理实例类型— 要处理的实例类型。
   - 正在处理实例计数— 用于处理的实例数。
   - 训练实例类型— 用于培训的实例类型。
   - 模型批准状态— 为了方便起见。
   - 输入数据— 输入数据的 S3 URI。
3. 选择 Submit（提交）。

- 要查看执行的详细信息或停止执行，请选择查看详细信息在状态横幅上。
- 要停止执行，请选择停止在状态横幅上。
- 要从停止的地方恢复执行，请选择Resume在状态横幅上。

Note

如果管道失败，状态横幅将显示已失败状态。排查失败步骤后，选择重试在状态横幅上，以便从该步骤恢复运行管道。

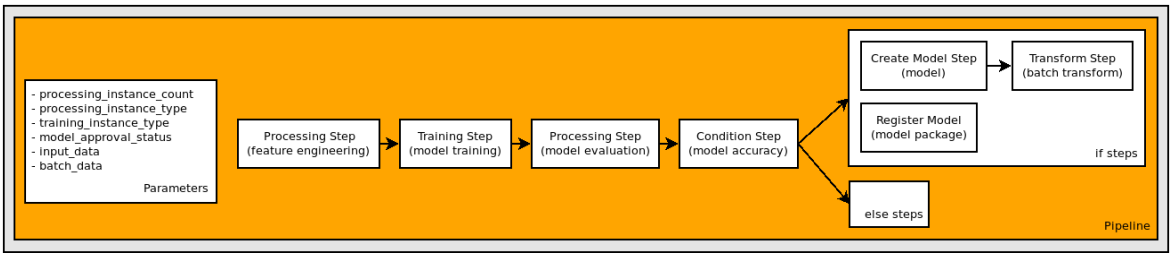
有关注册型号的列表，请参阅使用实现自动化 mLOP SageMaker 项目 (p. 2507).

跟踪的谱系 SageMaker ML 管道

在本教程中，您将使用 Amazon SageMaker 追踪亚马逊血统的工作室 SageMaker ML 管道。

创建管道是由与亚马逊协调工作 SageMaker 模型构建管道中的笔记本亚马逊 SageMaker 示例 GitHub 知识库. 有关如何创建管道的详细信息，请参阅定义管道 (p. 2478).

Studio 中的世系跟踪以有向无环图 (DAG) 为中心。DAG 表示管道中的步骤。从 DAG 中，您可以跟踪从任何步骤到任何其他步骤的世系。下图显示了管道中的步骤。这些步骤在 Studio 中显示为 DAG。

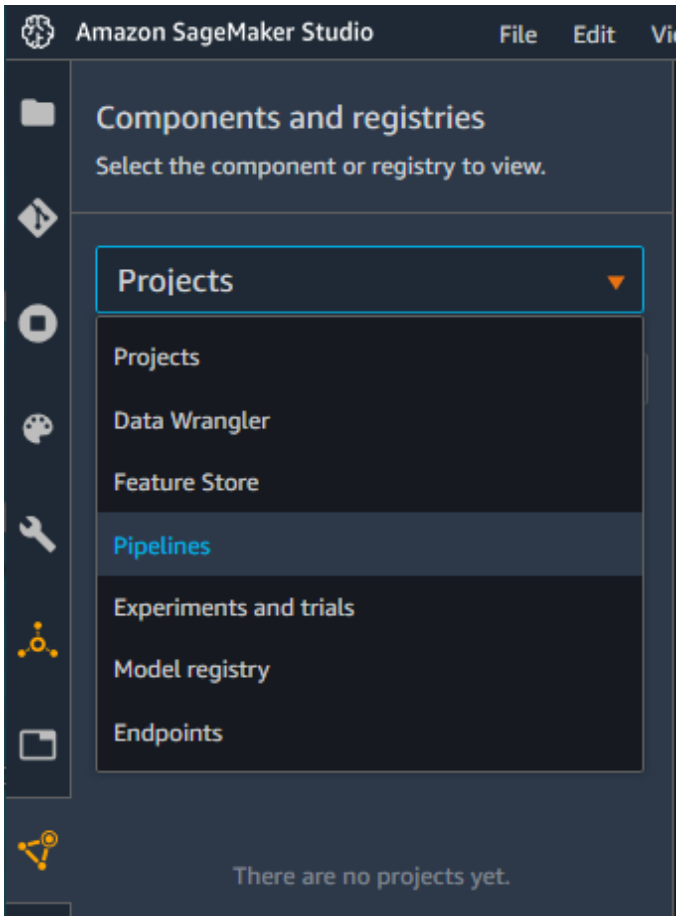


先决条件

- 访问 Amazon SageMaker Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
- 熟悉 SageMaker Studio 用户界面。有关更多信息，请参阅 亚马逊 SageMaker Studio UI 概述 (p. 77)。
- （推荐）示例笔记本的已完成运行。

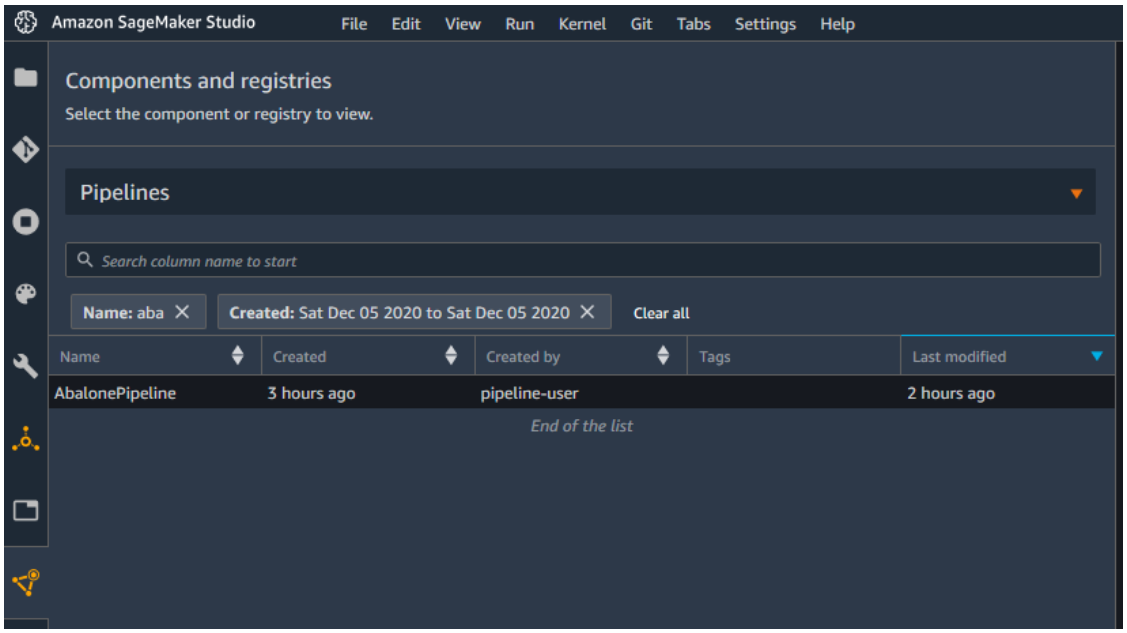
跟踪管道的血统

1. 登录到 SageMaker Studio。
2. 在 Studio 的左侧边栏中，选择SageMaker 资源（图标）。
3. 在下拉菜单中，选择管道.

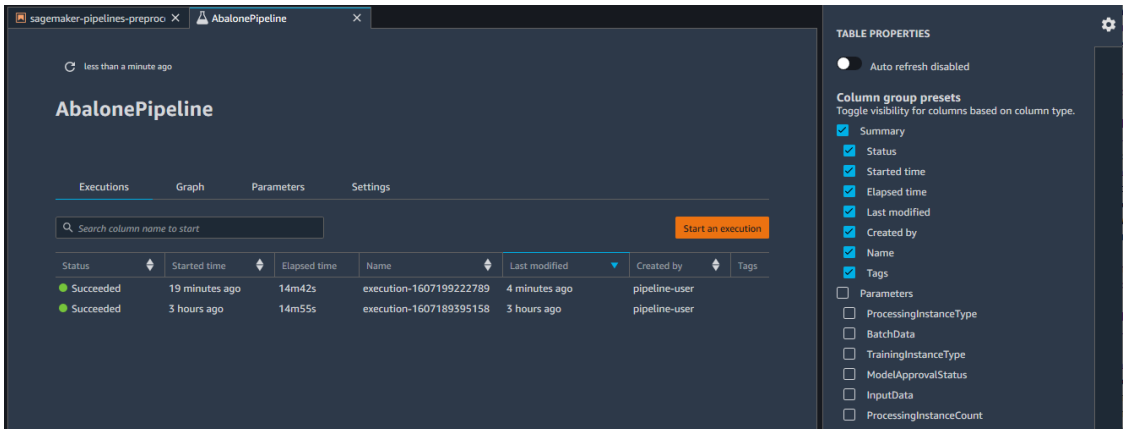


4. 使用搜索框来筛选管道列表。要查看所有可用的列，请将窗格的右边框向右拖动。有关更多信息，请参阅 使用亚马逊搜索实验 SageMaker 工作室 (p. 1772)。

   以下屏幕截图显示了按以 “aba” 开头且于 12 月 5 日创建于 12 月 5 日的名称过滤的列表。

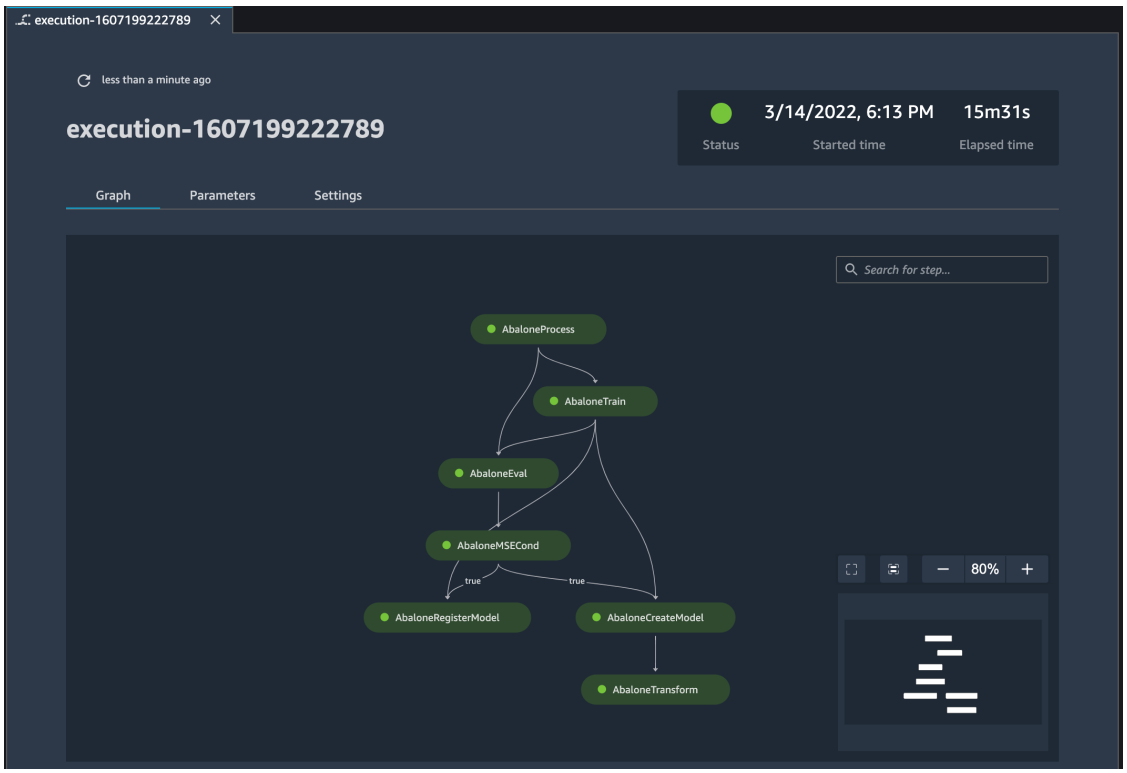


5. 双击AbalonePipeline管道来查看执行列表和有关管道的其他详细信息。以下屏幕截图显示表属性窗格打开，您可以在其中选择要查看的属性。

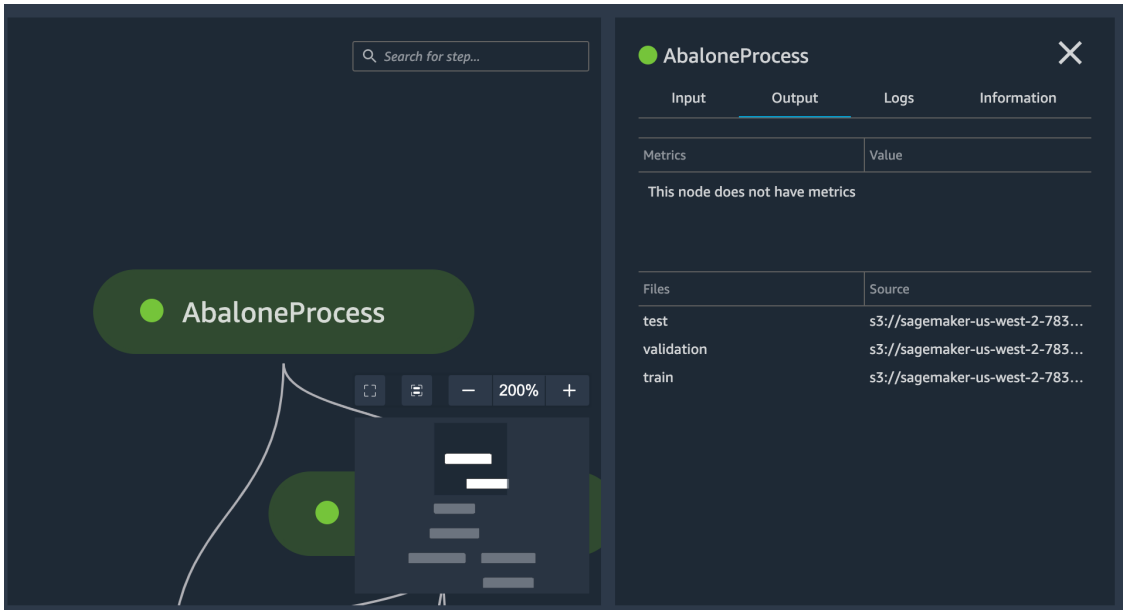


6. 选择设置选项卡，然后选择下载管道定义文件. 您可以查看文件以查看管道图是如何定义的。
7. 在存储库的执行选项卡上，双击执行列表中的第一行以查看其执行图表和有关执行的其他详细信息。请注意，此图表与本教程开始时显示的图表一致。

   您可以四处拖动图表（选择不在图表本身上的区域）或使用图表左下角的调整大小图标。图表右下角的插图显示您在图表中的位置。



8. 在存储库的图表选项卡上，选择AbaloneProcess步骤以查看有关该步骤的详细信息。



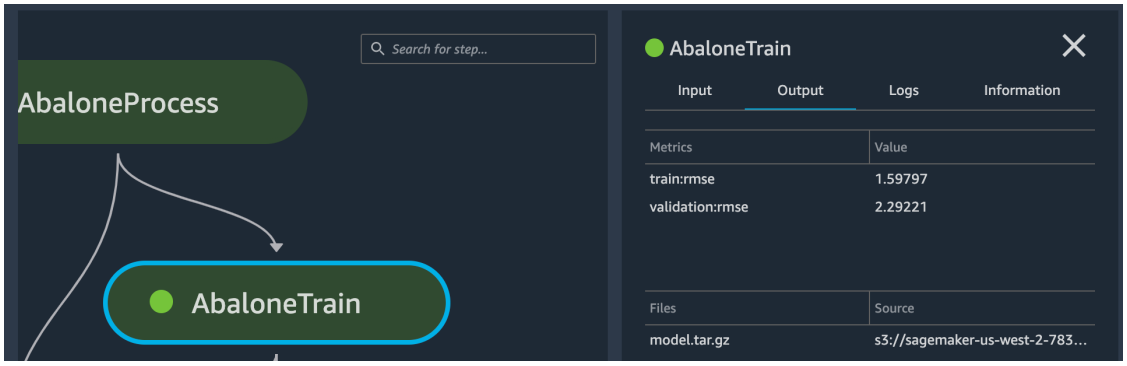
9. Amazon S3输出选项卡，下文件.

   Note

   要获取完整路径，请右键单击该路径，然后选择复制单元格内容.

```
s3://sagemaker-eu-west-1-acct-id/sklearn-abalone-process-2020-12-05-17-28-28-509/
output/train
s3://sagemaker-eu-west-1-acct-id/sklearn-abalone-process-2020-12-05-17-28-28-509/
output/validation
s3://sagemaker-eu-west-1-acct-id/sklearn-abalone-process-2020-12-05-17-28-28-509/
output/test
```

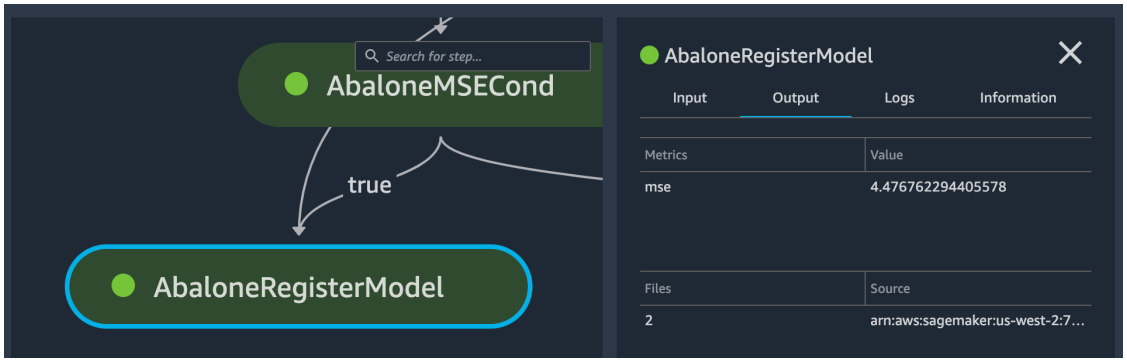
10. 选择`AbaloneTrain`步骤。



11. 在中找到模型项目的 Amazon S3 路径输出选项卡，下文件：

```
s3://sagemaker-eu-west-1-acct-id/AbaloneTrain/pipelines-6locnsqz4bfu-AbaloneTrain-
NtfEpI0Ahu/output/model.tar.gz
```

12. 选择`AbaloneRegisterModel`步骤。



13. 在输出选项卡，下文件：

```
arn:aws:sagemaker:eu-west-1:acct-id:model-package/abalonemodelpackagegroupname/2
```

使用实现自动化 mLOP SageMaker 项目

Create end-to-end 使用CI/CD 的 ML 解决方案 SageMaker 项目。

使用 SageMaker 旨在创建一个 mLOP 解决方案来编排和管理的项目：

- 构建用于处理、训练和推理的自定义图像
- 数据准备和特征工程
- 训练模型
- 评估模型
- 部署模型
- 监控和更新模型

主题

- 你为什么要使用 mLOP？ (p. 2507)
- 什么是 SageMaker 项目？ (p. 2509)
- 使用项目所需的 SageMaker Studio 权限 (p. 2511)
- 使用亚马逊创建 mLOPS 项目 SageMaker 工作室 (p. 2512)
- mLOPS 项目模板 (p. 2512)
- 查看项目资源 (p. 2519)
- 更新亚马逊中的 mLOPS 项目 SageMaker 工作室 (p. 2520)
- 使用亚马逊删除 mLOPS 项目 SageMaker 工作室 (p. 2521)
- SageMaker mLOPS 项目演练 (p. 2522)

你为什么要使用 mLOP？

当您从运行个人人工智能和机器学习 (AI/ML) 项目转向使用 AI/ML 大规模实现业务转型时，机器学习运营 (mLOP) 的纪律可以提供帮助。MPLOP 考虑了 AI/ML 项目在项目管理、CI/CD 和质量保证方面的独特方面，帮助您缩短交付时间、减少缺陷并提高数据科学的效率。mLOP 是指建立在应用基础上的方法 DevOps 机器学习工作负载的实践。对于讨论 DevOps 原则，请参阅白 paper简介 DevOps 上Amazon. 要了解有关实施的更多信息Amazon服务，请参阅在练 CI/CDAmazon和代码基础设施.

与 DevOps 一样，MPLOP 依赖于机器学习开发生命周期的协作和简化方法，其中人员、流程和技术的交叉优化了 end-to-end 开发、构建和操作机器学习工作负载所需的活动。

mLOP 重点关注数据科学和数据工程与现有结合的交叉点 DevOps 在整个机器学习开发生命周期中简化模型交付的做法。mLOP 是将机器学习工作负载集成到发布管理、CI/CD 和操作中的学科。MPLOP 需要软件开发、运营、数据工程和数据科学的集成。

MPLOP 面临的挑战

尽管 mLOP 可以提供有价值的工具来帮助您扩展业务，但在将 mLOP 集成到机器学习工作负载中时，您可能会面临某些问题。

项目管理

- 机器学习项目涉及数据科学家，这是一个相对较新的角色，而且不经常集成到跨职能团队中。这些新团队成员通常使用的技术语言与产品所有者和软件工程师截然不同，这使得将业务需求转化为技术要求的常见问题更加复杂。

沟通与协作

- 建立机器学习项目的可见性，实现不同利益相关方之间的协作，例如数据工程师、数据科学家、机器 DevOps 对于确保取得成功结果而言变得越来越重

一切都是代码

- 在开发活动中使用生产数据、更长的实验生命周期、对数据管道的依赖性、再培训部署管道以及评估模型性能的独特指标。
- 模型的生命周期通常独立于与这些模型集成的应用程序和系统。
- 整个 end-to-end 系统可以通过版本控制的代码和工件重现。 DevOps 项目使用基础设施即代码 (iAC) 和配置即代码 (CAC) 来构建环境，并使用管道即代码 (PAC) 来确保 CI/CD 模式的一致性。管道必须与大数据和机器学习培训工作流程集成。这通常意味着管道是传统的 CI/CD 工具和另一个工作流引擎的组合。许多机器学习项目都存在重要的政策问题，因此管道可能还需要强制执行这些政策。有偏见的输入数据会产生有偏见的结果，这是企业利益相关者

CI/CD

- 在 mLOP 中，源数据与源代码一起是一流的输入。这就是为什么 mLOP 呼吁对源数据进行版本控制并在源数据或推理数据发生变化时启动管道运行的原因。
- 为了提供可追溯性，管道还必须对机器学习模型以及输入和其他输出进行版本化。
- 自动测试必须包括在构建阶段和模型投入生产时对机器学习模型进行正确验证。
- 构建阶段可能包括模型训练和再培训，这是一个耗时且资源密集型的过程。管道必须足够细化，以便仅在源数据或机器学习代码发生更改时执行完整的训练周期，而不是在相关组件更改时执行。
- 由于机器学习代码通常是整体解决方案的一小部分，因此部署管道还可能包含将模型打包为其他应用程序和系统作为 API 使用所需的额外步骤。

监控和日志记录

- 捕获模型训练指标和模型实验所需的特征工程和模型训练阶段。调整 ML 模型需要操作输入数据的形式以及算法超参数，并系统地捕获这些实验。实验跟踪有助于数据科学家更有效地工作，并为他们的工作提供可重复的快照。
- 已部署的机器学习模型需要监控传递给模型进行推理的数据，以及标准端点稳定性和性能指标。监测系统还必须根据适当的机器学习指标评估，捕获模型输出的质量。

MPLOP 的优势

采用 mLOP 实践可以让你更快 time-to-market 通过提供以下好处来实现机器学习项目。

- 生产力：通过提供对精选数据集的访问权限的自助服务环境，可以让数据工程师和数据科学家更快地移动，减少对丢失或无效数据的
- 可重复性：自动执行 MOLDC 中的所有步骤有助于确保可重复的过程，包括如何训练、评估、版本控制和部署模型。
- 可靠性：结合 CI/CD 实践不仅可以快速部署，而且能够提高质量和一致性。
- 可审核性：对所有输入和输出进行版本控制，从数据科学实验到源数据再到训练模型，这意味着我们可以准确地展示模型是如何构建和部署在哪里的。
- 数据和模型质量：MPLOP 使我们能够强制执行防范模型偏差的政策，并跟踪随时间推移对数据统计属性和模型质量的变化。

什么是 SageMaker 项目？

SageMaker Projects 可以帮助组织为数据科学家设置和标准化开发人员环境，为 mLOP 工程师设置和标准化 CI/CD 系统。项目还可以帮助组织设置依赖关系管理、代码库管理、构建可重复性和工件共享。

您可以预置 SageMaker 来自Amazon使用自定义模板或 SageMaker 提供的模板的 Service Catalog。有关的信息Amazon请参阅 Service Catalog什么是AmazonService Catalog. 与 SageMaker 项目、mLOP 工程师和组织管理员可以定义自己的模板或使用 SageMaker 提供的模板。SageMaker 提供的模板通过源代码版本控制、自动机器学习管道和一组代码来引导机器学习工作流程，以便快速开始迭代 ML 用例。

什么时候应该使用 SageMaker 项目？

虽然笔记本有助于模型构建和实验，但共享代码的数据科学家和机器学习工程师团队需要一种更可扩展的方式来维护代码一致性和严格的版本控制。

每个组织都有自己的一套标准和做法，为其提供安全和治理Amazon环境。 SageMaker 为希望快速开始使用机器学习工作流程和 CI/CD 的组织提供了一套第一方模板。模板包括使用的项目Amazon-CI/CD 的原生服务，例如Amazon CodeBuild、Amazon CodePipeline, 和Amazon CodeCommit. 这些模板还提供了创建使用第三方工具（例如 Jenkins 和 GitHub）的项目的选项。要获取以下项目模板列表： SageMaker 提供了，请参阅使用 SageMaker 提供的项目模板 (p. 2513).

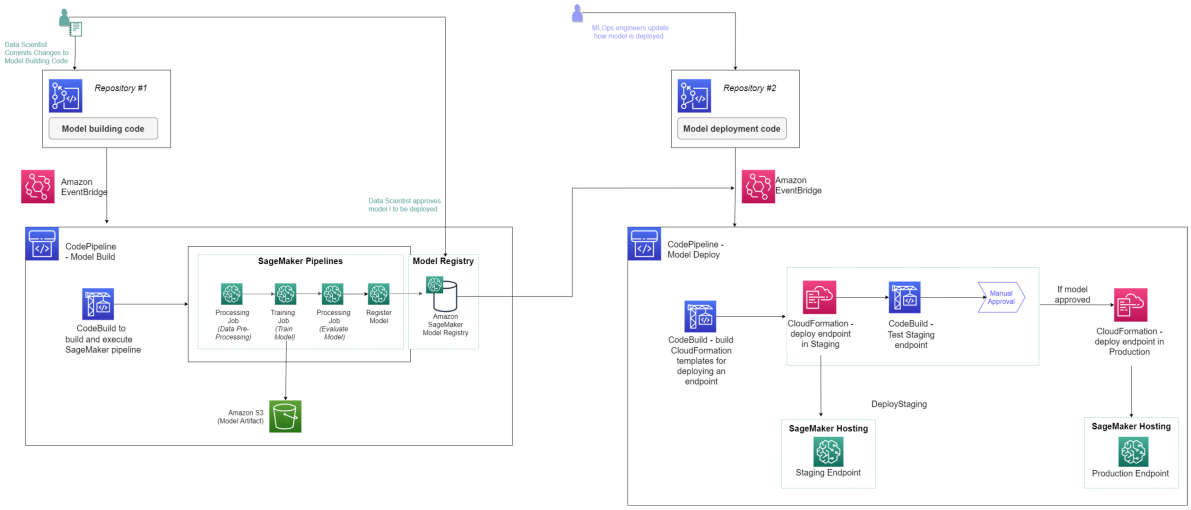
Organizations 通常需要严格控制他们预配和管理的 mLOP 资源。此类责任承担某些任务，包括配置 IAM 角色和策略、强制执行资源标签、强制加密以及跨多个账户解耦资源。 SageMaker 项目可以通过组织使用的自定义模板产品支持所有这些任务Amazon CloudFormation用于定义机器学习工作流所需资源的模板。数据科学家可以选择模板来引导和预配置他们的机器学习工作流。这些自定义模板的创建方式为Amazon Service Catalog产品，你可以在 Studio UI 下配置它们组织模板. 这些区域有：Amazon Service Catalog是一项服务，可帮助组织创建和管理已批准在上使用的产品的目录。Amazon. 有关创建自定义模板的更多信息，请参阅构建自定义 SageMaker 项目模板 — 最佳实践.

SageMaker Projects 可以帮助你管理 Git 仓库，以便你可以在团队之间更有效地协作，确保代码一致性并支持 CI/CD。 SageMaker 项目可以帮助您完成以下任务：

- 将 ML 生命周期的所有实体组织在一个项目下。
- 建立一键式方法，为模型训练和部署设置标准机器学习基础架构，其中包含了最佳实践。
- 为机器学习基础架构创建和共享模板以服务多个使用案例。
- 利用 SageMaker 提供的预构建模板快速开始专注于模型构建，或者使用组织特定的资源和指南创建自定义模板。
- 通过扩展项目模板与您选择的工具集成。有关示例，请参阅 。创建 SageMaker 要与之集成的项目 GitLab 和 GitLab 管道.
- 将 ML 生命周期的所有实体组织在一个项目下。

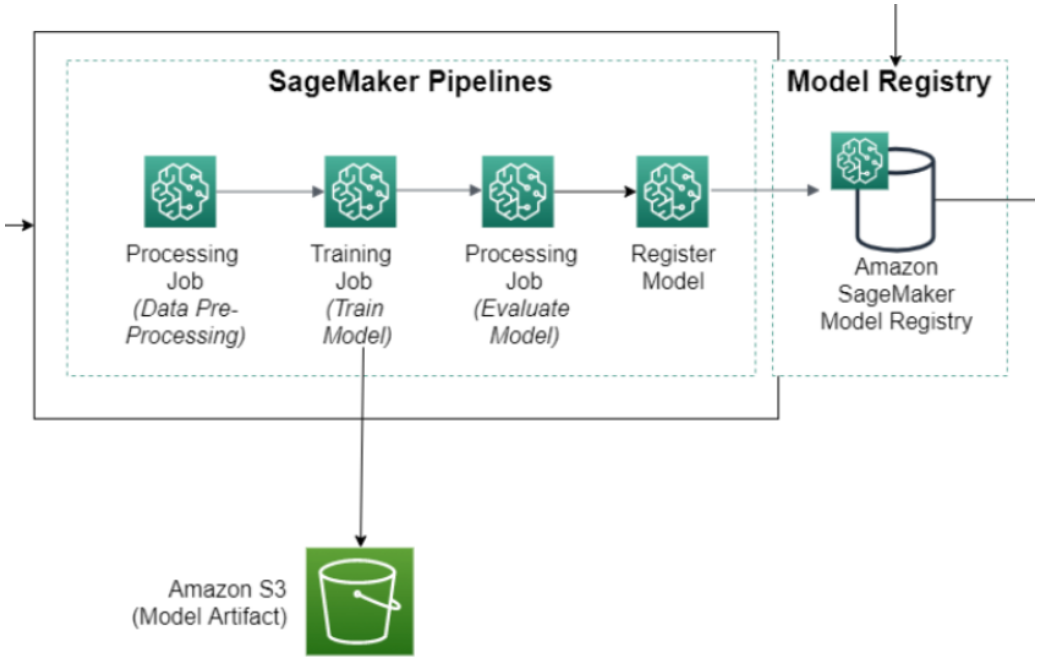
什么是 SageMaker 项目？

客户可以灵活地使用最适合其使用案例的资源来设置自己的项目。下面的示例展示了机器学习工作流程的 mLOP 设置，包括模型训练和部署。



使用 SageMaker 提供的模板的典型项目可能包括以下内容：

- 一个或多个具有示例代码的存储库来构建和部署 ML 解决方案。这些是可行的示例，您可以在本地克隆并根据需要进行修改。您拥有此代码，可以利用版本控制的存储库来执行任务。
- 一个 SageMaker 用于定义数据准备、训练、模型评估和模型部署的步骤，如下图所示。



- 一个 CodePipeline 或者运行你的 Jenkins 管道 SageMaker 每次签入新版本的代码时管道。有关 CodePipeline 的信息，请参阅是什么AmazonCodePipeline。有关 Jenkins 的更多信息，请参阅Jenkins 用户文档.

- 包含模型版本的模型组。每次批准生成的模型版本时 SageMaker 管道运行，你可以将其部署到 SageMaker 终端节点。

EQUE SageMaker 项目具有唯一的名称和 ID，作为标签应用于所有 SageMaker 和Amazon在项目中创建的资源。使用名称和 ID，您可以查看与项目关联的所有实体。其中包括：

- 管道
- 注册模型
- 已部署的模型（端点）
- 数据集
- Amazon Service Catalog产品
- CodePipeline 和詹金斯管道
- CodeCommit 和第三方 Git 存储库

我需要创建一个项目才能使用 SageMaker Pipeline 吗？

否。 SageMaker 管道是独立的实体，就像培训作业、处理作业和其他 SageMaker 个作业。您可以直接在笔记本中创建、更新和运行管道，方法是使用 SageMaker Python SDK 不使用 SageMaker 项目。

项目提供了一个额外的层来帮助您组织代码并采用生产质量系统所需的最佳操作实践。

使用项目所需的 SageMaker Studio 权限

用户可以查看 SageMaker 提供项目模板并在授予时使用这些模板创建项目项目用户的权限。在登录或更新亚马逊时，您可以授予这些权限。 SageMaker Studio。有两个权限可以授予。

1. Grant项目Studio 管理员允许 Studio 管理员查看 SageMaker 提供的模板的权限Amazon Service Catalog控制台。如果您授予其他 Studio 用户的使用权限，管理员可以看到他们创建的内容 SageMaker 项目。管理员还可以查看Amazon CloudFormationSageMaker 提供的项目模板在Amazon Service Catalog控制台。有关使用Amazon Service Catalog控制台，请参阅什么是Amazon Service Catalog中的Amazon Service Catalog用户指南.
2. 允许配置为使用域执行角色的 Studio 用户创建项目。这将授予 Studio 用户使用 SageMaker 提供的项目模板在 Studio 内创建项目的权限。

对于使用域执行角色以外的任何角色查看和使用 SageMaker 提供的项目模板的用户，您需要授予项目对单个用户配置文件的权限。

以下步骤显示了如何授予项目加入 Studio 后的权限。有关加入 Studio 的更多信息，请参阅注册到 Amazon SageMaker 域 (p. 32).

授予项目管理员和域执行角色用户的权限

1. 打开SageMaker 控制台.
2. 选择控制面板.
3. 如果选择快速设置要设置 SageMaker Domain，默认情况下，您有权使用项目模板。
4. 如果选择标准设置设置 SageMaker 域名，请确保你同时打开两者启用项目，以及 JumpStart 该 Studio 用户的模板和启用项目，以及 JumpStart Studio 用户的模板当你配置 Studio 设置时。
5. 为了确认 SageMaker 域具有活动的项目模板权限：
   a. 打开SageMaker 控制台.
   b. 选择控制面板.

c. 选择设置图标域卡。

d. 选择Studio 设置在左侧面板中。

e. UNDERJumpStart 项目，确保启用项目，以及 JumpStart 该 Studio 用户的模板和启用项目，以及 JumpStart Studio 用户的模板已打开。

使用亚马逊创建 mLOPS 项目 SageMaker 工作室

此过程演示如何使用亚马逊创建 mLOPS 项目。 SageMaker Studio。

先决条件

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的IAM Identity Center用于登录到 Studio 的 IAM 账户。有关信息，请参阅注册到 Amazon SageMaker 域 (p. 32)。
- 允许使用 SageMaker 提供的项目模板。有关信息，请参阅使用项目所需的 SageMaker Studio 权限 (p. 2511)。
- 基本熟悉 Studio 用户界面。有关信息，请参阅亚马逊 SageMaker Studio UI 概述 (p. 77)。

在 Studio 中创建项目

1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
2. 在 Studio 边栏中，选择SageMaker 资源（图标）。
3. Select项目从下拉列表中选择。
4. 请选择 Create project（创建项目）。

   这些区域有：创建项目选项卡打开，显示可用模板列表。
5. 适用于SageMaker 项目模板，选择SageMaker 模板. 有关项目模板的更多信息，请参阅mLOPS 项目模板 (p. 2512).
6. 选择用于模型构建、培训和部署的 mLOPS 模板.
7. 选择选择项目模板.

   这些区域有：创建项目选项卡更改要显示项目详情.
8. 输入以下信息：
   - 适用于项目详情中，输入项目的名称和描述。
   - 或者，添加标签，键值对您可以用于跟踪项目。
9. 选择创建项目等待项目出现在项目列表。

mLOPS 项目模板

Amazon SageMaker 项目模板可以自动设置和实施项目的 mLOP。一个 SageMaker 项目模板是Amazon Service Catalog产品 SageMaker 向亚马逊提供 SageMaker 工作室用户。这些Amazon Service Catalog产品在您的Amazon Service Catalog在加载或更新亚马逊时启用权限之后的控制台 SageMaker Studio。有关启用权限以使用 SageMaker 项目模板的信息，请参阅使用项目所需的 SageMaker Studio 权限 (p. 2511). 使用 SageMaker 项目模板来创建一个项目 end-to-end mLOPS 解决方案。

如果您是管理员，则可以从头开始创建自定义项目模板或修改 SageMaker 提供的其中一个项目模板。组织中的 Studio 用户可以使用这些自定义项目模板创建他们的项目。

主题

- 使用 SageMaker 提供的项目模板 (p. 2513)
- 创建自定义项目模板 (p. 2518)

使用 SageMaker 提供的项目模板

亚马逊 SageMaker 提供项目模板，用于创建 MLOPS 解决方案以便为 ML 模型的连续集成和连续部署 (CI/CD) 创建 MLOPS 解决方案所需的基础设施。使用这些模板处理数据、提取要素、训练和测试模型、在 SageMaker 模型注册表，然后部署模型进行推理。您可以自定义种子代码和配置文件以满足自己的要求。

> Important
>
> 截至 2021 年 7 月 27 日， SageMaker 项目可以使用第三方 Git 存储库。有关更多信息，请参阅 更新 SageMaker 要使用第三方 Git 存储库的项目 (p. 2518)。

SageMaker 项目模板为您提供以下代码存储库、工作流自动化工具和管道阶段的选择：

- 代码存储库：Amazon CodeCommit或者第三方 Git 存储库，例如 GitHub 和 Bitbucket
- CI/CD 工作流程自动化：Amazon CodePipeline或 Jenkins
- 管道阶段：模型构建和培训、模型部署或两者兼而有

以下每个主题都对应于创建时可以选择的模板 SageMaker 项目。您可以在中查看可用的模板第 1 步：创建项目的项目演练.

主题

- 用于模型构建、培训和部署的 mLOPS 模板 (p. 2513)
- 用于模型构建、培训、部署和亚马逊的 mLOP 模板 SageMaker 模型监控器 (p. 2515)
- 用于映像构建、模型构建和模型部署的 mLOPS 模板 (p. 2516)
- 使用 CodePipeline 使用第三方 Git 仓库进行模型构建、培训和部署的 mLOPS 模板 (p. 2516)
- 使用 Jenkins 使用第三方 Git 仓库进行模型构建、培训和部署的 mLOPS 模板 (p. 2517)
- 更新 SageMaker 要使用第三方 Git 存储库的项目 (p. 2518)

用于模型构建、培训和部署的 mLOPS 模板

此模板是以下两个模板的组合，每个模板都可以独立使用，并包含这些模板中提供的所有资源。

- 代码存储库：Amazon CodeCommit
- CI/CD 工作流程自动化：Amazon CodePipeline

用于模型构建和培训的 mLOPS 模板

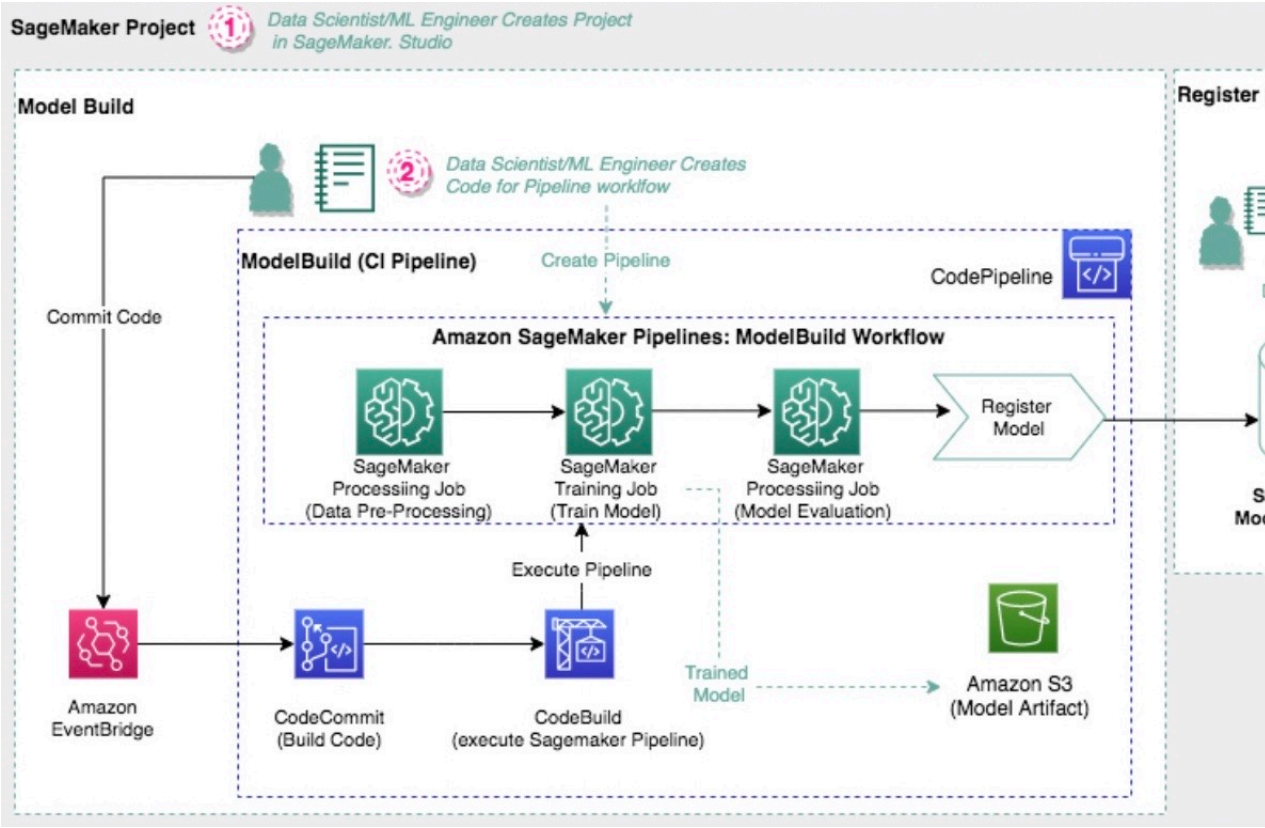
当您希望使用 mLOP 解决方案来处理数据、提取要素、训练和测试模型以及在 SageMaker 模型注册表。

此模板提供以下资源：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon CodeCommit包含创建亚马逊的示例代码的存储库 SageMaker 在 Python 代码中对构建管道管道进行模型并演示如何创建和更新 SageMaker 管道。此存储库还有一个 Python Jupyter 笔记本，您可以在 Studio 中打开和运行该笔记本。
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon CodePipeline具有源代码和构建步骤的管道。源步骤指向 CodeCommit 存储库。构建步骤从该存储库中获取代码，创建并更新 SageMaker 管道，启动管道执行，然后等待管道执行完成。

- 用于存储项的 Amazon S3 存储段，其中包括 CodePipeline 和 CodeBuild 工件，以及从 SageMaker 管道运行。

下列示意图说明了该工作流程和Amazon此模板用于帮助您构建和训练模型的资源。



模型部署的 mLOPS 模板

使用此模板在中自动部署模型 SageMaker 模型注册表 SageMaker 用于实时推理的终端节点。此模板可识别模型注册表中的更改。注册并批准新模型版本后，它会自动启动部署。

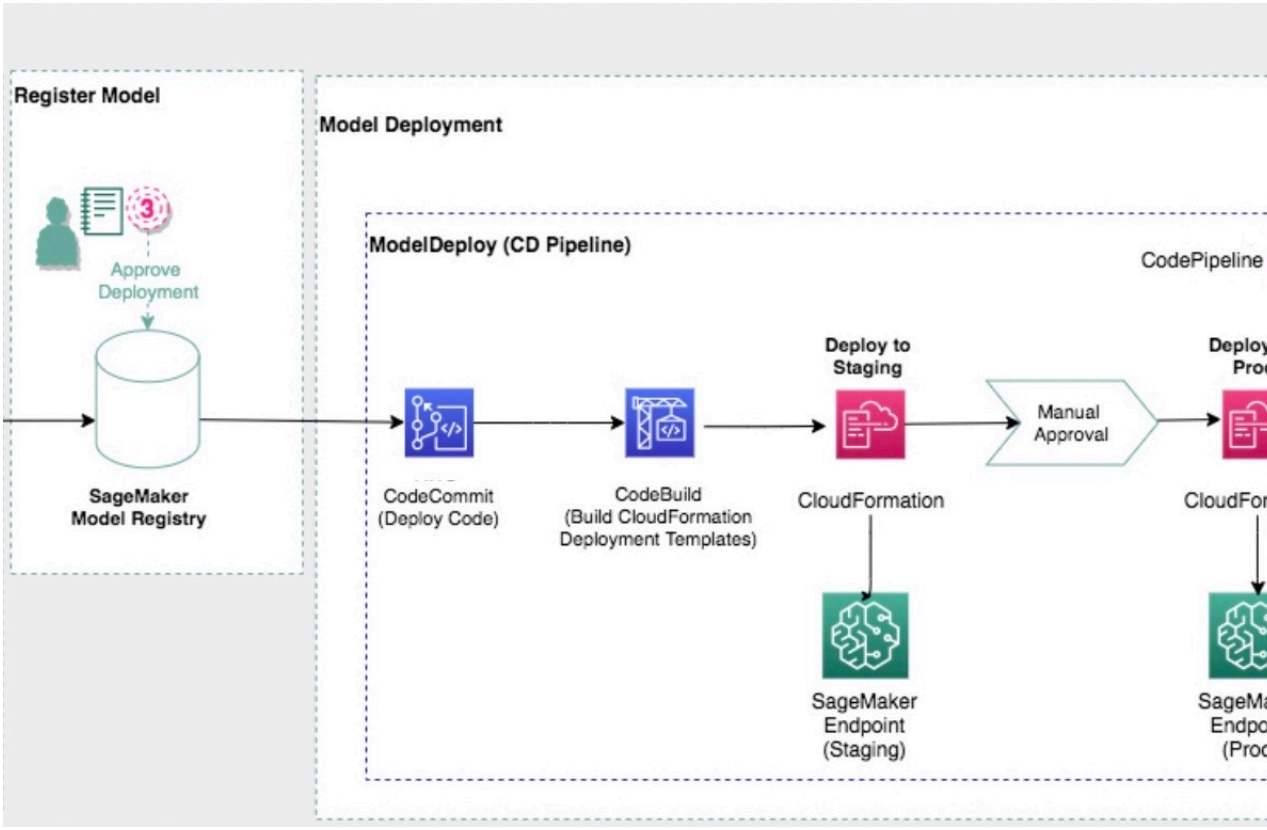
该模板规定了 CodeCommit 具有配置文件的存储库以指定模型部署步骤，Amazon CloudFormation将终端定义为基础设施的模板，以及用于测试终端节点的种子代码

此模板提供以下资源：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon CodeCommit存储库，其中包含将模型部署到暂存和生产环境中的终端节点的示例代码。
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon CodePipeline具有源代码、构建、部署到暂存的管道 deploy-to-production 步骤。源步骤指向 CodeCommit 存储库，构建步骤从该存储库中获取代码并生成 CloudFormation 要部署的堆栈。这些区域有： deploy-to-staging 和 deploy-to-production 步骤部署 CloudFormation 堆叠到各自的环境中。在暂存步骤和生产构建步骤之间有一个手动审批步骤，因此，在将模型部署到生产之前，必须先获得 MPLOP 工程师批准该模型。

  中的示例代码中还有一个程序化批准步骤，其中包含占位符测试 CodeCommit 存储库。您可以添加其他测试来替换占位符测试。
- 用于存储项的 Amazon S3 存储段，其中包括 CodePipeline 和 CodeBuild 工件，以及从 SageMaker 管道运行。
- 一个 CloudWatch 事件，当模型包版本被批准或拒绝时启动管道。

以下示意图说明了该工作流程和Amazon此模板用于帮助您部署模型的资源。



用于模型构建、培训、部署和亚马逊的 mLOP 模板 SageMaker 模型监控器

此模板是用于模型构建、培训和部署的 mLOPS 模板的扩展。它包括模板的模型构建、培训和部署组件，以及额外的亚马逊 SageMaker 模型监视器模板，提供以下类型的监控：

- 数据质量— 监控数据质量的漂移。
- 模型质量— 监控模型质量指标的偏移，例如准确性。
- 生产中模型的偏差漂移— 监控模型预测中的偏差。

- 代码存储库：AmazonCodeCommit
- CI/CD 工作流程自动化：AmazonCodePipeline

亚马逊的 mLOPS 模板 SageMaker 模型监控器

您可以将此模板用于 mLOPS 解决方案来部署一个或多个亚马逊 SageMaker 数据质量、模型质量、模型偏差和模型可解释性监视器，用于监控 SageMaker 推理端点。

此模板提供以下资源：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon CodeCommit 包含获取基线由来自 SageMaker 模型注册表，并更新模板的临时环境和生产环境的参数。它还包含Amazon CloudFormation用于创建亚马逊的模板 SageMaker 模型监控器。
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon CodePipeline 具有源代码、构建和部署步骤的管道。源步骤指向 CodePipeline 存储库。构建步骤从该存储库获取代码，从模型注册表获取基准，然后更新暂存和生产环境的模板参数。部署步骤会将配置的监视器部署到暂存环境和生产环境中。

手动批准步骤，在`DeployStaging`阶段，要求你验证生产 SageMaker 终端节点是`InService`在批准并移动到`DeployProd`阶段。

- 该模板使用由 mLOPS 模板创建的相同 S3 存储桶进行模型构建、培训和部署来存储监视器的输出。
- 两个亚马逊 EventBridge 活动规则启动亚马逊 SageMaker 模型监控器Amazon CodePipeline每次分期 SageMaker 终端节点已更新，或者将代码更改提交到 CodePipeline 存储库。

用于映像构建、模型构建和模型部署的 mLOPS 模板

此模板是用于模型构建、培训和部署的 mLOPS 模板 (p. 2513). 它包括该模板的模型构建、培训和部署组件以及以下选项：

- 包括处理图像 — 构建管道
- 包括培训图像 — 构建管道
- 包括推理图像 — 构建管道

对于在项目创建过程中选择的每个组件，使用模板创建以下组件：

- Amazon ECR 存储库
- SageMaker 映像
- 一个 CodeCommit 存储库中包含您可以自定义的 Dockerfile
- 一个 CodePipeline 这是由更改启动的 CodePipeline 知识库
- 一个 CodeBuild 构建 Docker 映像并将其注册到 Amazon ECR 存储库的项目
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的 EventBridge 启动 CodePipeline 按计划

当 CodePipeline 已启动，它会构建一个新的 Docker 容器并将其注册到 Amazon ECR 存储库中。在 Amazon ECR 存储库中注册新容器时，一个新的容器`ImageVersion`被添加到 SageMaker 映像。这将启动模型构建管道，而模型构建管道又启动了部署管道。

新创建的映像将在工作流的模型构建、培训和部署部分中使用（如果适用）。

使用 CodePipeline 使用第三方 Git 仓库进行模型构建、培训和部署的 mLOPS 模板

- 代码存储库：第三方 Git。建立 AWS CodeStar 从您的连接Amazon您的账户 GitHub 用户或组织。用钥匙添加标签`sagemaker`和值`true`加入此 AWS CodeStar 连接。
- CI/CD 工作流程自动化：Amazon CodePipeline

此模板提供以下资源：

- 与一个或多个客户指定的 Git 仓库关联。
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon CodePipeline具有源代码、构建、部署到暂存的管道 deploy-to-production 步骤。源步骤指向第三方 Git 存储库，构建步骤从该存储库中获取代码并生成 CloudFormation 要部署的堆栈。这些区域有： deploy-to-staging 和 deploy-to-production 步骤将 CloudFormation 堆栈部署到各自的环境中。在暂存步骤和生产构建步骤之间有一个手动审批步骤，因此，在将模型部署到生产之前，必须先获得 MPLOP 工程师批准该模型。
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon CodeBuild项目以使用种子代码信息填充 Git 仓库。这需要 AWS CodeStar 从您的连接Amazon账户转到 Git 仓库主机上的账户。
- 用于存储项的 Amazon S3 存储段，其中包括 CodePipeline 和 CodeBuild 工件，以及从 SageMaker 管道运行。

使用 Jenkins 使用第三方 Git 仓库进行模型构建、培训和部署的 mLOPS 模板

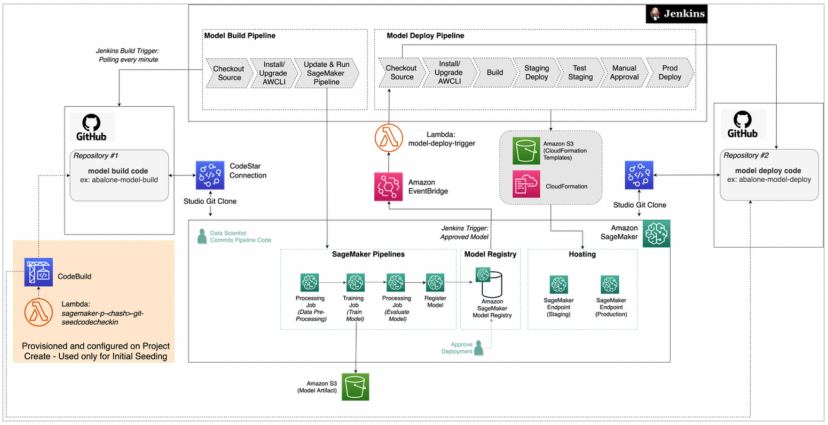
- 代码存储库：第三方 Git。建立 AWS CodeStar 从您的连接Amazon您的账户 GitHub 用户或组织。用钥匙添加标签`sagemaker`和价值`true`加入此 AWS CodeStar 连接。
- CI/CD 工作流程自动化：Jenkins

此模板提供以下资源：

- 与一个或多个客户指定的 Git 仓库关联。
- 用于生成具有源代码、构建、部署到暂存和的 Jenkins 管道的种子代码 deploy-to-production 步骤。源步骤指向客户指定的 Git 存储库。构建步骤从该存储库中获取代码并生成两个 CloudFormation 堆栈。部署步骤部署 CloudFormation 堆叠到各自的环境中。分段步骤和生产步骤之间有一个批准步骤。
- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的Amazon CodeBuild项目以使用种子代码信息填充 Git 仓库。这需要 AWS CodeStar 从您的连接Amazon账户转到 Git 仓库主机上的账户。
- 用于存储对象的 Amazon S3 存储段 SageMaker 项目和 SageMaker 管道。

模板在项目和源代码管理存储库之间创建关联，但是您需要执行额外的手动步骤才能在您的Amazon账户和 Jenkins。有关详细步骤，请参阅博客文章。创建 Amazon SageMaker 使用第三方源码控制和 Jenkins 的项目.

这些说明可帮助您构建下图所示的体系结构，在本示例中，Github 作为源代码管理存储库。如图所示，您正在将 Git 存储库附加到项目以签入和管理代码版本。Jenkins 在检测到 Git 存储库中模型构建代码的更改时启动模型构建管道。您还要将项目连接到 Jenkins 以编排模型部署步骤，这些步骤从您批准在模型注册表中注册的模型时开始，或者 Jenkins 检测到模型部署代码的更改。



总之，这些步骤将引导您完成以下任务：

1. 在你的之间建立连接Amazon和 Github 账户。
2. 创建 Jenkins 帐户并导入所需的插件。
3. 创建 Jenkins IAM 用户和权限策略。
4. 设置AmazonJenkins 服务器上 Jenkins IAM 用户的凭证。
5. 创建 API 令牌以便与 Jenkins 服务器进行通信。
6. 使用 CloudFormation 用于设置的模板 EventBridge 规则以监控模型注册表中是否有新批准的模型。
7. 创建 SageMaker 项目，它使用模型构建和部署代码为 Github 仓库种子。
8. 使用模型构建种子代码创建 Jenkins 模型构建管道。
9. 使用模型部署种子代码创建 Jenkins 模型部署管道。

更新 SageMaker 要使用第三方 Git 存储库的项目

附加到的托管策略`AmazonSageMakerServiceCatalogProductsUseRole`角色于 2021 年 7 月 27 日更新，以便与第三方 Git 模板一起使用。加入亚马逊的用户 SageMaker 此日期之后的 Studio 并启用项目模板使用新策略。在此日期之前加入的用户必须更新策略才能使用这些模板。使用以下选项之一更新策略：

- 删除角色并切换 Studio 设置
  1. 在 IAM 控制台中，删除`AmazonSageMakerServiceCatalogProductsUseRole`.
  2. 在 Studio 控制面板中，选择编辑设置.
  3. 切换两个设置然后选择提交.
- 在 IAM 控制台中，将以下权限添加至`AmazonSageMakerServiceCatalogProductsUseRole`：

```
{
      "Effect": "Allow",
      "Action": [
          "codestar-connections:UseConnection"
      ],
      "Resource": "arn:aws:codestar-connections:*:*:connection/*",
      "Condition": {
          "StringEqualsIgnoreCase": {
              "aws:ResourceTag/sagemaker": "true"
          }
      }
  },
  {
      "Effect": "Allow",
      "Action": [
          "s3:PutObjectAcl"
      ],
      "Resource": [
          "arn:aws:s3:::sagemaker-*"
      ]
  }
```

创建自定义项目模板

如果 SageMaker 提供的模板不符合您的需求（例如，您希望在 CodePipeline 通过多个阶段或自定义审批步骤），创建自己的模板。

我们建议首先使用 SageMaker 提供的模板，了解如何组织代码和资源并在其基础上进行构建。要执行此操作，在启用管理员访问 SageMaker 模板，请登录https://console.aws.amazon.com/servicecatalog/，选择产品组合选择，然后选择导入. 有关的信息Amazon Service Catalog，请参阅概述Amazon Service Catalog中的Amazon Service Catalog用户指南.

创建自己的项目模板以自定义你的 mLOPS 项目。 SageMaker 项目模板是Amazon Service Catalog— 预配置产品以为您的 mLOPS 项目配置资源。

要创建自定义项目模板，请完成以下步骤。

1. 创建产品组合。想要了解有关信息，请参阅步骤 3: 创建Amazon Service Catalog产品组合.
2. 创建产品。产品是 CloudFormation 模板。您可以创建产品的多个版本。想要了解有关信息，请参阅步骤 4: 创建Amazon Service Catalog产品.

   对于要使用的产品 SageMaker 项目中，将以下参数添加到您的产品模板中。

```
SageMakerProjectName:
Type: String
Description: Name of the project
```

```
SageMakerProjectId:
Type: String
Description: Service generated Id of the project.
```

Important

建议您将 CodeCommit 存储库中 SageMaker 在 VPC 模式下可见项目仓库的代码库。以下代码示例中显示了示例模板和所需添加内容。
原始（示例）模板：

```
ModelBuildCodeCommitRepository:
    Type: AWS::CodeCommit::Repository
    Properties:
      # Max allowed length: 100 chars
      RepositoryName: !Sub sagemaker-${SageMakerProjectName}-
${SageMakerProjectId}-modelbuild # max: 10+33+15+10=68
      RepositoryDescription: !Sub SageMaker Model building workflow
 infrastructure as code for the Project ${SageMakerProjectName}
      Code:
        S3:
          Bucket: SEEDCODE_BUCKETNAME
          Key: toolchain/model-building-workflow-v1.0.zip
        BranchName: main
```

要在 VPC 模式下添加的其他内容：

```
SageMakerRepository:
    Type: AWS::SageMaker::CodeRepository
    Properties:
        GitConfig:
            RepositoryUrl: !GetAtt ModelBuildCodeCommitRepository.CloneUrlHttp
            Branch: main
```

3. 添加启动约束。启动约束指定了 IAM 角色Amazon Service Catalog假设用户何时启动产品。想要了解有关信息，请参阅步骤 6：添加启动约束以分配 IAM 角色.
4. 在上预配产品https://console.aws.amazon.com/servicecatalog/来测试模板。如果您对模板感到满意，请继续下一步，使模板在 Studio 中可用。
5. 授予对Amazon Service Catalog您在 Studio 执行角色的步骤 1 中创建的产品组合。使用 Studio 域执行角色或具有 Studio 访问权限的用户角色。有关将角色添加到产品组合的更多信息，请参阅。步骤 7：向最终用户授予产品组合的访问.
6. 要使项目模板可用于组织模板在 Studio 中列出，创建一个标签，其中包含以下键和值Amazon Service Catalog您在步骤 2 中创建的产品。

   - 密钥：`sagemaker:studio-visibility`
   - 值：`true`

完成这些步骤后，组织中的 Studio 用户可以使用您创建的模板创建项目，方法是按照使用亚马逊创建 mLOPS 项目 SageMaker 工作室 (p. 2512)然后选择组织模板当您选择模板时。

查看项目资源

创建某个项目后，请在亚马逊中查看与项目关联的资源。 SageMaker Studio。

在 Studio 中创建项目

1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。

2. 选择SageMaker 资源，然后选择项目.
3. 双击要查看其详细信息的项目的名称。

   将出现包含项目详细信息的选项卡。

在项目详细信息选项卡上，您可以查看与项目关联的以下实体。

- 存储库：与此项目关联的代码存储库（存储库）。如果在创建项目时使用 SageMaker 提供的模板，它会创建Amazon CodeCommit回购或第三方 Git 仓库。有关 CodeCommit 的更多信息，请参阅是什么 Amazon CodeCommit.
- 管道 SageMaker 机器学习管道，用于定义准备数据、训练和部署模型的步骤。有关的信息 SageMaker ML 管道，请参阅创建和管理 SageMaker 管道 (p. 2478).
- 实验：一个或多个亚马逊 SageMaker 与项目关联的自动驾驶仪实验。有关 Autopilot 的信息，请参阅。使用亚马逊自动开发模型 SageMaker Autopilot (p. 170).
- 模型组：在项目中通过管道执行创建的模型版本组。有关模型组的信息，请参阅创建模型组 (p. 2254).
- 终端节点： SageMaker 终端节点托管已部署模型进行实时推理。批准模型版本后，它将部署到终端节点。
- 设置：项目的设置。这包括项目的名称和描述、有关项目模板的信息以及`SourceModelPackageGroupName`，以及有关项目的元数据。

更新亚马逊中的 mLOPS 项目 SageMaker 工作室

此过程演示如何在亚马逊中更新 MPops 项目。 SageMaker Studio。您可以更新说明、模板版本和模板参数。

先决条件

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的IAM Identity Center用于登录到 Studio 的 IAM 账户。有关信息，请参阅注册到 Amazon SageMaker 域 (p. 32)。
- 基本熟悉 Studio 用户界面。有关信息，请参阅亚马逊 SageMaker Studio UI 概述 (p. 77)。
- 将以下自定义内联策略添加到指定的角色：

  用户创建的角色`AmazonSageMakerFullAccess`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "servicecatalog:CreateProvisionedProductPlan",
                "servicecatalog:DescribeProvisionedProductPlan",
                "servicecatalog:DeleteProvisionedProductPlan"
            ],
            "Resource": "*"
        }
    ]
}
```

  `AmazonSageMakerServiceCatalogProductsLaunchRole`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:DescribeChangeSet"
            ],
            "Resource": "arn:aws:cloudformation:*:*:stack/SC-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:PutRepositoryTriggers"
            ],
            "Resource": "arn:aws:codecommit:*:*:sagemaker-*"
        }
    ]
}
```

更新 Studio 中的项目

1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
2. 在 Studio 边栏中，选择SageMaker 资源（图标）。
3. Select项目从下拉列表中选择。

   此时将显示您的项目列表。
4. 您可以打开更新项目通过以下方式之一对话框：

   a. 你可以从项目列表中打开它。

      右键单击目标项目，然后选择更新从下拉列表中选择。

   b. 你可以从项目选项卡中打开它。

      i. 双击项目列表中的项目。
      ii. 选择更新来自 的操作项目选项卡右上角的菜单。
5. 在更新项目对话框中，您可以编辑说明、模板版本和模板参数。
6. 选择显示差异.

   将显示一个对话框，显示原始和更新的项目设置。项目设置的任何更改都可以修改或删除当前项目中的资源。该对话框还会显示这些更改。
7. 您可能需要等待几分钟来获得更新按钮变为活动状态。选择 Update（更新）。
8. 项目更新可能需要几分钟才能完成。Select设置在项目选项卡中，并确保参数已正确更新。

使用亚马逊删除 mLOPS 项目 SageMaker 工作室

此过程演示如何使用亚马逊删除 mLOPS 项目。 SageMaker Studio。

先决条件

> Note
>
> 您只能删除在 Studio 中创建的项目。此条件是服务目录权限的一部分。`servicecatalog:TerminateProvisionedProduct`中的`AmazonSageMakerFullAccess`政策。如果需要，您可以更新此策略以删除此条件。

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的IAM Identity Center用于登录到 Studio 的 IAM 账户。有关信息，请参阅注册到 Amazon SageMaker 域 (p. 32)。

- 基本熟悉 Studio 用户界面。有关信息，请参阅亚马逊 SageMaker Studio UI 概述 (p. 77)。

删除 Amazon 中的项目 SageMaker 工作室

1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
2. 在 Studio 边栏中，选择SageMaker 资源（图标）。
3. Select项目从下拉列表中选择。
4. 从下拉列表中选择目标项目。如果你没看到你的项目，请键入项目名称并应用筛选器来查找你的项目。
5. 您可以通过以下方式之一删除 Studio 项目：

   a. 您可以从项目列表中删除项目。

      右键单击目标项目，然后选择Delete从下拉列表中选择。

      Note

      Studio 版本 v3.17.1 或更高版本支持此功能。有关更多信息，请参阅 关闭并更新 SageMaker 工作室 (p. 129)。

   b. 您可以通过项目详情部分。

      i. 找到项目后，双击该项目以查看其详细信息。
      ii. 选择Delete来自 的操作菜单。
6. 通过选择确认您的选择Delete来自 的删除项目窗口。

SageMaker mLOPS 项目演练

本演练演示了如何使用 mLOPS 项目创建 CI/CD 系统以构建、训练和部署模型。

先决条件

要完成本演练，您需要：

- 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的IAM Identity Center用于登录到 Studio 的 IAM 账户。有关信息，请参阅注册到 Amazon SageMaker 域 (p. 32)。
- 允许使用 SageMaker 提供的项目模板。有关信息，请参阅使用项目所需的 SageMaker Studio 权限 (p. 2511)。
- 基本熟悉 Studio 用户界面。有关信息，请参阅亚马逊 SageMaker Studio UI 概述 (p. 77)。

主题

- 第 1 步：创建 项目 (p. 2522)
- 第 2 步：克隆代码存储库 (p. 2523)
- 第 3 步：对守则进行修改 (p. 2523)
- 第 4 步：批准模型 (p. 2526)
- （可选）步骤 5：将模型版本部署到生产 (p. 2526)
- 第 6 步：清理资源 (p. 2526)

第 1 步：创建 项目

在此步骤中，您将创建一个 SageMaker 使用 SageMaker 提供的项目模板构建、训练和部署模型。

创建 SageMaker MPLOPS 项目

1. 登录到 Studio。有关更多信息，请参阅 注册到 Amazon SageMaker 域 (p. 32)。
2. 选择SageMaker 资源，然后选择项目从下拉列表中选择。
3. 请选择 Create project（创建项目）。

   这些区域有：创建项目将显示选项卡。
4. 适用于SageMaker 项目模板，选择组织模板选择，然后选择用于模型构建、培训和部署的 mLOPS 模板.
5. 适用于项目详情中，输入项目的名称和描述。

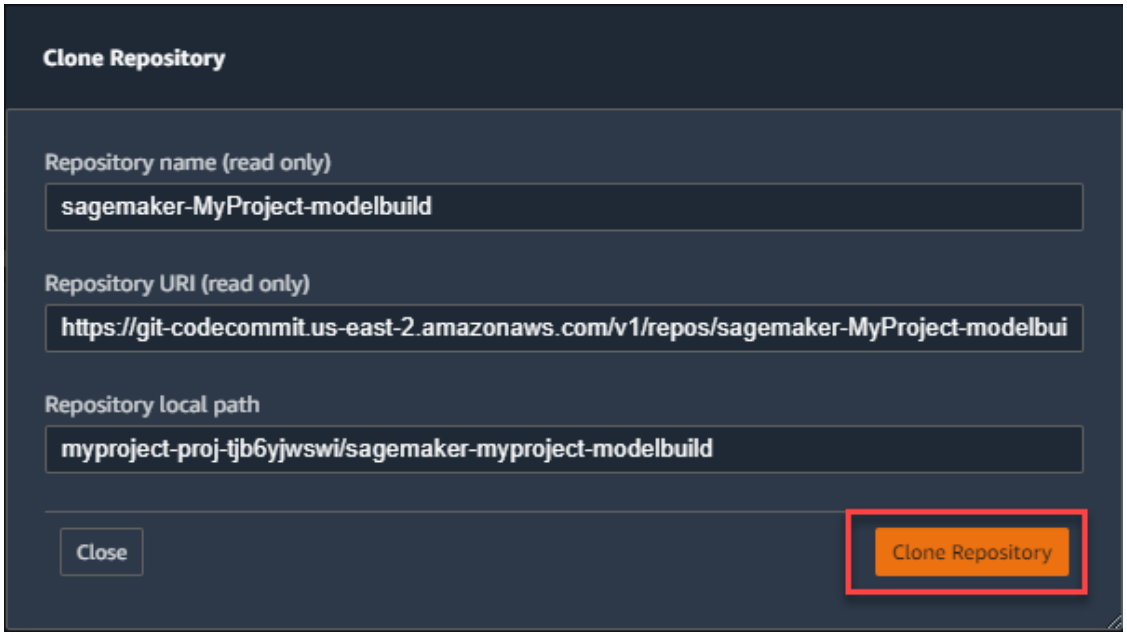
当项目出现在项目列出状态的已创建，请继续执行下一步。

第 2 步：克隆代码存储库

在创建项目后，两个 CodeCommit 存储库是在项目中创建的。其中一个存储库包含用于构建和训练模型的代码，另一个存储库包含用于部署模型的代码。在此步骤中，您将存储库克隆到包含构建代码的本地 SageMaker 项目中，并将模型训练到本地 Studio 环境，以便您可以使用代码。

克隆代码存储库

1. 选择SageMaker 资源，然后选择项目从下拉列表中选择。
2. 找到您在上一步中创建的项目的名称，然后双击该名称以打开项目的项目选项卡。
3. 在项目选项卡上，选择存储库，然后在本地路径以结尾的存储库的列模型构建，选择克隆存储库....
4. 在出现的对话框中，接受默认值并选择克隆存储库.



   当仓库的克隆完成后，本地路径将显示在本地路径column. 选择路径以打开包含 Studio 中存储库代码的本地文件夹。

第 3 步：对守则进行修改

现在对构建模型的管道代码进行更改并签入更改以启动新的管道运行。管道运行注册了一个新的模型版本。

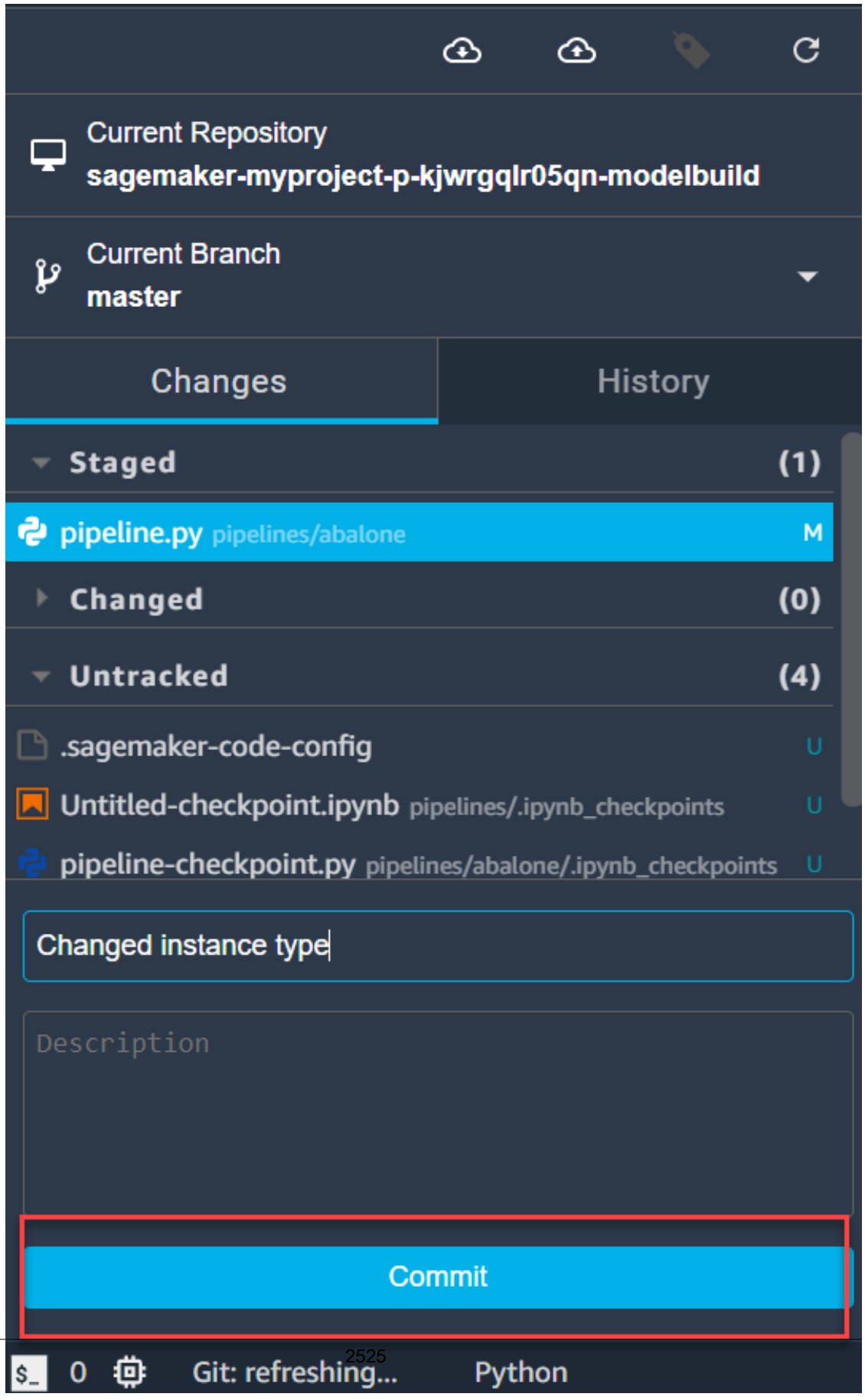
要更改代码

1. 在 Studio 中，选择文件浏览器图标 ( )，然后导航到`pipelines/abalonefolder`。双击`pipeline.py`打开代码文件。
2. 在`pipeline.py`文件中，找到设置训练实例类型的行。

```
training_instance_type = ParameterString(
        name="TrainingInstanceType", default_value="ml.m5.xlarge"
```

变更`ml.m5.xlarge`到`ml.m5.large`，然后键入`Ctrl+S`以保存更改。

3. 选择饭桶（图标 ）。转移、提交和推送更改`pipeline.py`. 有关在 Studio 中使用 Git 的信息，请参阅在中克隆 Git 存储库 SageMaker 工作室 (p. 125).


Current Repository
sagemaker-myproject-p-kjwrgqlr05qn-modelbuild
Current Branch
master
Changes
History
Staged (1)
pipeline.py pipelines/abalone M
Changed (0)
Untracked (4)
.sagemaker-code-config U
Untitled-checkpoint.ipynb pipelines/.ipynb_checkpoints U
pipeline-checkpoint.py pipelines/abalone/.ipynb_checkpoints U
Changed instance type
Description
Commit
$_ 0 Git: refreshing... Python

推送代码更改后，mLOPS 系统会启动创建新模型版本的管道运行。在下一个步骤中，您将批准新模型版本将其部署到生产环境中。

第 4 步：批准模型

现在您批准在上一步中创建的新模型版本，以启动将模型版本部署到 SageMaker 终端节点。

批准模型版本

1. 选择SageMaker 资源，然后选择项目从下拉列表中选择。
2. 找到您在第一步中创建的项目的名称，然后双击该名称以打开项目的项目选项卡。
3. 在项目选项卡上，选择模型组，然后双击显示的模型组的名称。

   此时显示模型组选项卡。
4. 在模型组选项卡中，双击版本 2. 这些区域有：版本 2将打开选项卡。选择更新状态
5. 在模型中更新模型版本状态对话框中的状态下拉列表中，选择审批选择，然后选择更新状态.

   批准模型版本会导致 mLOPS 系统将模型部署到暂存中。要查看终端节点，请选择终端节点项目选项卡上的选项卡。

（可选）步骤 5：将模型版本部署到生产

现在，您可以将模型版本部署到生产环境中。

> Note
>
> 要完成此步骤，您需要成为 Studio 域中的管理员。如果您不是管理员，请跳过此步骤。

将模型版本部署到生产环境

1. 登录到 CodePipeline 控制台https://console.aws.amazon.com/codepipeline/
2. 选择管道，然后选择名称的管道sagemak--***ProjectName*-*Projectid***-模型部署其中，*ProjectName*是您的项目的名称，*Projectid*是你的项目的 ID。
3. 在部署暂存选择阶段，选择审核.
4. 在审核对话框中，选择审批.

   批准部署暂存阶段会导致 mLOPS 系统将模型部署到生产环境中。要查看终端节点，请选择终端节点Studio 中项目选项卡上的选项卡。

第 6 步：清理资源

要停止产生费用，请清理在本演练中创建的资源。为此，请完成以下步骤。

> Note
>
> 删除Amazon CloudFormation堆栈和 Amazon S3 存储桶，您需要成为 Studio 中的管理员。如果您不是管理员，请让管理员完成这些步骤。

1. 在 Studio 边栏中，选择SageMaker 资源（图标 ）。
2. 选择项目从下拉列表中选择。
3. 从下拉列表中选择目标项目。如果你没看到你的项目，请键入项目名称并应用筛选器来查找你的项目。

4. 您可以通过以下方式之一删除 Studio 项目：

   a. 您可以从项目列表中删除项目。

      右键单击目标项目，然后选择Delete从下拉列表中选择。

      Note

      Studio 版本 v3.17.1 或更高版本支持此功能。有关更多信息，请参阅 关闭并更新 SageMaker 工作室 (p. 129)。

   b. 您可以通过项目详情部分。

      i. 找到项目后，双击它以在主面板中查看其详细信息。

      ii. 选择Delete来自 的操作菜单。

5. 通过选择确认您的选择Delete来自 的删除项目窗口。

   这将删除Amazon Service Catalog项目创建的预配置产品。这包括 CodeCommit、CodePipeline 和 CodeBuild 为项目创建的资源。

6. 删除存储在设备上的Amazon CloudFormation项目创建的堆栈。有两个堆栈，一个用于暂存，一个用于生产。堆栈的名字是sagemak--***ProjectName-## ID***-部署-暂存和sagemak--***ProjectName-## ID***-deploy-prod其中，*ProjectName*是您的项目的名称，*## ID*是你的项目的 ID。

   有关如何删除的信息，请参阅Amazon CloudFormation堆栈，请参阅删除堆栈Amazon CloudFormation控制台中的Amazon CloudFormation用户指南.

7. 删除项目创建的 Amazon S3 存储桶。存储桶的名称是sagemaker-Project***## ID***其中，*## ID*是你的项目的 ID。

亚马逊 SageMaker ML 世系跟踪

亚马逊 SageMaker ML 世系跟踪创建并存储有关从数据准备到模型部署的机器学习 (ML) 工作流程步骤的信息。利用跟踪信息，您可以重现工作流步骤、跟踪模型和数据集血统，并建立模型治理和审计标准。

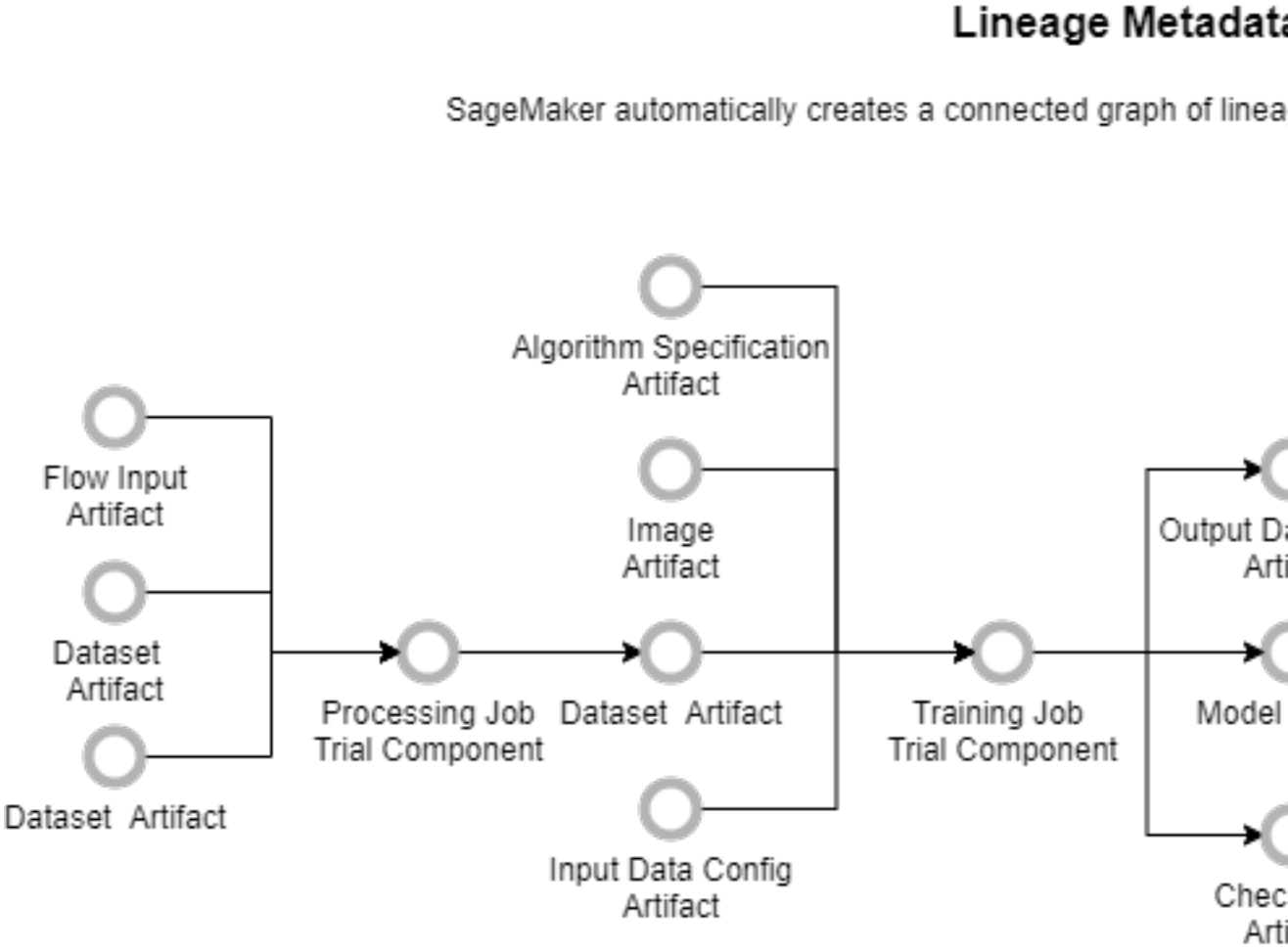
与 SageMaker 沿袭追踪数据科学家和模型构建者可以执行以下操作：

- 保持模型发现实验的运行历史。
- 通过跟踪模型世系伪造以进行审计和合规性验证来建立模型治理。

主题

- 谱系跟踪实体 (p. 2528)
- 亚马逊 SageMaker — 创建的追踪实体 (p. 2530)
- 手动创建跟踪实体 (p. 2531)
- 查询谱系实体 (p. 2534)
- 跨账户世系追踪 (p. 2540)

下图显示了亚马逊的世系图示例。 SageMaker 自动在 end-to-end 模型训练和部署 ML 工作流程。



谱系跟踪实体

跟踪实体维护您的所有元素的表示 end-to-end 机器学习工作流程。您可以使用此表示法来建立模型治理、重现工作流程以及维护工作历史记录。

亚马逊 SageMaker 在创建试验组件时自动为试验组件及其关联的试验和实验创建跟踪实体 SageMaker 处理作业、训练作业和批量转换作业等作业。除了 auto 跟踪之外，你还可以手动创建跟踪实体 (p. 2531)为您的工作流程中的自定义步骤建模 有关更多信息，请参阅 使用亚马逊管理 Machine Learning SageMaker 实验 (p. 1758)。

SageMaker 还会为工作流程中的其他步骤自动创建跟踪实体，以便您可以端到端跟踪工作流程。有关更多信息，请参阅 亚马逊 SageMaker — 创建的追踪实体 (p. 2530)。

您可以创建其他实体来补充 SageMaker 创建的实体。有关更多信息，请参阅 手动创建跟踪实体 (p. 2531)。

SageMaker 会重复使用任何现有实体，而不是创建新实体。例如，只能有一个具有唯一性的工件`SourceUri`.

查询世系的关键概念

- 血统— 用于跟踪机器学习工作流中各个实体之间关系的元数据。
- QueryLineage— 检查你的血统和发现实体之间关系的行动。

- 谱系实体— 你的世系所构成的元数据元素。
- 跨账户谱系— 您的机器学习工作流程可能跨越多个账户。借助跨账户世系，您可以配置多个账户，以便在共享实体资源之间自动创建世系关联。 QueryLineage 那么甚至可以从这些共享账户中返回实体。

定义了以下跟踪实体：

实验实体

- 试验组件— 机器学习试用的阶段。包括处理作业、训练作业和批量转换作业。
- 试用— 通常生成模型的试验组件的组合。
- 试验— 一组试验通常专注于解决特定使用案例。

谱系实体

- 试验组件— 代表世系中的处理、训练和转换工作。也是实验管理的一部分。
- 上下文— 提供其他跟踪或实验实体的逻辑分组。从概念上讲，实验和试验都是上下文。一些示例是终端节点和模型包。
- 操作— 表示操作或活动。通常，操作至少涉及一个输入工件或输出工件。一些示例是工作流程步骤和模型部署。
- Artifact— 表示 URI 可寻址的对象或数据。工件通常是试验组件或操作的输入或输出。一些示例包括数据集（S3 存储桶 URI）或映像（Amazon ECR 注册表路径）。
- 协会— 链接其他跟踪或实验实体，例如训练数据位置与培训作业之间的关联。

  关联有可选的`AssociationType`财产。提供以下值以及每种类型的建议用途。 SageMaker 对它们的使用没有限制：
  - `ContributedTo`— 来源为目的地做出了贡献或参与启用目的地。例如，培训数据为培训工作做出了贡献。
  - `AssociatedWith`— 源已连接到目的地。例如，批准工作流程与模型部署相关联。
  - `DerivedFrom`-目的地是对来源的修改。例如，处理作业的频道输入的摘要输出源于原始输入。
  - `Produced`— 源生成了目的地。例如，培训作业生成了模型工件。
  - `SameAs`— 当在不同账户中使用同一个世系实体时。

通用属性

- 输入属性

  操作、对象和上下文实体具有类型财产，`ActionType`、`ArtifactType`, 和`ContextType`，分别为。此属性是一个自定义字符串，它可以将有意义的信息与实体关联起来，并用作 List API 中的过滤器。
- 源属性

  操作、对象和上下文实体具有`Source`财产。此属性提供实体所代表的底层 URI。一些示例包括：
  - 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`UpdateEndpoint`来源所在的操作`EndpointArn`.
  - 处理作业的图像工件，其中源是`ImageUri`.
  - 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`Endpoint`来源所在的上下文`EndpointArn`.
- 元数据属性

  动作和工件实体有可选的`Metadata`属性，该属性可提供以下信息：
  - `ProjectId`— 例如，的 ID SageMaker 模型所属的 mLOPS 项目。
  - `GeneratedBy`— 例如， SageMaker 注册模型包版本的管道执行。

- `Repository`— 例如，包含算法的存储库。
- `CommitId`— 例如，算法版本的提交 ID。

亚马逊 SageMaker — 创建的追踪实体

亚马逊 SageMaker 自动创建跟踪实体 SageMaker 作业、模型、模型包和端点（如果数据可用）。SageMaker 创建的世系实体的数量没有限制。

有关如何手动创建跟踪实体的信息，请参阅。手动创建跟踪实体 (p. 2531).

主题

- 跟踪实体 SageMaker 作业 (p. 2530)
- 跟踪模型包的实体 (p. 2530)
- 跟踪终端节点的实体 (p. 2531)

跟踪实体 SageMaker 作业

SageMaker 为每个组件创建一个试用组件并与之关联 SageMaker 任务。 SageMaker 创建工件以跟踪作业元数据以及每个工件与作业之间的关联。

对象是为以下作业属性创建 ARN，并与 SageMaker 任务。Artifact`SourceUri`在括号中列出。

训练作业

- 包含训练算法的图像 (`TrainingImage`)。
- 每个输入通道的数据源 (`S3Uri`)。
- 模型的位置 (`S3OutputPath`).
- 托管点检查点数据的位置 (`S3Uri`)。

处理 Job

- 要由处理作业运行的容器 (`ImageUri`)。
- 每个处理输入和处理输出的数据位置 (`S3Uri`)。

转换 Job

- 要转换的输入数据源 (`S3Uri`)。
- 转换的结果 (`S3OutputPath`)。

> **Note**
>
> 例如，根据提供给创建 API 的 Amazon S3 URI 值对 Amazon Simple Storage Service (Amazon S3) 项目进行跟踪。CreateTrainingJob，而不是 Amazon S3 密钥以及每个文件的哈希值或 etag 值。

跟踪模型包的实体

将创建以下实体：

模型包

- 每个模型包组的上下文。

- 每个模型包的工件。
- 每个模型包对象与包所属的每个模型包组的上下文之间的关联。
- 创建模型包版本的操作。
- 模型包对象和创建操作之间的关联。
- 模型包对象与包所属的每个模型包组上下文之间的关联。
- 推理容器
  - 模型包中定义的每个容器中使用的图像的工件。
  - 每个容器中使用的模型的工件。
  - 每个工件和模型包工件之间的关联。
- 算法
  - 模型包中定义的每个算法的工件。
  - 每个算法创建的模型的工件。
  - 每个工件和模型包工件之间的关联。

跟踪终端节点的实体

Amazon SageMaker 创建了以下实体：

端点

- 每个终端节点的上下文
- 用于创建每个终端节点的模型部署的操作
- 部署到端点的每个模型的工件
- 模型中使用的图像的工件
- 模型的模型包的工件
- 部署到终端节点的每个映像的工件
- 每个工件和模型部署操作之间的关联

手动创建跟踪实体

您可以为任何属性手动创建跟踪实体。有关亚马逊的追踪实体的信息 SageMaker 自动创建，请参阅亚马逊 SageMaker — 创建的追踪实体 (p. 2530).

您可以将标签添加至除关联之外的所有实体。标签是提供自定义信息的任意键值对。您可以按标签对列表或搜索查询进行筛选或排序。有关更多信息，请参阅《Amazon 一般参考》中的标记Amazon资源。

有关演示如何创建世系实体的示例笔记本，请参阅Amazon SageMaker 世系中的笔记本亚马逊 SageMaker 示例 GitHub 知识库.

主题

- 手动创建实体 (p. 2531)
- 手动跟踪工作流 (p. 2533)
- 限制 (p. 2534)

手动创建实体

以下步骤介绍了如何在 SageMaker 训练作业和终端节点之间创建和关联构件。请执行下列步骤：

导入跟踪实体和关联

1. 导入世系追踪实体。

```
import sys
!{sys.executable} -m pip install -q sagemaker

from sagemaker import get_execution_role
from sagemaker.session import Session
from sagemaker.lineage import context, artifact, association, action

import boto3
boto_session = boto3.Session(region_name=region)
sagemaker_client = boto_session.client("sagemaker")
```

2. 创建输入和输出项目。

```
code_location_arn = artifact.Artifact.create(
    artifact_name='source-code-location',
    source_uri='s3://...',
    artifact_type='code-location'
).artifact_arn

# Similar constructs for train_data_location_arn and test_data_location_arn

model_location_arn = artifact.Artifact.create(
    artifact_name='model-location',
    source_uri='s3://...',
    artifact_type='model-location'
).artifact_arn
```

3. 训练模型并获取`trial_component_arn`这代表培训作业。
4. 将输入项目和输出工件与训练作业（试用组件）关联。

```
input_artifacts = [code_location_arn, train_data_location_arn, test_data_location_arn]
for artifact_arn in input_artifacts:
    try:
        association.Association.create(
            source_arn=artifact_arn,
            destination_arn=trial_component_arn,
            association_type='ContributedTo'
        )
    except:
        logging.info('association between {} and {} already exists', artifact_arn,
 trial_component_arn)

output_artifacts = [model_location_arn]
for artifact_arn in output_artifacts:
    try:
         association.Association.create(
            source_arn=trial_component_arn,
            destination_arn=artifact_arn,
            association_type='Produced'
        )
    except:
        logging.info('association between {} and {} already exists', artifact_arn,
 trial_component_arn)
```

5. 创建推理端点。

```
predictor = mnist_estimator.deploy(initial_instance_count=1,
                                   instance_type='ml.m4.xlarge')
```

6. 创建终端节点上下文。

```
from sagemaker.lineage import context

endpoint = sagemaker_client.describe_endpoint(EndpointName=predictor.endpoint_name)
endpoint_arn = endpoint['EndpointArn']

endpoint_context_arn = context.Context.create(
    context_name=predictor.endpoint_name,
    context_type='Endpoint',
    source_uri=endpoint_arn
).context_arn
```

7. 将培训作业（试用组件）和终端上下文关联起来。

```
association.Association.create(
    source_arn=trial_component_arn,
    destination_arn=endpoint_context_arn
)
```

手动跟踪工作流

您可以手动跟踪在上一部分中创建的工作流。

鉴于上一示例中的终端节点 Amazon Resource Name (ARN)，以下过程向您展示了如何跟踪工作流程回到用于训练部署到终端节点的模型的数据集。请执行下列步骤：

跟踪从端点到培训数据源的工作流

1. 导入跟踪实体。

```
import sys
!{sys.executable} -m pip install -q sagemaker

from sagemaker import get_execution_role
from sagemaker.session import Session
from sagemaker.lineage import context, artifact, association, action

import boto3
boto_session = boto3.Session(region_name=region)
sagemaker_client = boto_session.client("sagemaker")
```

2. 从终端节点 ARN 获取终端节点上下文。

```
endpoint_context_arn = sagemaker_client.list_contexts(
    SourceUri=endpoint_arn)['ContextSummaries'][0]['ContextArn']
```

3. 从试用组件和终端上下文之间的关联中获取试用组件。

```
trial_component_arn = sagemaker_client.list_associations(
    DestinationArn=endpoint_context_arn)['AssociationSummaries'][0]['SourceArn']
```

4. 从试用组件和终端上下文之间的关联中获取训练数据位置工件。

```
train_data_location_artifact_arn = sagemaker_client.list_associations(
    DestinationArn=trial_component_arn, SourceType='Model')['AssociationSummaries'][0]
['SourceArn']
```

5. 从训练数据位置工件中获取训练数据位置。

```
train_data_location = sagemaker_client.describe_artifact(
    ArtifactArn=train_data_location_artifact_arn)['Source']['SourceUri']
    print(train_data_location)
```

响应:

```
s3://sagemaker-sample-data-us-east-2/mxnet/mnist/train
```

限制

您可以在任何实体、实验和世系之间创建关联，但以下情况除外：

- 不能在两个实验实体之间创建关联。实验实体由实验、试验和试验组件组成。
- 你可以创建与另一个关联的关联。

如果您尝试创建已存在的实体，则会出错。

手动创建的世系实体的最大数量

- 操作: 3000
- ：Artifacts 6000
- 关联：6000
- 上下文：500

Amazon SageMaker 自动创建的世系实体的数量没有限制。

查询谱系实体

亚马逊 SageMaker 在使用世系实体时自动生成它们的图形。你可以查询这些数据来回答各种问题。例如，您可以查询您的世系实体，以便：

- 检索创建模型的所有数据集。
- 检索创建终端节点的所有作业。
- 检索所有使用数据集的模型。
- 检索使用模型的所有终端节点。
- 检索从某个数据集派生的终端节点。
- 检索创建培训作业的管道执行。
- 检索实体之间的关系，以进行调查、治理和可重复性。
- 探索你的血统以实现可视化。

主题

- 入门查询世系实体 (p. 2535)

限制

- 在以下区域中不可用世系查询：
  - 亚太地区（大阪）– ap-northeast-3

  - 欧洲（米兰）— eu-Sain1
  - 非洲（开普敦）—
- 目前要发现的最大关系深度限制为 10。
- 筛选限于以下属性：上次修改日期、创建日期、类型和世系实体类型。

入门查询世系实体

开始使用的最简单方式是：

- 亚马逊 SageMaker 适用于 Python 的开发工具包它定义了许多常见的使用案例。
- 对于演示如何使用的笔记本 SageMaker 世系 API 用于查询世系图中的关系，请参阅sagemaker-line-多跳查询 .ipynb.

以下示例显示了如何使用`LineageQuery`和`LineageFilter`API 用于构建查询以回答有关世系图的问题，并为少数用例提取实体关系。

Example 使用**`LineageQuery`**查找实体关联的 API

```
from sagemaker.lineage.context import Context, EndpointContext
from sagemaker.lineage.action import Action
from sagemaker.lineage.association import Association
from sagemaker.lineage.artifact import Artifact, ModelArtifact, DatasetArtifact

from sagemaker.lineage.query import (
    LineageQuery,
    LineageFilter,
    LineageSourceEnum,
    LineageEntityEnum,
    LineageQueryDirectionEnum,
)
# Find the endpoint context and model artifact that should be used for the lineage queries.

contexts = Context.list(source_uri=endpoint_arn)
context_name = list(contexts)[0].context_name
endpoint_context = EndpointContext.load(context_name=context_name)
```

Example 查找与终端节点关联的所有数据集

```
# Define the LineageFilter to look for entities of type `ARTIFACT` and the source of type
 `DATASET`.

query_filter = LineageFilter(
    entities=[LineageEntityEnum.ARTIFACT], sources=[LineageSourceEnum.DATASET]
)

# Providing this `LineageFilter` to the `LineageQuery` constructs a query that traverses
 through the given context `endpoint_context`
# and find all datasets.

query_result = LineageQuery(sagemaker_session).query(
    start_arns=[endpoint_context.context_arn],
    query_filter=query_filter,
    direction=LineageQueryDirectionEnum.ASCENDANTS,
    include_edges=False,
)

# Parse through the query results to get the lineage objects corresponding to the datasets
dataset_artifacts = []
```

```
for vertex in query_result.vertices:
    dataset_artifacts.append(vertex.to_lineage_object().source.source_uri)

pp.pprint(dataset_artifacts)
```

Example 查找与端点关联的模型

```
# Define the LineageFilter to look for entities of type `ARTIFACT` and the source of type
 `MODEL`.

query_filter = LineageFilter(
    entities=[LineageEntityEnum.ARTIFACT], sources=[LineageSourceEnum.MODEL]
)

# Providing this `LineageFilter` to the `LineageQuery` constructs a query that traverses
 through the given context `endpoint_context`
# and find all datasets.

query_result = LineageQuery(sagemaker_session).query(
    start_arns=[endpoint_context.context_arn],
    query_filter=query_filter,
    direction=LineageQueryDirectionEnum.ASCENDANTS,
    include_edges=False,
)

# Parse through the query results to get the lineage objects corresponding to the model
model_artifacts = []
for vertex in query_result.vertices:
    model_artifacts.append(vertex.to_lineage_object().source.source_uri)

# The results of the `LineageQuery` API call return the ARN of the model deployed to the
 endpoint along with
# the S3 URI to the model.tar.gz file associated with the model
pp.pprint(model_artifacts)
```

Example 查找与终端节点关联的试用组件

```
# Define the LineageFilter to look for entities of type `TRIAL_COMPONENT` and the source of
 type `TRAINING_JOB`.

query_filter = LineageFilter(
    entities=[LineageEntityEnum.TRIAL_COMPONENT],
    sources=[LineageSourceEnum.TRAINING_JOB],
)

# Providing this `LineageFilter` to the `LineageQuery` constructs a query that traverses
 through the given context `endpoint_context`
# and find all datasets.

query_result = LineageQuery(sagemaker_session).query(
    start_arns=[endpoint_context.context_arn],
    query_filter=query_filter,
    direction=LineageQueryDirectionEnum.ASCENDANTS,
    include_edges=False,
)

# Parse through the query results to get the ARNs of the training jobs associated with this
 Endpoint
trial_components = []
for vertex in query_result.vertices:
    trial_components.append(vertex.arn)

pp.pprint(trial_components)
```

Example 改变血统的焦点

这些区域有：LineageQuery可以修改为有不同的start_arns这改变了世系的焦点。此外，LineageFilter可以采用多个来源和实体来扩大查询的范围。

在下文中，我们使用该模型作为世系焦点，并找到与其关联的端点和数据集。

```
# Get the ModelArtifact

model_artifact_summary = list(Artifact.list(source_uri=model_package_arn))[0]
model_artifact = ModelArtifact.load(artifact_arn=model_artifact_summary.artifact_arn)
query_filter = LineageFilter(
    entities=[LineageEntityEnum.ARTIFACT],
    sources=[LineageSourceEnum.ENDPOINT, LineageSourceEnum.DATASET],
)

query_result = LineageQuery(sagemaker_session).query(
    start_arns=[model_artifact.artifact_arn],  # Model is the starting artifact
    query_filter=query_filter,
    # Find all the entities that descend from the model, i.e. the endpoint
    direction=LineageQueryDirectionEnum.DESCENDANTS,
    include_edges=False,
)

associations = []
for vertex in query_result.vertices:
    associations.append(vertex.to_lineage_object().source.source_uri)

query_result = LineageQuery(sagemaker_session).query(
    start_arns=[model_artifact.artifact_arn],  # Model is the starting artifact
    query_filter=query_filter,
    # Find all the entities that ascend from the model, i.e. the datasets
    direction=LineageQueryDirectionEnum.ASCENDANTS,
    include_edges=False,
)

for vertex in query_result.vertices:
    associations.append(vertex.to_lineage_object().source.source_uri)

pp.pprint(associations)
```

Example 使用**LineageQueryDirectionEnum.BOTH**找到上升和后代的关系

当方向设置为BOTH，查询会遍历图表以查找上升和后代关系。这种遍历不仅从起始节点进行，而且从访问的每个节点进行。例如；如果训练作业运行了两次，并且训练作业生成的两个模型都部署到终端节点，则查询的结果将方向设置为BOTH显示两个终端节点。这是因为相同的图像用于训练和部署模型。由于图像对模型来说是常见的，因此start_arn以及两个端点都会出现在查询结果中。

```
query_filter = LineageFilter(
    entities=[LineageEntityEnum.ARTIFACT],
    sources=[LineageSourceEnum.ENDPOINT, LineageSourceEnum.DATASET],
)

query_result = LineageQuery(sagemaker_session).query(
    start_arns=[model_artifact.artifact_arn],  # Model is the starting artifact
    query_filter=query_filter,
    # This specifies that the query should look for associations both ascending and
 descending for the start
    direction=LineageQueryDirectionEnum.BOTH,
    include_edges=False,
)
```

```
associations = []
for vertex in query_result.vertices:
    associations.append(vertex.to_lineage_object().source.source_uri)

pp.pprint(associations)
```

Example 中的路线LineageQuery-ASCENDANTS与DESCENDANTS

要了解世系图中的方向，请使用以下实体关系图表-数据集-> 训练 Job-> 模型-> 端点

终端节点是模型的后代，模型是数据集的后代。同样，该模型也是端点的上升。这些区域有：direction参数可用于指定查询是否应返回属于中实体的后代还是上升的实体start_arns. 如果start_arns包含模型，方向是DESCENDANTS，查询返回终端节点。如果方向是ASCENDANTS，查询返回数据集。

```
# In this example, we'll look at the impact of specifying the direction as ASCENDANT or
 DESCENDANT in a `LineageQuery`.

query_filter = LineageFilter(
    entities=[LineageEntityEnum.ARTIFACT],
    sources=[
        LineageSourceEnum.ENDPOINT,
        LineageSourceEnum.MODEL,
        LineageSourceEnum.DATASET,
        LineageSourceEnum.TRAINING_JOB,
    ],
)

query_result = LineageQuery(sagemaker_session).query(
    start_arns=[model_artifact.artifact_arn],
    query_filter=query_filter,
    direction=LineageQueryDirectionEnum.ASCENDANTS,
    include_edges=False,
)

ascendant_artifacts = []

# The lineage entity returned for the Training Job is a TrialComponent which can't be
 converted to a
# lineage object using the method `to_lineage_object()` so we extract the TrialComponent
 ARN.
for vertex in query_result.vertices:
    try:
        ascendant_artifacts.append(vertex.to_lineage_object().source.source_uri)
    except:
        ascendant_artifacts.append(vertex.arn)

print("Ascendant artifacts : ")
pp.pprint(ascendant_artifacts)

query_result = LineageQuery(sagemaker_session).query(
    start_arns=[model_artifact.artifact_arn],
    query_filter=query_filter,
    direction=LineageQueryDirectionEnum.DESCENDANTS,
    include_edges=False,
)

descendant_artifacts = []
for vertex in query_result.vertices:
    try:
        descendant_artifacts.append(vertex.to_lineage_object().source.source_uri)
    except:
        # Handling TrialComponents.
        descendant_artifacts.append(vertex.arn)
```

```
print("Descendant artifacts : ")
pp.pprint(descendant_artifacts)
```

Example SDK 帮助程序功能使世系查询变得更容易

类EndpointContext、ModelArtifact, 和DatasetArtifact有帮助函数是包装在LineageQueryAPI使某些世系查询更容易利用。以下示例演示如何使用这些帮助程序函数。

```
# Find all the datasets associated with this endpoint

datasets = []
dataset_artifacts = endpoint_context.dataset_artifacts()
for dataset in dataset_artifacts:
    datasets.append(dataset.source.source_uri)
print("Datasets : ", datasets)

# Find the training jobs associated with the endpoint
training_job_artifacts = endpoint_context.training_job_arns()
training_jobs = []
for training_job in training_job_artifacts:
    training_jobs.append(training_job)
print("Training Jobs : ", training_jobs)

# Get the ARN for the pipeline execution associated with this endpoint (if any)
pipeline_executions = endpoint_context.pipeline_execution_arn()
if pipeline_executions:
    for pipeline in pipelines_executions:
        print(pipeline)

# Here we use the `ModelArtifact` class to find all the datasets and endpoints associated
 with the model

dataset_artifacts = model_artifact.dataset_artifacts()
endpoint_contexts = model_artifact.endpoint_contexts()

datasets = [dataset.source.source_uri for dataset in dataset_artifacts]
endpoints = [endpoint.source.source_uri for endpoint in endpoint_contexts]

print("Datasets associated with this model : ")
pp.pprint(datasets)

print("Endpoints associated with this model : ")
pp.pprint(endpoints)

# Here we use the `DatasetArtifact` class to find all the endpoints hosting models that
 were trained with a particular dataset
# Find the artifact associated with the dataset

dataset_artifact_arn = list(Artifact.list(source_uri=training_data))[0].artifact_arn
dataset_artifact = DatasetArtifact.load(artifact_arn=dataset_artifact_arn)

# Find the endpoints that used this training dataset
endpoint_contexts = dataset_artifact.endpoint_contexts()
endpoints = [endpoint.source.source_uri for endpoint in endpoint_contexts]

print("Endpoints associated with the training dataset {}".format(training_data))
pp.pprint(endpoints)
```

Example 获取世系图可视化

帮助程序类Visualizer在样本笔记本中提供visualizer.py以帮助绘制世系图。呈现查询响应时，将显示来自StartArns此时将显示。从StartArns该可视化显示了与中返回的其他世系实体的关系query_lineageAPI 操作。

```
# Graph APIs
# Here we use the boto3 `query_lineage` API to generate the query response to plot.

from visualizer import Visualizer

query_response = sm_client.query_lineage(
    StartArns=[endpoint_context.context_arn], Direction="Ascendants", IncludeEdges=True
)

viz = Visualizer()
viz.render(query_response, "Endpoint")

        query_response = sm_client.query_lineage(
    StartArns=[model_artifact.artifact_arn], Direction="Ascendants", IncludeEdges=True
)
viz.render(query_response, "Model")
```

跨账户世系追踪

亚马逊 SageMaker 支持跟踪来自不同的世系实体Amazonaccount. 其他Amazon账户可以与你共享他们的世系实体，你可以通过直接 API 调用来访问这些世系实体或 SageMaker 世系查询。

SageMaker 使用Amazon Resource Access Manager以帮助你安全地共享你的世系资源。你可以通过Amazon RAM控制台.

设置跨账户世系跟踪

你可以分组和分享

跟踪实体维护您的所有元素的表示 end-to-end 机器学习工作流程。您可以使用此表示法来建立模型治理、重现工作流程以及维护工作历史记录。

亚马逊 SageMaker 在创建试验组件时自动为试验组件及其关联的试验和实验创建跟踪实体 SageMaker 处理作业、训练作业和批量转换作业等作业。除了 auto 跟踪之外，你还可以手动创建跟踪实体 (p. 2531)为您的工作流程中的自定义步骤建模 有关更多信息，请参阅 使用亚马逊管理 Machine Learning SageMaker 实验 (p. 1758)。

SageMaker 还会为工作流程中的其他步骤自动创建跟踪实体，以便您可以端到端跟踪工作流程。有关更多信息，请参阅 亚马逊 SageMaker — 创建的追踪实体 (p. 2530)。

您可以创建其他实体来补充 SageMaker 创建的实体。有关更多信息，请参阅 手动创建跟踪实体 (p. 2531)。

SageMaker 会重复使用任何现有实体，而不是创建新实体。例如，只能有一个具有唯一性的工件`SourceUri`.

查询世系的关键概念

- 血统— 用于跟踪机器学习工作流中各个实体之间关系的元数据。
- QueryLineage— 检查你的血统和发现实体之间关系的行动。
- 谱系实体— 你的世系所构成的元数据元素。

- 跨账户谱系— 您的机器学习工作流程可能跨越多个账户。借助跨账户世系，您可以配置多个账户，以便在共享实体资源之间自动创建世系关联。 QueryLineage 那么甚至可以从这些共享账户中返回实体。

定义了以下跟踪实体：

实验实体

- 试验组件— 机器学习试用的阶段。包括处理作业、训练作业和批量转换作业。

- 试用— 通常生成模型的试验组件的组合。
- 试验— 一组试验通常专注于解决特定使用案例。

谱系实体

- 试验组件— 代表世系中的处理、训练和转换工作。也是实验管理的一部分。

- 上下文— 提供其他跟踪或实验实体的逻辑分组。从概念上讲，实验和试验都是上下文。一些示例是终端节点和模型包。
- 操作— 表示操作或活动。通常，操作至少涉及一个输入工件或输出工件。一些示例是工作流程步骤和模型部署。
- Artifact— 表示 URI 可寻址的对象或数据。工件通常是试验组件或操作的输入或输出。一些示例包括数据集（S3 存储桶 URI）或映像（Amazon ECR 注册表路径）。
- 协会— 链接其他跟踪或实验实体，例如训练数据位置与培训作业之间的关联。
  关联有可选的`AssociationType`财产。提供以下值以及每种类型的建议用途。 SageMaker 对它们的使用没有限制：
  - `ContributedTo`— 来源为目的地做出了贡献或参与启用目的地。例如，培训数据为培训工作做出了贡献。
  - `AssociatedWith`— 源已连接到目的地。例如，批准工作流程与模型部署相关联。
  - `DerivedFrom`-目的地是对来源的修改。例如，处理作业的频道输入的摘要输出源于原始输入。
  - `Produced`— 源生成了目的地。例如，培训作业生成了模型工件。
  - `SameAs`— 当在不同账户中使用同一个世系实体时。

通用属性

- 输入属性
  操作、对象和上下文实体具有类型财产，`ActionType`、`ArtifactType`, 和`ContextType`，分别为。此属性是一个自定义字符串，它可以将有意义的信息与实体关联起来，并用作 List API 中的过滤器。
- 源属性
  操作、对象和上下文实体具有`Source`财产。此属性提供实体所代表的底层 URI。一些示例包括：
  - 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`UpdateEndpoint`来源所在的操作`EndpointArn`.
  - 处理作业的图像工件，其中源是`ImageUri`.
  - 网络 ACL 和安全组都允许 (因此可到达您的实例) 的发起 ping 的`Endpoint`来源所在的上下文`EndpointArn`.
- 元数据属性
  动作和工件实体有可选的`Metadata`属性，该属性可提供以下信息：

- `ProjectId`— 例如，的 ID SageMaker 模型所属的 mLOPS 项目。

- `GeneratedBy`— 例如， SageMaker 注册模型包版本的管道执行。
- `Repository`— 例如，包含算法的存储库。
- `CommitId`— 例如，算法版本的提交 ID。

(p. 2528)通过 Amazon SageMaker 中的世系组进行。 SageMaker 每个账户只支持一个默认的世系组。SageMaker 无论何时在您的账户中创建世系实体，都会创建默认的世系组。您账户拥有的每个世系实体都被分配给此默认世系组。要与另一个账户共享世系实体，您可以与该账户共享此默认世系组。

Note

您可以共享世系组中的所有世系追踪实体，也可以不共享。

使用为世系实体创建资源共享Amazon Resource Access Manager控制台。有关更多信息，请参阅 。共享您的Amazon资源中的Amazon Resource Access Manager用户指南.

Note

创建资源共享后，可能需要花几分钟时间，才能完成资源和委托人关联。设置关联后，共享账户将收到加入资源共享的邀请。共享账户必须接受邀请才能获得对共享资源的访问权限。有关接受资源共享邀请的更多信息Amazon RAM，请参阅使用共享Amazon资源中的AmazonResource Access Manager 用户指南.

你的跨账户世系跟踪资源政策

亚马逊 SageMaker 仅支持一种类型的资源策略。这些区域有： SageMaker 资源策略必须允许以下所有操作：

```
"sagemaker:DescribeAction"
"sagemaker:DescribeArtifact"
"sagemaker:DescribeContext"
"sagemaker:DescribeTrialComponent"
"sagemaker:AddAssociation"
"sagemaker:DeleteAssociation"
"sagemaker:QueryLineage"
```

Example 以下是 SageMaker 使用创建资源策略Amazon Resource Access Manager用于为账户世系组创建资源共享。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullLineageAccess",
      "Effect": "Allow",
      "Principal": {
        "AWS": "123456789012" #account-id
      },
      "Action": [
        "sagemaker:DescribeAction",
        "sagemaker:DescribeArtifact",
        "sagemaker:DescribeContext",
        "sagemaker:DescribeTrialComponent",
        "sagemaker:AddAssociation",
        "sagemaker:DeleteAssociation",
        "sagemaker:QueryLineage"
      ],
      "Resource": "arn:aws:sagemaker:us-west-2:111111111111:lineage-group/sagemaker-
default-lineage-group" #Sample lineage group resource
    }
```

```
  ]
}
```

追踪跨账户世系实体

通过跨账户世系跟踪，您可以使用相同的账户关联不同账户中的世系实体`AddAssociation`API 操作。当你关联两个世系实体时， SageMaker 验证您是否有权限执行`AddAssociation`对两个世系实体进行 API 操作。 SageMaker 然后建立协会。如果您没有权限， SageMaker 不创建关联。建立跨账户关联后，您可以通过`QueryLineage`API 操作。有关更多信息，请参阅 查询谱系实体 (p. 2534)。

除此外 SageMaker 如果你有跨账户访问权限，自动创建世系实体， SageMaker 连接引用相同对象或数据的工件。如果来自一个账户的数据被不同账户用于世系跟踪， SageMaker 在每个账户中创建一个工件来跟踪该数据。随时随地使用跨账户世系 SageMaker 创建新的工件， SageMaker 检查是否还有为同一数据创建的其他工件也与您共享。 SageMaker 然后在新创建的工件与与您共享的每个工件之间建立关联`AssociationType`设置为`SameAs`. 然后，您可以使用`QueryLineage`API 操作，用于遍历您自己账户中的世系实体到与您共享但由其他人拥有的世系实体Amazonaccount. 有关更多信息，请参阅 。查询谱系实体 (p. 2534)

主题

- 从其他账户访问世系资源 (p. 2543)
- 查询跨账户世系实体的授权 (p. 2543)

从其他账户访问世系资源

一旦设置了共享世系的跨账户访问权限，您可以致电以下 SageMaker 直接使用 ARN 进行 API 操作以描述来自另一个账户的共享世系实体：

- `DescribeAction`
- `DescribeArtifact`
- `DescribeContext`
- `DescribeTrialComponent`

您还可以管理Associations对于与您共享的不同账户拥有的世系实体，请使用以下命令 SageMaker API 操作：

- `AddAssociation`
- `DeleteAssociation`

对于演示如何使用的笔记本 SageMaker 世系 API 用于查询账户间的世系。 ，请参阅sagemaker-linage-跨账户-ram.ipynb.

查询跨账户世系实体的授权

亚马逊 SageMaker 必须验证您拥有执行`QueryLineage`上的 API 操作`StartArns`. 这是通过附加到`LineageGroup`. 此操作的结果包括您有权访问的所有世系实体，无论它们是您的账户所有还是由另一个账户共享。有关更多信息，请参阅 查询谱系实体 (p. 2534)。

Kubernetes 编排

你可以编排你的 SageMaker 培训和推理工作 SageMaker Kubernetes 和运营商 SageMaker Kubeflow 管道的组件。 SageMaker Kubernetes 的运营商使开发人员和数据科学家可以更轻松地使用 Kubernetes 在 SageMaker 中训练、调整和部署机器学习 (ML) 模型。 SageMaker Kubeflow Pipeline 的组件允许您将数据处理和训练作业从 Kubernetes 集群移动到 SageMaker 的机器学习优化托管服务。

目录

- Kubernetes 的 SageMaker 运营商 (p. 2544)
- 用于 Kubeflow 管道的 SageMaker 组件 (p. 2580)

Kubernetes 的 SageMaker 运营商

Kubernetes 的 SageMaker 运营商使开发人员和数据科学家更轻松地在 SageMaker 中训练、调整和部署机器学习 (ML) 模型。您可以安装它们 SageMaker 在 Amazon Elastic Kubernetes Service (Amazon EKS) 中创建的 Kubernetes 集群上的运营商 SageMaker 本地使用 Kubernetes API 和命令行 Kubernetes 工具的作业，例如`kubectl`. 本指南向您展示如何设置操作员。该指南还介绍了如何使用运算符运行模型训练、超参数调整和推理（实时和批量）。

使用这些运营商无需额外付费。通过这些运营商使用的任何 SageMaker 资源确实会产生费用。这里的程序和指南假设你熟悉 Kubernetes 及其基本命令。

目录

- 什么是运算符？ (p. 2544)
- IAM 基于角色的设置和操作员部署 (p. 2545)
- 删除运算符 (p. 2553)
- 问题排查 (p. 2555)
- 在每个区域中的映像和 SMLOG (p. 2556)
- 使用 Amazon SageMaker 作业 (p. 2556)

什么是运算符？

Kubernetes 基于所谓的控制器模式. 这种模式允许应用程序和工具聆听中央州长（ETCD）的意见，并在发生某种情况时采取行动。此类应用程序示例包括：`cloud-controller-manager`和`controller-manager`. 控制器模式允许您创建分离体验，而不必担心其他组件是如何集成的。要向 Kubernetes 添加新功能，开发人员可以通过创建包含其应用程序特定或域特定逻辑和组件的自定义资源来扩展 Kubernetes API。Kubernetes 中的运营商允许用户本地调用这些自定义资源并自动执行相关工作流程。

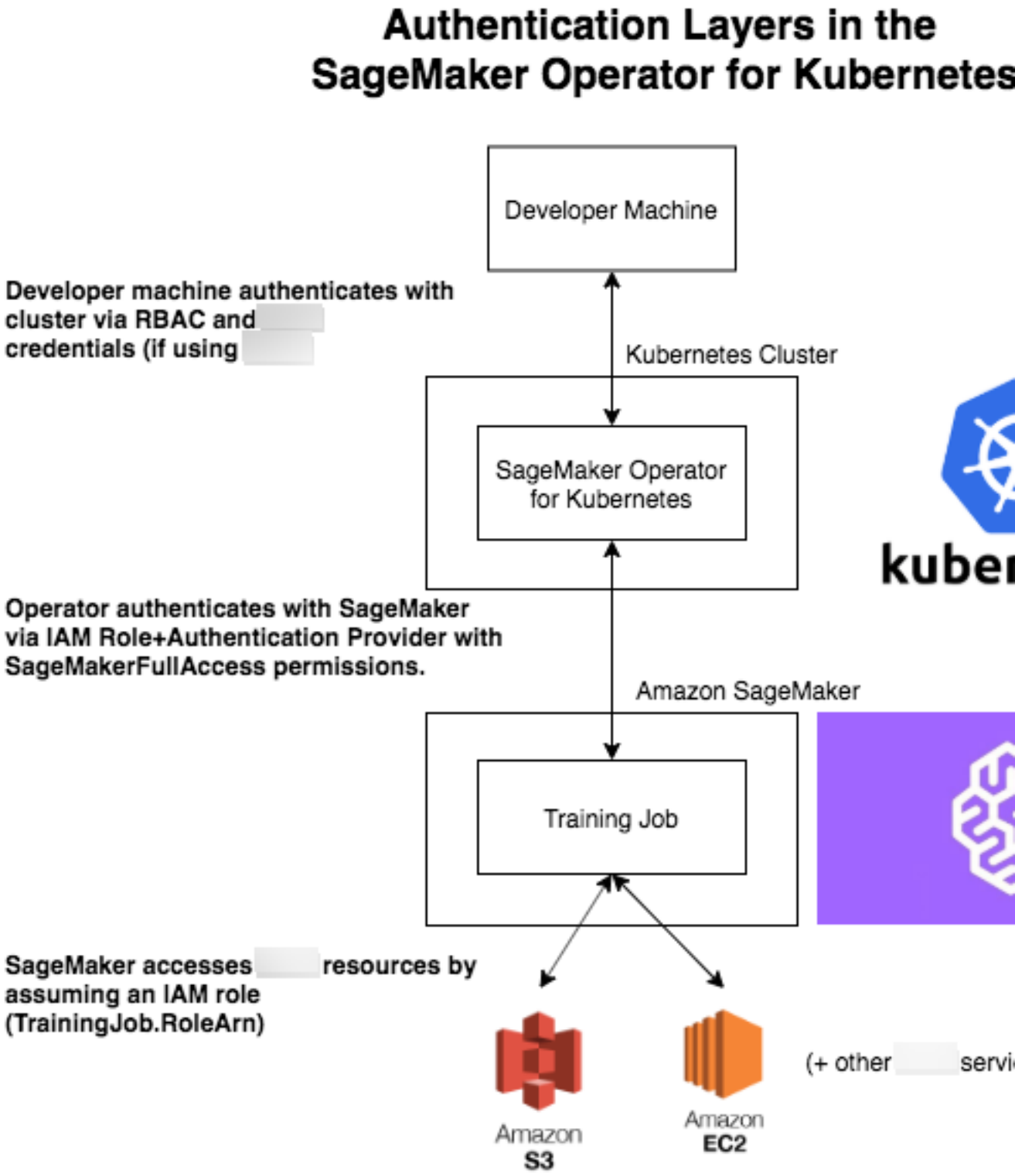
先决条件

本指南假设您已完成以下先决条件：

- 在用于访问 Kubernetes 集群的客户端计算机上安装了以下工具：
  - `kubectl`版本 1.13 或更高版本。使用`kubectl`您的 Amazon EKS 集群控制层面的一个次要版本内的版本。例如，1.13`kubectl`客户端使用 Kubernetes 1.13 和 1.14 集群。1.13 之前的版本不支持 OpenID Connect (OIDC)。
  - `eksctl`版本 0.7.0 或更高版本
  - AmazonCLI版本 1.16.232 或更高版本
  - （可选）Helm版本 3.0 或更高版本
  - aws-iam-认证器
- 具有 IAM 权限，以创建角色并将策略附加到角色。
- 创建了用于运行操作员的 Kubernetes 集群。它应该是 Kubernetes 版本 1.13 或 1.14。对于自动创建群集，使用`eksctl`，请参阅入门 eksctl. 置备集群需要 20 到 30 分钟。

权限概述

这些区域有： SageMaker Kubernetes 的运营商允许你管理职位 SageMaker 从您的 Kubernetes 集群中。运营商访问 SageMaker 代表您的资源。运营商担任与之交互的 IAM 角色Amazon资源不同于您用于访问 Kubernetes 集群的凭证。该角色也不同于 SageMaker 假设在运行机器学习作业时。下图解释了这种设计和流程。



IAM 基于角色的设置和操作员部署

以下各部分介绍了设置和部署操作员的步骤。

集群范围内的部署

在使用 IAM 角色部署运营商之前，请将 OpenID Connect (OIDC) 提供商与您的角色相关联，以便使用 IAM 服务进行身份验证。

为集群创建 OpenID Connect 提供商

以下说明介绍了如何创建 OIDC 提供商并将其与 Amazon EKS 集群关联。

1. 设置本地`CLUSTER_NAME`和`AWS_REGION`环境变量如下：

```
# Set the Region and cluster
export CLUSTER_NAME="<your cluster name>"
export Amazon_REGION="<your region>"
```

2. 使用以下命令将 OIDC 提供商与集群关联。有关更多信息，请参阅 。在集群上为服务账户启用 IAM 角色。

```
eksctl utils associate-iam-oidc-provider --cluster ${CLUSTER_NAME} \
    --region ${AWS_REGION} --approve
```

您的输出应与以下内容类似：

```
[_]  eksctl version 0.10.1
[_]  using region us-east-1
[_]  IAM OpenID Connect provider is associated with cluster "my-cluster" in "us-east-1"
```

既然集群有 OIDC 身份提供商，您可以创建角色并为 Kubernetes 提供 ServiceAccount 担任角色的权限。

获取 OIDC ID

要设置 ServiceAccount，请使用以下命令获取 OpenID Connect 发行者 URL：

```
aws eks describe-cluster --name ${CLUSTER_NAME} --region ${AWS_REGION} \
    --query cluster.identity.oidc.issuer --output text
```

此命令会返回类似以下内容的 URL：

```
https://oidc.eks.${AWS_REGION}.amazonaws.com/id/D48675832CA65BD10A532F597OIDCID
```

在此 URL 中，值`D48675832CA65BD10A532F597OIDCID`是 OIDC ID。集群的 OIDC ID 不同。你需要这个 OIDC ID 值才能创建角色。

如果您的输出是`None`，这意味着你的客户端版本已经旧了。要解决此问题，请运行以下命令：

```
aws eks describe-cluster --region ${AWS_REGION} --query cluster --name ${CLUSTER_NAME} --
output text | grep OIDC
```

OIDC URL 返回如下：

```
OIDC https://oidc.eks.us-east-1.amazonaws.com/id/D48675832CA65BD10A532F597OIDCID
```

创建 IAM 角色

1. 创建一个名为的文件`trust.json`然后将以下信任关系代码块插入到其中。请务必替换所有内容`<OIDC ID>`、`<Amazon account number>`, 和`<EKS Cluster region>`具有与您的集群对应值的占位符。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws-cn:iam::<Amazon account number>:oidc-provider/
oidc.eks.<EKS Cluster region>.amazonaws.com/id/<OIDC ID>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.<EKS Cluster region>.amazonaws.com/id/<OIDC ID>:aud":
 "sts.amazonaws.com",
          "oidc.eks.<EKS Cluster region>.amazonaws.com/id/<OIDC ID>:sub":
 "system:serviceaccount:sagemaker-k8s-operator-system:sagemaker-k8s-operator-default"
        }
      }
    }
  ]
}
```

2. 运行以下命令以创建具有中定义的信任关系的角色。`trust.json`. 此角色使 Amazon EKS 群集能够从 IAM 获取和刷新凭证。

```
aws iam create-role --region ${AWS_REGION} --role-name <role name> --assume-role-
policy-document file://trust.json --output=text
```

您的输出应与以下内容类似：

```
ROLE    arn:aws:iam::123456789012:role/my-role 2019-11-22T21:46:10Z    /
 ABCDEFSFODNN7EXAMPLE   my-role
ASSUMEROLEPOLICYDOCUMENT        2012-10-17
STATEMENT       sts:AssumeRoleWithWebIdentity   Allow
STRINGEQUALS    sts.amazonaws.com       system:serviceaccount:sagemaker-k8s-operator-
system:sagemaker-k8s-operator-default
PRINCIPAL       arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/
```

记下`ROLE ARN`; 你将这个值传递给你的运营商。

将 AmazonSageMakerFullAccess 角色的策略

要授予角色对 SageMaker 的访问权限，请附加AmazonSageMakerFullAccess政策。如果要限制对运营商的权限，可以创建自己的自定义策略并附加它。

附加以下策略：

```
aws iam attach-role-policy --region ${AWS_REGION} --role-name <role name> --policy-arn
 arn:aws-cn:iam::aws:policy/AmazonSageMakerFullAccess
```

Kubernetes ServiceAccount `sagemaker-k8s-operator-default`应该具有`AmazonSageMakerFullAccess`权限。在您安装操作员时，请确认这一点。

部署运算符

在部署操作员时，您可以使用 YAML 文件或 Helm 图表。

使用 YAML 部署操作员

这是部署操作符的最简单方法。过程如下所述：

1. 下载以下安装程序脚本。WHEN`installer.yaml`被引用，使用`installer_china.yaml`相反。

```
wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
release/rolebased/china/installer_china.yaml
```

2. 编辑`installer.yaml`要替换的文件`eks.amazonaws.com/role-arn`. 在此将 ARN 替换为您创建的基于 OIDC 的角色的 Amazon 资源名称 (ARN)。
3. 使用以下命令来部署集群：

```
kubectl apply -f installer.yaml
```

使用 Helm 图表部署操作员

使用提供的头盔图安装操作员。

1. 使用以下命令克隆 Helm 安装程序目录：

```
git clone https://github.com/aws/amazon-sagemaker-operator-for-k8s.git
```

2. 导航到 `amazon-sagemaker-operator-for-k8s/hack/charts/installer` 文件夹。编辑`rolebased/values.yaml`文件，其中包括图表的高级参数。在此将角色 ARN 替换为您创建的基于 OIDC 的角色的 Amazon 资源名称 (ARN)。
3. 使用以下命令安装 Helm Chart:

```
kubectl create namespace sagemaker-k8s-operator-system
helm install --namespace sagemaker-k8s-operator-system sagemaker-operator rolebased/
```

   如果您决定将运算符安装到指定的命名空间以外的命名空间中，则需要调整 IAM 角色中定义的命名空间`trust.json`要匹配的文件。
4. 过了一会儿，图表安装了一个随机生成的名称。运行以下命令验证安装是否成功：

```
helm ls
```

   您的输出应与以下内容类似：

```
NAME                    NAMESPACE                       REVISION        UPDATED
                         STATUS          CHART                           APP VERSION
sagemaker-operator      sagemaker-k8s-operator-system   1               2019-11-20
 23:14:59.6777082 +0000 UTC    deployed        sagemaker-k8s-operator-0.1.0
```

验证操作员的部署

1. 您应该可以看到 SageMaker 运行以下命令以对部署到集群的每个操作员进行自定义资源定义 (CRD)：

```
kubectl get crd | grep sagemaker
```

   您的输出应与以下内容类似：

```
batchtransformjobs.sagemaker.aws.amazon.com         2019-11-20T17:12:34Z
```

```
endpointconfigs.sagemaker.aws.amazon.com            2019-11-20T17:12:34Z
hostingdeployments.sagemaker.aws.amazon.com         2019-11-20T17:12:34Z
hyperparametertuningjobs.sagemaker.aws.amazon.com   2019-11-20T17:12:34Z
models.sagemaker.aws.amazon.com                     2019-11-20T17:12:34Z
trainingjobs.sagemaker.aws.amazon.com               2019-11-20T17:12:34Z
```

2. 请确保操作员 Pod 成功运行。使用以下命令列出所有 Pod：

```
kubectl -n sagemaker-k8s-operator-system get pods
```

您应看到一个名为的 podsagemaker-k8s-operator-controller-manager-*****在命名空间中sagemaker-k8s-operator-system如下所示：

```
NAME                                                     READY   STATUS    RESTARTS
   AGE
sagemaker-k8s-operator-controller-manager-12345678-r8abc   2/2     Running   0
   23s
```

命名空间范围内的部署

您可以选择在单个 Kubernetes 命名空间的范围内安装操作员。在此模式下，控制器只监控和协调资源 SageMaker 如果资源是在该命名空间中创建的。这样可以更精细地控制哪个控制器管理哪些资源。这对于部署到多个非常有用Amazon或控制哪些用户可以访问特定作业。

本指南概述了如何将运算符安装到特定的、预定义的命名空间中。要将控制器部署到第二个命名空间，请从头到尾遵循指南并在每个步骤中更改命名空间。

为您的亚马逊 EKS 集群创建 OpenID Connect 提供商

以下说明介绍了如何创建 OIDC 提供商并将其与 Amazon EKS 集群关联。

1. 设置本地CLUSTER_NAME和AWS_REGION环境变量如下：

```
# Set the region and cluster
export CLUSTER_NAME="<your cluster name>"
export Amazon_REGION="<your region>"
```

2. 使用以下命令将 OIDC 提供商与集群关联。有关更多信息，请参阅 。在集群上为服务账户启用 IAM 角色。

```
eksctl utils associate-iam-oidc-provider --cluster ${CLUSTER_NAME} \
    --region ${AWS_REGION} --approve
```

您的输出应与以下内容类似：

```
[_]  eksctl version 0.10.1
[_]  using region us-east-1
[_]  IAM OpenID Connect provider is associated with cluster "my-cluster" in "us-east-1"
```

既然集群有 OIDC 身份提供商，请创建一个角色并赋予 Kubernetes ServiceAccount 担任角色的权限。

获取你的 OIDC ID

要设置 ServiceAccount，首先使用以下命令获取 OpenID Connect 发行者 URL：

```
aws eks describe-cluster --name ${CLUSTER_NAME} --region ${AWS_REGION} \
```

```
    --query cluster.identity.oidc.issuer --output text
```

此命令会返回类似以下内容的 URL：

```
https://oidc.eks.${AWS_REGION}.amazonaws.com/id/D48675832CA65BD10A532F597OIDCID
```

在此 URL 中，值 D48675832CA65BD10A532F597OIDCID 是 OIDC ID。集群的 OIDC ID 将会有所不同。你需要这个 OIDC ID 值才能创建角色。

如果您的输出是None，这意味着你的客户端版本已经旧了。要解决此问题，请运行以下命令：

```
aws eks describe-cluster --region ${AWS_REGION} --query cluster --name ${CLUSTER_NAME} --
output text | grep OIDC
```

OIDC URL 返回如下：

```
OIDC https://oidc.eks.us-east-1.amazonaws.com/id/D48675832CA65BD10A532F597OIDCID
```

创建 IAM 角色

1. 创建一个名为的文件trust.json然后将以下信任关系代码块插入到其中。请务必替换所有内容<OIDC ID>、<Amazon account number>, 和<EKS Cluster region>具有与您的集群对应值的占位符。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws-cn:iam::<Amazon account number>:oidc-provider/
oidc.eks.<EKS Cluster region>.amazonaws.com/id/<OIDC ID>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.<EKS Cluster region>.amazonaws.com/id/<OIDC ID>:aud":
 "sts.amazonaws.com",
          "oidc.eks.<EKS Cluster region>.amazonaws.com/id/<OIDC ID>:sub":
 "system:serviceaccount:<Namespace>:sagemaker-k8s-operator-default"
        }
      }
    }
  ]
}
```

2. 运行以下命令以创建具有中定义的信任关系的角色。trust.json. 此角色使 Amazon EKS 群集能够从 IAM 获取和刷新凭证。

```
aws iam create-role --region ${AWS_REGION} --role-name <role name> --assume-role-
policy-document file://trust.json --output=text
```

您的输出应与以下内容类似：

```
ROLE    arn:aws:iam::123456789012:role/my-role 2019-11-22T21:46:10Z    /
 ABCDEFSFODNN7EXAMPLE   my-role
ASSUMEROLEPOLICYDOCUMENT        2012-10-17
STATEMENT       sts:AssumeRoleWithWebIdentity   Allow
STRINGEQUALS    sts.amazonaws.com       system:serviceaccount:my-namespace:sagemaker-
k8s-operator-default
```

```
PRINCIPAL       arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/
```

记下ROLE ARN. 你把这个值传递给你的运营商。

将 AmazonSageMakerFullAccess 您角色的策略

要授予角色对 SageMaker 的访问权限，请附加AmazonSageMakerFullAccess政策。如果要限制对运营商的权限，可以创建自己的自定义策略并附加它。

附加以下策略：

```
aws iam attach-role-policy --region ${AWS_REGION} --role-name <role name> --policy-arn
 arn:aws-cn:iam::aws:policy/AmazonSageMakerFullAccess
```

Kubernetes ServiceAccount sagemaker-k8s-operator-default应该具有AmazonSageMakerFullAccess权限。在您安装操作员时，请确认这一点。

将操作员部署到你的命名空间

在部署操作员时，您可以使用 YAML 文件或 Helm 图表。

使用 YAML 将操作员部署到你的命名空间

在命名空间范围内部署运算符有两部分。第一个是在群集级别安装的 CRD 集。每个 Kubernetes 集群只需安装一次这些资源定义。第二部分是操作员权限和部署本身。

如果尚未将 CRD 安装到群集中，请使用以下命令应用 CRD 安装程序 YAML：

```
kubectl apply -f https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/
master/release/rolebased/namespaced/crd.yaml
```

要将操作员安装到群集上：

1. 下载以下操作员脚本。WHENoperator.yaml被引用，使用operator_china.yaml相反。

   ```
   wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
   release/rolebased/namespaced/china/operator_china.yaml
   ```

2. 使用以下命令更新安装程序 YAML 以将资源放入指定的命名空间：

   ```
   sed -i -e 's/PLACEHOLDER-NAMESPACE/<YOUR NAMESPACE>/g' operator.yaml
   ```

3. 编辑operator.yaml文件将资源放入eks.amazonaws.com/role-arn. 在此将 ARN 替换为您创建的基于 OIDC 的角色的 Amazon 资源名称 (ARN)。
4. 使用以下命令来部署集群：

   ```
   kubectl apply -f operator.yaml
   ```

使用 Helm Tchart 将操作员部署到你的命名空间

在命名空间范围内部署运算符需要两个部分。第一个是在群集级别安装的 CRD 集。每个 Kubernetes 集群只需安装一次这些资源定义。第二部分是操作员权限和部署本身。当使用头盔图时，你必须首先使用kubectl.

1. 使用以下命令克隆 Helm 安装程序目录：

```
git clone https://github.com/aws/amazon-sagemaker-operator-for-k8s.git
```

2. 导航到 amazon-sagemaker-operator-for-k8s/hack/charts/installer/namespaced 文件夹。编辑rolebased/values.yaml文件，其中包括图表的高级参数。在此将角色 ARN 替换为您创建的基于 OIDC 的角色的 Amazon 资源名称 (ARN)。
3. 使用以下命令安装 Helm Chart:

```
helm install crds crd_chart/
```

4. 使用以下命令创建所需的命名空间并安装运算符：

```
kubectl create namespace <namespace>
helm install --n <namespace> op operator_chart/
```

5. 过了一会儿，图表用名称安装sagemaker-operator. 运行以下命令验证安装是否成功：

```
helm ls
```

您的输出应与以下内容类似：

```
NAME                    NAMESPACE                       REVISION        UPDATED
                         STATUS          CHART                           APP VERSION
sagemaker-operator      my-namespace                    1               2019-11-20
 23:14:59.6777082 +0000 UTC    deployed        sagemaker-k8s-operator-0.1.0
```

验证操作员部署到你的命名空间

1. 您应该可以看到 SageMaker 运行以下命令以对部署到集群的每个操作员进行自定义资源定义 (CRD)：

```
kubectl get crd | grep sagemaker
```

您的输出应与以下内容类似：

```
batchtransformjobs.sagemaker.aws.amazon.com          2019-11-20T17:12:34Z
endpointconfigs.sagemaker.aws.amazon.com             2019-11-20T17:12:34Z
hostingdeployments.sagemaker.aws.amazon.com          2019-11-20T17:12:34Z
hyperparametertuningjobs.sagemaker.aws.amazon.com    2019-11-20T17:12:34Z
models.sagemaker.aws.amazon.com                      2019-11-20T17:12:34Z
trainingjobs.sagemaker.aws.amazon.com                2019-11-20T17:12:34Z
```

2. 请确保操作员 Pod 成功运行。使用以下命令列出所有 Pod：

```
kubectl -n my-namespace get pods
```

您应看到一个名为的 podsagemaker-k8s-operator-controller-manager-*****在命名空间中my-namespace如下所示：

```
NAME                                                     READY   STATUS    RESTARTS
   AGE
sagemaker-k8s-operator-controller-manager-12345678-r8abc     2/2     Running   0
   23s
```

安装 SageMaker 圆木kubectl插件

作为的一部分 SageMaker Kubernetes 的运算符，您可以使用smlogs 插件为了kubectl. 这启用 SageMaker 要流式传输的 CloudWatch 日志kubectl.kubectl必须安装到您的路径. 以下命令将二进制文件放在sagemaker-k8s-bin目录在你的主目录中，然后将该目录添加到PATH.

```
export os="linux"

wget https://amazon-sagemaker-operator-for-k8s-us-east-1.s3.amazonaws.com/kubectl-smlogs-
plugin/v1/${os}.amd64.tar.gz
tar xvzf ${os}.amd64.tar.gz

# Move binaries to a directory in your homedir.
mkdir ~/sagemaker-k8s-bin
cp ./kubectl-smlogs.${os}.amd64/kubectl-smlogs ~/sagemaker-k8s-bin/.

# This line adds the binaries to your PATH in your .bashrc.

echo 'export PATH=$PATH:~/sagemaker-k8s-bin' >> ~/.bashrc

# Source your .bashrc to update environment variables:
source ~/.bashrc
```

使用以下命令验证kubectl正确安装插件：

```
kubectl smlogs
```

如果kubectl插件安装正确，您的输出应与以下内容类似：

```
View SageMaker logs via Kubernetes

Usage:
  smlogs [command]

Aliases:
  smlogs, SMLogs, Smlogs

Available Commands:
  BatchTransformJob       View BatchTransformJob logs via Kubernetes
  TrainingJob             View TrainingJob logs via Kubernetes
  help                    Help about any command

Flags:
  -h, --help   help for smlogs

Use "smlogs [command] --help" for more information about a command.
```

删除运算符

删除基于集群的运算符

使用 YAML 安装的操作员

要从群集中卸载操作员，请确保所有 SageMaker 资源已从群集中删除。否则会导致操作员删除操作挂起。删除所有操作后 SageMaker 工作，使用kubectl从集群中删除运算符。运行以下命令停止所有作业并从集群中删除操作员：

```
# Delete all SageMaker jobs from Kubernetes
```

```
kubectl delete --all --all-namespaces hyperparametertuningjob.sagemaker.aws.amazon.com
kubectl delete --all --all-namespaces trainingjobs.sagemaker.aws.amazon.com
kubectl delete --all --all-namespaces batchtransformjob.sagemaker.aws.amazon.com
kubectl delete --all --all-namespaces hostingdeployment.sagemaker.aws.amazon.com

# Delete the operator and its resources
kubectl delete -f /installer.yaml
```

您应看到类似下面的输出：

```
$ kubectl delete --all --all-namespaces trainingjobs.sagemaker.aws.amazon.com
trainingjobs.sagemaker.aws.amazon.com "xgboost-mnist-from-for-s3" deleted

$ kubectl delete --all --all-namespaces hyperparametertuningjob.sagemaker.aws.amazon.com
hyperparametertuningjob.sagemaker.aws.amazon.com "xgboost-mnist-hpo" deleted

$ kubectl delete --all --all-namespaces batchtransformjob.sagemaker.aws.amazon.com
batchtransformjob.sagemaker.aws.amazon.com "xgboost-mnist" deleted

$ kubectl delete --all --all-namespaces hostingdeployment.sagemaker.aws.amazon.com
hostingdeployment.sagemaker.aws.amazon.com "host-xgboost" deleted

$ kubectl delete -f raw-yaml/installer.yaml
namespace "sagemaker-k8s-operator-system" deleted
customresourcedefinition.apiextensions.k8s.io "batchtransformjobs.sagemaker.aws.amazon.com"
 deleted
customresourcedefinition.apiextensions.k8s.io "endpointconfigs.sagemaker.aws.amazon.com"
 deleted
customresourcedefinition.apiextensions.k8s.io "hostingdeployments.sagemaker.aws.amazon.com"
 deleted
customresourcedefinition.apiextensions.k8s.io
 "hyperparametertuningjobs.sagemaker.aws.amazon.com" deleted
customresourcedefinition.apiextensions.k8s.io "models.sagemaker.aws.amazon.com" deleted
customresourcedefinition.apiextensions.k8s.io "trainingjobs.sagemaker.aws.amazon.com"
 deleted
role.rbac.authorization.k8s.io "sagemaker-k8s-operator-leader-election-role" deleted
clusterrole.rbac.authorization.k8s.io "sagemaker-k8s-operator-manager-role" deleted
clusterrole.rbac.authorization.k8s.io "sagemaker-k8s-operator-proxy-role" deleted
rolebinding.rbac.authorization.k8s.io "sagemaker-k8s-operator-leader-election-rolebinding"
 deleted
clusterrolebinding.rbac.authorization.k8s.io "sagemaker-k8s-operator-manager-rolebinding"
 deleted
clusterrolebinding.rbac.authorization.k8s.io "sagemaker-k8s-operator-proxy-rolebinding"
 deleted
service "sagemaker-k8s-operator-controller-manager-metrics-service" deleted
deployment.apps "sagemaker-k8s-operator-controller-manager" deleted
secrets "sagemaker-k8s-operator-abcde" deleted
```

使用 Helm Chart 安装的操作员

要删除操作员 CRD，首先删除所有正在运行的作业。然后删除用于使用以下命令部署操作员的 Helm Chart：

```
# get the helm charts
helm ls

# delete the charts
helm delete <chart name>
```

删除基于命名空间的运算符

使用 YAML 安装的操作员

要从群集中卸载操作员，请确保所有 SageMaker 资源已从群集中删除。否则会导致操作员删除操作挂起。删除所有操作后 SageMaker 工作，使用`kubectl`首先从命名空间中删除运算符，然后从集群中删除 CRD。运行以下命令停止所有作业并从集群中删除操作员：

```
# Delete all SageMaker jobs from Kubernetes
kubectl delete --all --all-namespaces hyperparametertuningjob.sagemaker.aws.amazon.com
kubectl delete --all --all-namespaces trainingjobs.sagemaker.aws.amazon.com
kubectl delete --all --all-namespaces batchtransformjob.sagemaker.aws.amazon.com
kubectl delete --all --all-namespaces hostingdeployment.sagemaker.aws.amazon.com
```

```
# Delete the operator using the same yaml file that was used to install the operator
kubectl delete -f operator.yaml

# Now delete the CRDs using the CRD installer yaml
kubectl delete -f https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/
master/release/rolebased/namespaced/crd.yaml

# Now you can delete the namespace if you want
kubectl delete namespace <namespace>
```

安装了 Helm Tharts 的操作员

要删除操作员 CRD，首先删除所有正在运行的作业。然后删除用于使用以下命令部署操作员的 Helm Chart：

```
# Delete the operator
helm delete -n <namespace> op

# delete the crds
helm delete crds

# optionally delete the namespace
kubectl delete namespace <namespace>
```

问题排查

调试失败的 Job

- 通过运行以下命令检查作业状态：

```
kubectl get <CRD Type> <job name>
```

- 如果作业是在 SageMaker 中创建的，则可以使用以下命令查看`STATUS`和`SageMaker Job Name`：

```
kubectl get <crd type> <job name>
```

- 您可以使用`smlogs`要查找问题的原因，请使用以下命令：

```
kubectl smlogs <crd type> <job name>
```

- 您也可以使用`describe`命令可以使用以下命令获取有关作业的更多详细信息。输出有`additional`该字段包含有关作业状态的更多信息。

```
kubectl describe <crd type> <job name>
```

- 如果作业不是在 SageMaker 中创建的，则使用操作员 Pod 的日志来查找问题的原因，如下所示：

```
$ kubectl get pods -A | grep sagemaker
# Output:
sagemaker-k8s-operator-system   sagemaker-k8s-operator-controller-manager-5cd7df4d74-
wh22z   2/2     Running   0          3h33m

$ kubectl logs -p <pod name> -c manager -n sagemaker-k8s-operator-system
```

删除操作员 CRD

如果删除作业不起作用，请检查操作员是否在运行。如果运算符没有运行，则必须使用以下步骤删除终结器：

1. 在新终端中，使用以下方法在编辑器中打开作业`kubectl edit`如下所示：

   ```
   kubectl edit <crd type> <job name>
   ```

2. 通过从文件中删除以下两行来编辑作业以删除终结器。保存该文件，然后将作业删除。

   ```
   finalizers:
   - sagemaker-operator-finalizer
   ```

在每个区域中的映像和 SMLOG

下表列出了每个区域中可用的操作员映像和 SMLOG。

区域	控制器映像	Linux SMLOG
us-east-1	957583890962.dkr.ecr.us-east-1.amazonaws.com/amazon-sagemaker-operator-for-k8s:v1	https://s3.us-east-1.amazonaws.com/amazon-sagemaker-operator-for-k8s-us-east-1/kubectl-smlogs-plugin/v1/linux.amd64.tar.gz
us-east-2	922499468684.dkr.ecr.us-east-2.amazonaws.com/amazon-sagemaker-operator-for-k8s:v1	https://s3.us-east-2.amazonaws.com/amazon-sagemaker-operator-for-k8s-us-east-2/kubectl-smlogs-plugin/v1/linux.amd64.tar.gz
us-west-2	640106867763.dkr.ecr.us-west-2.amazonaws.com/amazon-sagemaker-operator-for-k8s:v1	https://s3.us-west-2.amazonaws.com/amazon-sagemaker-operator-for-k8s-us-west-2/kubectl-smlogs-plugin/v1/linux.amd64.tar.gz
eu-west-1	613661167059.dkr.ecr.eu-west-1.amazonaws.com/amazon-sagemaker-operator-for-k8s:v1	https://s3.eu-west-1.amazonaws.com/amazon-sagemaker-operator-for-k8s-eu-west-1/kubectl-smlogs-plugin/v1/linux.amd64.tar.gz

使用 Amazon SageMaker 作业

运行亚马逊 SageMaker 使用 Kubernetes 的运算符作业，您可以应用 YAML 文件或使用提供的头盔图表。

以下教程中的所有示例操作员作业都使用从公共 MNIST 数据集中获取的示例数据。要运行这些示例，请将数据集下载到 Amazon S3 存储桶中。您可以在中找到数据集下载 MNIST 数据集。

目录

- 训练作业运算符 (p. 2557)
- 超参数优化 Job 运算符 (p. 2561)
- BatchTransjob 运算符 (p. 2566)
- HostingDeployment 运算符 (p. 2569)
- CrocessingJob 运算符 (p. 2573)
- 托管自动扩展策略 (HAP) 运营商 (p. 2577)

训练作业运算符

培训工作操作员将您指定的培训职位规格与 SageMaker 通过在 SageMaker 中为你启动它。你能了解有关的更多信息 SageMaker 培训中的作业 SageMaker CreateTrainingJob API 文档.

主题

- 创建 TrainingJob 使用 YAML 文件 (p. 2557)
- 创建 TrainingJob 使用 Helm Chart (p. 2557)
- 列出 TrainingJobs (p. 2558)
- 描述培训作业 (p. 2559)
- 查看 TTrainingJobs 中的日志 (p. 2560)
- 删除 TTrainingJobs (p. 2561)

创建 TrainingJob 使用 YAML 文件

1. 使用以下命令，下载示例 YAML 文件进行训练：

```
wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
samples/xgboost-mnist-trainingjob.yaml
```

2. 编辑`xgboost-mnist-trainingjob.yaml`文件来替换`roleArn`带有你的参数`<sagemaker-execution-role>`, 和`outputPath`使用 Amazon S3 存储桶 SageMaker 执行角色有写权限。这些区域有：`roleArn`必须具有权限以便 SageMaker 可以代表您访问 Amazon S3、Amazon CloudWatch 和其他服务。有关创建 SageMaker ExecutionRole，请参阅SageMaker 角色. 使用以下命令应用 YAML 文件：

```
kubectl apply -f xgboost-mnist-trainingjob.yaml
```

创建 TrainingJob 使用 Helm Chart

你可以使用头盔图来运行 TTrainingJobs。

1. 克隆 GitHub 使用以下命令获取源代码：

```
git clone https://github.com/aws/amazon-sagemaker-operator-for-k8s.git
```

2. 导航到`amazon-sagemaker-operator-for-k8s/hack/charts/training-jobs/`文件夹然后编辑`values.yaml`文件来替换类似的值`rolearn`和`outputpath`其值与您的账户对应。RoleARN 必须具有权限才能 SageMaker 可以代表您访问 Amazon S3、Amazon CloudWatch 和其他服务。有关创建 SageMaker ExecutionRole，请参阅SageMaker 角色.

创建 TrainJob

将角色和 Amazon S3 存储桶替换为中的适当值`values.yaml`，您可以使用以下命令创建训练任务：

```
helm install . --generate-name
```

您的输出应与以下内容类似：

```
NAME: chart-12345678
LAST DEPLOYED: Wed Nov 20 23:35:49 2019
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thanks for installing the sagemaker-k8s-trainingjob.
```

验证你的训练头盔图

要验证 Helm Chart 是否已成功创建，请运行：

```
helm ls
```

您的输出应与以下内容类似：

```
NAME                    NAMESPACE       REVISION        UPDATED
     STATUS          CHART                           APP VERSION
chart-12345678        default         1               2019-11-20 23:35:49.9136092 +0000 UTC
   deployed        sagemaker-k8s-trainingjob-0.1.0
rolebased-12345678    default         1               2019-11-20 23:14:59.6777082 +0000 UTC
   deployed        sagemaker-k8s-operator-0.1.0
```

`helm install`创建`TrainingJob`Kubernetes 资源。运营商启动了实际的培训工作 SageMaker 并更新`TrainingJob`用于反映 SageMaker 中作业状态的 Kubernetes 资源。在作业期间使用的 SageMaker 资源需要支付费用。作业完成或停止后，不会产生任何费用。

注意： SageMaker 不允许你更新正在运行的训练作业。您无法编辑任何参数并重新应用文件/配置。要么更改元数据名称，或者删除现有作业并创建一个新的作业。类似于 Kubeflow 中的 TFJob 等现有的培训职位操作员，`update`不支持 。

列出 TrainingJobs

使用以下命令列出使用 Kubernetes 运算符创建的所有作业：

```
kubectl get TrainingJob
```

列出所有作业的输出应与以下内容类似：

```
kubectl get trainingjobs
NAME                        STATUS       SECONDARY-STATUS    CREATION-TIME
 SAGEMAKER-JOB-NAME
xgboost-mnist-from-for-s3   InProgress   Starting            2019-11-20T23:42:35Z   xgboost-
mnist-from-for-s3-examplef11eab94e0ed4671d5a8f
```

在作业完成或失败后，培训作业将继续列出。你可以删除`TrainingJob`按照删除培训 Job 步骤从列表中找到作业。已完成或停止的作业不会产生任何费用 SageMaker 资源的费用。

培训工作状态值

这些区域有：STATUS字段可以是以下值之一：

- Completed
- InProgress
- Failed
- Stopped
- Stopping

这些状态直接来自 SageMaker 官方的API 文档.

除了官员 SageMaker 状态，有可能STATUS要成为SynchronizingK8sJobWithSageMaker. 这意味着操作员尚未处理任务。

辅助状态值

次要状态直接来自 SageMaker 官方的API 文档. 它们包含有关作业状态的更多细致信息。

描述培训作业

您可以使用describe kubectl命令。这通常用于调试问题或检查培训作业的参数。要获取有关训练任务的信息，请使用以下命令：

```
kubectl describe trainingjob xgboost-mnist-from-for-s3
```

您的训练作业的输出应与以下内容类似：

```
Name:         xgboost-mnist-from-for-s3
Namespace:    default
Labels:       <none>
Annotations:  <none>
API Version:  sagemaker.aws.amazon.com/v1
Kind:         TrainingJob
Metadata:
  Creation Timestamp:  2019-11-20T23:42:35Z
  Finalizers:
    sagemaker-operator-finalizer
  Generation:        2
  Resource Version:  23119
  Self Link:         /apis/sagemaker.aws.amazon.com/v1/namespaces/default/trainingjobs/
xgboost-mnist-from-for-s3
  UID:               6d7uiui-0bef-11ea-b94e-0ed467example
Spec:
  Algorithm Specification:
    Training Image:       8256416981234.dkr.ecr.us-east-2.amazonaws.com/xgboost:1
    Training Input Mode:  File
  Hyper Parameters:
    Name:   eta
    Value:  0.2
    Name:   gamma
    Value:  4
    Name:   max_depth
    Value:  5
    Name:   min_child_weight
    Value:  6
    Name:   num_class
    Value:  10
    Name:   num_round
    Value:  10
    Name:   objective
```

```
    Value:  multi:softmax
    Name:   silent
    Value:  0
  Input Data Config:
    Channel Name:      train
    Compression Type:  None
    Content Type:      text/csv
    Data Source:
      S 3 Data Source:
        S 3 Data Distribution Type:  FullyReplicated
        S 3 Data Type:               S3Prefix
        S 3 Uri:                     https://s3-us-east-2.amazonaws.com/my-bucket/
sagemaker/xgboost-mnist/train/
    Channel Name:                    validation
    Compression Type:                None
    Content Type:                    text/csv
    Data Source:
      S 3 Data Source:
        S 3 Data Distribution Type:  FullyReplicated
        S 3 Data Type:               S3Prefix
        S 3 Uri:                     https://s3-us-east-2.amazonaws.com/my-bucket/
sagemaker/xgboost-mnist/validation/
  Output Data Config:
    S 3 Output Path:  s3://my-bucket/sagemaker/xgboost-mnist/xgboost/
  Region:             us-east-2
  Resource Config:
    Instance Count:     1
    Instance Type:      ml.m4.xlarge
    Volume Size In GB:  5
  Role Arn:             arn:aws:iam::12345678910:role/service-role/AmazonSageMaker-
ExecutionRole
  Stopping Condition:
    Max Runtime In Seconds:  86400
  Training Job Name:         xgboost-mnist-from-for-s3-6d7fa0af0bef11eab94e0example
Status:
  Cloud Watch Log URL:           https://us-east-2.console.aws.amazon.com/
cloudwatch/home?region=us-east-2#logStream:group=/aws/sagemaker/
TrainingJobs;prefix=<example>;streamFilter=typeLogStreamPrefix
  Last Check Time:               2019-11-20T23:44:29Z
  Sage Maker Training Job Name:  xgboost-mnist-from-for-s3-6d7fa0af0bef11eab94eexample
  Secondary Status:              Downloading
  Training Job Status:           InProgress
Events:                          <none>
```

查看 TTrainingJobs 中的日志

使用以下命令从查看日志kmeans-mnist训练任务：

```
kubectl smlogs trainingjob xgboost-mnist-from-for-s3
```

您的输出应类似于以下内容。来自实例的日志按时间顺序排序。

```
"xgboost-mnist-from-for-s3" has SageMaker TrainingJobName "xgboost-mnist-from-for-
s3-123456789" in region "us-east-2", status "InProgress" and secondary status "Starting"
xgboost-mnist-from-for-s3-6d7fa0af0bef11eab94e0ed46example/algo-1-1574293123 2019-11-20
 23:45:24.7 +0000 UTC Arguments: train
xgboost-mnist-from-for-s3-6d7fa0af0bef11eab94e0ed46example/algo-1-1574293123 2019-11-20
 23:45:24.7 +0000 UTC [2019-11-20:23:45:22:INFO] Running standalone xgboost training.
xgboost-mnist-from-for-s3-6d7fa0af0bef11eab94e0ed46example/algo-1-1574293123 2019-11-20
 23:45:24.7 +0000 UTC [2019-11-20:23:45:22:INFO] File size need to be processed in the
 node: 1122.95mb. Available memory size in the node: 8586.0mb
xgboost-mnist-from-for-s3-6d7fa0af0bef11eab94e0ed46example/algo-1-1574293123 2019-11-20
 23:45:24.7 +0000 UTC [2019-11-20:23:45:22:INFO] Determined delimiter of CSV input is ','
```

```
xgboost-mnist-from-for-s3-6d7fa0af0bef11eab94e0ed46example/algo-1-1574293123 2019-11-20
 23:45:24.7 +0000 UTC [23:45:22] S3DistributionType set as FullyReplicated
```

删除 TTrainingJobs

使用以下命令停止 Amazon SageMaker 上的训练任务：

```
kubectl delete trainingjob xgboost-mnist-from-for-s3
```

此命令删除 SageMaker 来自 Kubernetes 的培训工作。此命令将返回以下输出：

```
trainingjob.sagemaker.aws.amazon.com "xgboost-mnist-from-for-s3" deleted
```

如果 SageMaker 上的作业仍在进行中，则作业将停止。你不会产生任何费用 SageMaker 作业停止或完成后的资源。

注意： SageMaker 不会删除训练作业。停止的作业继续显示在 SageMaker 控制台。这些区域有：`delete`命令大约需要 2 分钟才能清理 SageMaker 的资源。

超参数优化 Job 运算符

超参数调整作业运算符将指定的超参数调整作业规范协调到 SageMaker 通过在 SageMaker 中启动它。你能了解有关的更多信息 SageMaker 中的超参数优化作业 SageMaker CreateHyperParameterTuningJob API 文档.

主题

- 创建 HyperparameterTuningJob 使用 YAML 文件 (p. 2561)
- 创建 HyperparameterTuningJob 使用 Helm Chart (p. 2561)
- 列出超参数调整 Job (p. 2562)
- 描述超级参数优化作业 (p. 2563)
- 从超参数优化作业查看日志 (p. 2566)
- 删除超参数调整 Job (p. 2566)

创建 HyperparameterTuningJob 使用 YAML 文件

1. 使用以下命令下载超参数调整作业的示例 YAML 文件：

   ```
   wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
   samples/xgboost-mnist-hpo.yaml
   ```

2. 编辑`xgboost-mnist-hpo.yaml`文件来替换`roleArn`带有你的参数`sagemaker-execution-role`. 要使超参数调整作业成功，还必须更改`s3InputPath`和`s3OutputPath`转换为与您的账户对应的值。使用以下命令应用更新 YAML 文件：

   ```
   kubectl apply -f xgboost-mnist-hpo.yaml
   ```

创建 HyperparameterTuningJob 使用 Helm Chart

您可以使用 Helm Charts 来运行超参数调整作业。

1. 克隆 GitHub 使用以下命令获取源代码：

   ```
   git clone https://github.com/aws/amazon-sagemaker-operator-for-k8s.git
   ```

2. 导航到 amazon-sagemaker-operator-for-k8s/hack/charts/hyperparameter-tuning-jobs/ 文件夹。
3. 编辑values.yaml文件来替换roleArn带有你的参数sagemaker-execution-role. 要使超参数调整作业成功，还必须更改s3InputPath和s3OutputPath转换为与您的账户对应的值。

创建超参数调整 Job

将角色和 Amazon S3 路径替换为中的适当值values.yaml，您可以使用以下命令创建超参数调整作业：

```
helm install . --generate-name
```

您的输出应类似于以下内容：

```
NAME: chart-1574292948
LAST DEPLOYED: Wed Nov 20 23:35:49 2019
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thanks for installing the sagemaker-k8s-hyperparametertuningjob.
```

验证图表安装

要验证 Helm Chart 是否已成功创建，请运行以下命令：

```
helm ls
```

您的输出应与以下内容类似：

```
NAME                    NAMESPACE       REVISION        UPDATED
chart-1474292948        default         1               2019-11-20 23:35:49.9136092 +0000
 UTC    deployed        sagemaker-k8s-hyperparametertuningjob-0.1.0
      STATUS          CHART                           APP VERSION
chart-1574292948        default         1               2019-11-20 23:35:49.9136092 +0000
 UTC    deployed        sagemaker-k8s-trainingjob-0.1.0
rolebased-1574291698    default         1               2019-11-20 23:14:59.6777082 +0000
 UTC    deployed        sagemaker-k8s-operator-0.1.0
```

helm install创建HyperParameterTuningJobKubernetes 资源。运营商在中启动实际的超参数优化作业 SageMaker 并更新HyperParameterTuningJob用于反映 SageMaker 中作业状态的 Kubernetes 资源。您将向您收取费用 SageMaker 在工作期间使用的资源。作业完成或停止后，不会产生任何费用。

注意： SageMaker 不允许您更新正在运行的超参数调整作业。您无法编辑任何参数并重新应用文件/配置。您必须更改元数据名称或删除现有作业并创建新的作业。类似于 Kubeflow 中的 TFJob 等现有的培训职位操作员，update不支持 。

列出超参数调整 Job

使用以下命令列出使用 Kubernetes 运算符创建的所有作业：

```
kubectl get hyperparametertuningjob
```

您的输出应与以下内容类似：

```
NAME          STATUS      CREATION-TIME          COMPLETED   INPROGRESS   ERRORS   STOPPED
 BEST-TRAINING-JOB                                SAGEMAKER-JOB-NAME
```

```
xgboost-mnist-hpo    Completed    2019-10-17T01:15:52Z    10          0             0         0
        xgboostha92f5e3cf07b11e9bf6c06d6-009-4c7a123    xgboostha92f5e3cf07b11e9bf6c123
```

作业完成或失败后，超参数调整作业将继续列出。你可以删除hyperparametertuningjob按照删除超参数调整作业中的步骤从列表中执行操作。已完成或停止的作业不会产生任何费用 SageMaker 资源的费用。

超参数优化 Job 状态值

这些区域有：STATUS字段可以是以下值之一：

- Completed
- InProgress
- Failed
- Stopped
- Stopping

这些状态直接来自 SageMaker 官方的API 文档.

除了官员 SageMaker 状态，有可能STATUS要成为SynchronizingK8sJobWithSageMaker. 这意味着操作员尚未处理任务。

状态计数器

输出有几个计数器，例如COMPLETED和INPROGRESS. 这些工作分别表示已经完成和正在进行的培训工作数量。有关如何确定这些值的更多信息，请参阅。训练作业状态计数器中的 SageMaker API 文档。

最佳培训作业

此列包含TrainingJob最好优化了选定的指标。

要查看调整后的超参数的摘要，请运行：

```
kubectl describe hyperparametertuningjob xgboost-mnist-hpo
```

要查看有关的详细信息TrainingJob运行：

```
kubectl describe trainingjobs <job name>
```

生成训 TrainingJobs

你还可以跟踪 Kubernetes 发起的所有 10 个培训职位HyperparameterTuningJob通过运行以下命令：

```
kubectl get trainingjobs
```

描述超级参数优化作业

您可以使用describe kubectl命令。

```
kubectl describe hyperparametertuningjob xgboost-mnist-hpo
```

除了有关调整作业的信息之外， SageMaker Kubernetes 的运营商也公开了最佳训练作业在中找到超参数优化作业describe输出如下所示：

```
Name:         xgboost-mnist-hpo
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
```

```
               {"apiVersion":"sagemaker.aws.amazon.com/
v1","kind":"HyperparameterTuningJob","metadata":{"annotations":{},"name":"xgboost-mnist-
hpo","namespace":...
API Version:  sagemaker.aws.amazon.com/v1
Kind:         HyperparameterTuningJob
Metadata:
  Creation Timestamp:  2019-10-17T01:15:52Z
  Finalizers:
    sagemaker-operator-finalizer
  Generation:        2
  Resource Version:  8167
  Self Link:         /apis/sagemaker.aws.amazon.com/v1/namespaces/default/
hyperparametertuningjobs/xgboost-mnist-hpo
  UID:               a92f5e3c-f07b-11e9-bf6c-06d6f303uidu
Spec:
  Hyper Parameter Tuning Job Config:
    Hyper Parameter Tuning Job Objective:
      Metric Name:  validation:error
      Type:         Minimize
    Parameter Ranges:
      Integer Parameter Ranges:
        Max Value:     20
        Min Value:     10
        Name:          num_round
        Scaling Type:  Linear
    Resource Limits:
      Max Number Of Training Jobs:     10
      Max Parallel Training Jobs:      10
    Strategy:                          Bayesian
    Training Job Early Stopping Type:  Off
  Hyper Parameter Tuning Job Name:     xgboostha92f5e3cf07b11e9bf6c06d6
  Region:                              us-east-2
  Training Job Definition:
    Algorithm Specification:
      Training Image:       12345678910.dkr.ecr.us-east-2.amazonaws.com/xgboost:1
      Training Input Mode:  File
    Input Data Config:
      Channel Name:  train
      Content Type:  text/csv
      Data Source:
        s3DataSource:
          s3DataDistributionType:  FullyReplicated
          s3DataType:              S3Prefix
          s3Uri:                   https://s3-us-east-2.amazonaws.com/my-bucket/sagemaker/
xgboost-mnist/train/
      Channel Name:                validation
      Content Type:                text/csv
      Data Source:
        s3DataSource:
          s3DataDistributionType:  FullyReplicated
          s3DataType:              S3Prefix
          s3Uri:                   https://s3-us-east-2.amazonaws.com/my-bucket/sagemaker/
xgboost-mnist/validation/
    Output Data Config:
      s3OutputPath:  https://s3-us-east-2.amazonaws.com/my-bucket/sagemaker/xgboost-mnist/
xgboost
    Resource Config:
      Instance Count:     1
      Instance Type:      ml.m4.xlarge
      Volume Size In GB:  5
    Role Arn:             arn:aws:iam::123456789012:role/service-role/AmazonSageMaker-
ExecutionRole
    Static Hyper Parameters:
      Name:   base_score
      Value:  0.5
      Name:   booster
```

```
      Value:  gbtree
      Name:   csv_weights
      Value:  0
      Name:   dsplit
      Value:  row
      Name:   grow_policy
      Value:  depthwise
      Name:   lambda_bias
      Value:  0.0
      Name:   max_bin
      Value:  256
      Name:   max_leaves
      Value:  0
      Name:   normalize_type
      Value:  tree
      Name:   objective
      Value:  reg:linear
      Name:   one_drop
      Value:  0
      Name:   prob_buffer_row
      Value:  1.0
      Name:   process_type
      Value:  default
      Name:   rate_drop
      Value:  0.0
      Name:   refresh_leaf
      Value:  1
      Name:   sample_type
      Value:  uniform
      Name:   scale_pos_weight
      Value:  1.0
      Name:   silent
      Value:  0
      Name:   sketch_eps
      Value:  0.03
      Name:   skip_drop
      Value:  0.0
      Name:   tree_method
      Value:  auto
      Name:   tweedie_variance_power
      Value:  1.5
    Stopping Condition:
      Max Runtime In Seconds:  86400
Status:
  Best Training Job:
    Creation Time:  2019-10-17T01:16:14Z
    Final Hyper Parameter Tuning Job Objective Metric:
      Metric Name:        validation:error
      Value:
    Objective Status:     Succeeded
    Training End Time:    2019-10-17T01:20:24Z
    Training Job Arn:     arn:aws:sagemaker:us-east-2:123456789012:training-job/
xgboostha92f5e3cf07b11e9bf6c06d6-009-4sample
    Training Job Name:    xgboostha92f5e3cf07b11e9bf6c06d6-009-4c7a3059
    Training Job Status:  Completed
    Training Start Time:  2019-10-17T01:18:35Z
    Tuned Hyper Parameters:
      Name:                                     num_round
      Value:                                    18
  Hyper Parameter Tuning Job Status:            Completed
  Last Check Time:                              2019-10-17T01:21:01Z
  Sage Maker Hyper Parameter Tuning Job Name:  xgboostha92f5e3cf07b11e9bf6c06d6
  Training Job Status Counters:
    Completed:            10
    In Progress:          0
    Non Retryable Error:  0
```

```
    Retryable Error:      0
    Stopped:              0
    Total Error:          0
Events:                   <none>
```

从超参数优化作业查看日志

超参数调整作业没有日志，但它们启动的所有训练作业都有日志。可以像正常的培训工作一样访问这些日志。有关更多信息，请参阅查看训练作业中的日志。

删除超参数调整 Job

使用以下命令停止 SageMaker 中的超参数作业。

```
kubectl delete hyperparametertuningjob xgboost-mnist-hpo
```

此命令将超参数调整作业和关联的训练作业从 Kubernetes 集群中删除，然后在 SageMaker 中停止它们。已停止或完成的作业不会产生任何费用 SageMaker 资源的费用。 SageMaker 不删除超参数优化作业。停止的作业继续显示在 SageMaker 控制台。

您的输出应与以下内容类似：

```
hyperparametertuningjob.sagemaker.aws.amazon.com "xgboost-mnist-hpo" deleted
```

注意：要清除命令来自 SageMaker 的资源，大约需要 2 分钟时间。

BatchTransjob 运算符

Batch 转换作业操作员将指定的批量转换作业规范协调到 SageMaker 通过在 SageMaker 中启动它。你能了解有关的更多信息 SageMaker 中的批量转换作业 SageMaker CreateTransformJob API 文档.

主题

- 创建 BatchTransformJob 使用 YAML 文件 (p. 2566)
- 创建 BatchTransformJob 使用 Helm Chart (p. 2566)
- 列表 BatchTransJobs (p. 2567)
- 描述 BatchTransMob (p. 2568)
- 从 BatchTransJob 查看日志 (p. 2569)
- 删除一个 BatchTransJob (p. 2569)

创建 BatchTransformJob 使用 YAML 文件

1. 使用以下命令下载批量转换作业的示例 YAML 文件：

   ```
   wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
   samples/xgboost-mnist-batchtransform.yaml
   ```

2. 编辑文件`xgboost-mnist-batchtransform.yaml`更改必要的参数以替换`inputdataconfig`使用您的输入数据和`s3OutputPath`使用 Amazon S3 存储桶， SageMaker 执行角色有写权限。
3. 使用以下命令应用 YAML 文件：

   ```
   kubectl apply -f xgboost-mnist-batchtransform.yaml
   ```

创建 BatchTransformJob 使用 Helm Chart

您可以使用 Helm Chart 来运行批量转换作业。

获取 Helm 安装程序目录

克隆 GitHub 使用以下命令获取源代码：

```
git clone https://github.com/aws/amazon-sagemaker-operator-for-k8s.git
```

配置头盔图

导航到 `amazon-sagemaker-operator-for-k8s/hack/charts/batch-transform-jobs/` 文件夹。

编辑`values.yaml`文件来替换`inputdataconfig`使用输入数据和 OutputPath 与 S3 存储桶一起使用 SageMaker 执行角色具有写入权限。

创建一个 BatchTransJob

1. 使用以下命令创建批量转换任务：

   ```
   helm install . --generate-name
   ```

   您的输出应与以下内容类似：

   ```
   NAME: chart-1574292948
   LAST DEPLOYED: Wed Nov 20 23:35:49 2019
   NAMESPACE: default
   STATUS: deployed
   REVISION: 1
   TEST SUITE: None
   NOTES:
   Thanks for installing the sagemaker-k8s-batch-transform-job.
   ```

2. 要验证 Helm Chart 是否已成功创建，请运行以下命令：

   ```
   helm ls
   NAME                    NAMESPACE       REVISION        UPDATED
            STATUS          CHART                           APP VERSION
   chart-1474292948        default         1               2019-11-20 23:35:49.9136092
    +0000 UTC   deployed        sagemaker-k8s-batchtransformjob-0.1.0
   chart-1474292948        default         1               2019-11-20 23:35:49.9136092
    +0000 UTC   deployed        sagemaker-k8s-hyperparametertuningjob-0.1.0
   chart-1574292948        default         1               2019-11-20 23:35:49.9136092
    +0000 UTC   deployed        sagemaker-k8s-trainingjob-0.1.0
   rolebased-1574291698    default         1               2019-11-20 23:14:59.6777082
    +0000 UTC   deployed        sagemaker-k8s-operator-0.1.0
   ```

   此命令将创建`BatchTransformJob`Kubernetes 资源。运营商在中启动实际的转换作业 SageMaker 并更新`BatchTransformJob`用于反映 SageMaker 中作业状态的 Kubernetes 资源。您将向您收取费用 SageMaker 在工作期间使用的资源。作业完成或停止后，不会产生任何费用。

注意： SageMaker 不允许您更新正在运行的批处理转换作业。您无法编辑任何参数并重新应用文件/配置。您必须更改元数据名称或删除现有作业并创建新的作业。类似于 Kubeflow 中的 TFJob 等现有的培训职位操作员，`update`不支持 。

列表 BatchTransJobs

使用以下命令列出使用 Kubernetes 运算符创建的所有作业：

```
kubectl get batchtransformjob
```

您的输出应与以下内容类似：

```
NAME                                  STATUS      CREATION-TIME          SAGEMAKER-JOB-NAME
xgboost-mnist-batch-transform         Completed   2019-11-18T03:44:00Z   xgboost-mnist-
a88fb19809b511eaac440aa8axgboost
```

作业完成或失败后，批处理转换作业将继续列出。你可以删除hyperparametertuningjob按照删除 Batch 转换 Job 步骤从列表中执行。已完成或停止的作业不会产生任何费用 SageMaker 资源的费用。

Batch 转换状态值

这些区域有：STATUS字段可以是以下值之一：

- Completed
- InProgress
- Failed
- Stopped
- Stopping

这些状态直接来自 SageMaker 官方的API 文档.

除了官员 SageMaker 状态，有可能STATUS要成为SynchronizingK8sJobWithSageMaker. 这意味着操作员尚未处理任务。

描述 BatchTransMob

您可以使用describe kubectl命令。

```
kubectl describe batchtransformjob xgboost-mnist-batch-transform
```

您的输出应与以下内容类似：

```
Name:         xgboost-mnist-batch-transform
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"sagemaker.aws.amazon.com/
v1","kind":"BatchTransformJob","metadata":{"annotations":{},"name":"xgboost-
mnist","namespace"...
API Version:  sagemaker.aws.amazon.com/v1
Kind:         BatchTransformJob
Metadata:
  Creation Timestamp:  2019-11-18T03:44:00Z
  Finalizers:
    sagemaker-operator-finalizer
  Generation:        2
  Resource Version:  21990924
  Self Link:         /apis/sagemaker.aws.amazon.com/v1/namespaces/default/
batchtransformjobs/xgboost-mnist
  UID:               a88fb198-09b5-11ea-ac44-0aa8a9UIDNUM
Spec:
  Model Name:  TrainingJob-20190814SMJOb-IKEB
  Region:      us-east-1
  Transform Input:
    Content Type:  text/csv
    Data Source:
      S 3 Data Source:
        S 3 Data Type:  S3Prefix
        S 3 Uri:        s3://my-bucket/mnist_kmeans_example/input
  Transform Job Name:   xgboost-mnist-a88fb19809b511eaac440aa8a9SMJOB
```

```
  Transform Output:
    S 3 Output Path:  s3://my-bucket/mnist_kmeans_example/output
  Transform Resources:
    Instance Count:  1
    Instance Type:   ml.m4.xlarge
Status:
  Last Check Time:                2019-11-19T22:50:40Z
  Sage Maker Transform Job Name:  xgboost-mnist-a88fb19809b511eaac440aaSMJOB
  Transform Job Status:           Completed
Events:                           <none>
```

从 BatchTransJob 查看日志

使用以下命令从查看日志`xgboost-mnist`批量转换任务：

```
kubectl smlogs batchtransformjob xgboost-mnist-batch-transform
```

删除一个 BatchTransJob

使用以下命令停止 SageMaker 中的批量转换任务。

```
kubectl delete batchTransformJob xgboost-mnist-batch-transform
```

您的输出应与以下内容类似：

```
batchtransformjob.sagemaker.aws.amazon.com "xgboost-mnist" deleted
```

此命令将从 Kubernetes 集群中删除批处理转换作业，并在 SageMaker 中停止它们。已停止或完成的作业不会对 SageMaker 资源产生任何费用。删除大约需要 2 分钟才能清除 SageMaker 的资源。

注意： SageMaker 不会删除批量转换作业。停止的作业继续显示在 SageMaker 控制台。

HostingDeployment 运算符

HostingDeploy 操作员支持创建和删除终端节点，以及更新现有终端节点以实时推断。托管部署运营商会将您指定的托管部署作业规范与 SageMaker 通过在 SageMaker 中创建模型、端点配置和端点。你能了解有关的更多信息 SageMaker 中的推理 SageMaker CreateEndpoint 节点 API 文档.

主题

- 配置 HostingDeployment 资源 (p. 2569)
- 创建托管部署 (p. 2570)
- 列出托管/部署 (p. 2570)
- 描述托管部署 (p. 2570)
- 调用终端节点 (p. 2572)
- 更新托管部署 (p. 2572)
- 删除托管部署 (p. 2573)

配置 HostingDeployment 资源

使用以下命令下载托管部署作业的示例 YAML 文件：

```
wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
samples/xgboost-mnist-hostingdeployment.yaml
```

这些区域有：`xgboost-mnist-hostingdeployment.yaml`文件具有以下组件，可以根据需要进行编辑：

- ProductionVariants. 生产变体是为单个模型提供服务的一组实例。 SageMaker 根据设定的权重在所有生产型号之间进行负载平衡。
- 模型. 模型是为模型提供服务所必需的容器和执行角色 ARN。它至少需要一个容器。
- 容器. 容器指定数据集和服务影像。如果您使用自己的自定义算法而非 SageMaker 提供的算法，则推理代码必须满足 SageMaker 要求。有关更多信息，请参阅 。在 SageMaker 中使用您自己的算法.

创建托管部署

要创建托管部署，请使用`kubectl`应用文件`hosting.yaml`使用以下命令：

```
kubectl apply -f hosting.yaml
```

SageMaker 使用指定配置创建端点。在终端节点生命周期内使用的 SageMaker 资源需要支付费用。删除终端节点后，您不会产生任何费用。

创建过程大约需要 10 分钟。

列出托管/部署

验证 HostingDeployment 已创建，请使用以下命令：

```
kubectl get hostingdeployments
```

您的输出应与以下内容类似：

```
NAME           STATUS     SAGEMAKER-ENDPOINT-NAME
host-xgboost   Creating   host-xgboost-def0e83e0d5f11eaaa450aSMLOGS
```

托管部署状态值

状态字段可以是以下值之一：

- `SynchronizingK8sJobWithSageMaker`：操作符正准备创建终端节点。
- `ReconcilingEndpoint`：操作员正在创建、更新或删除终端节点资源。如果 HostingDeployment 仍处于此状态，请使用`kubectl describe`查看原因`Additional`字段中返回的子位置类型。
- `OutOfService`：终端节点不可用于接收传入的请求。
- `Creating`：CreateEndpoint正在执行。
- `Updating`：UpdateEndpoint要么UpdateEndpointWeightsAndCapacities正在执行。
- `SystemUpdating`：终端正在进行维护，在完成之前无法更新、删除或重新扩展。此维护操作不会更改任何客户指定的值，例如 VPC 配置、KMS 加密、模型、实例类型或实例计数。
- `RollingBack`：Endpoint 无法向上或缩小或更改其变体权重，并且正在回滚到以前的配置。回滚完成后，终端节点将返回到`InService`状态。此过渡状态仅适用于启用了自动缩放并且作为UpdateEndpointWeightsAndCapacities打电话或当UpdateEndpointWeightsAndCapacities显式调用操作。
- `InService`：终端节点可用于处理传入的请求。
- `Deleting`：DeleteEndpoint正在执行。
- `Failed`：无法创建、更新或重新缩放端点。使用描述端点：失败原因有关失败的信息。DeleteEndpoint是唯一可以在失败的终端节点上执行的操作。

描述托管部署

您可以使用`describe kubectl`命令。

```
kubectl describe hostingdeployment
```

您的输出应与以下内容类似：

```
Name:         host-xgboost
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"sagemaker.aws.amazon.com/
v1","kind":"HostingDeployment","metadata":{"annotations":{},"name":"host-
xgboost","namespace":"def..."
API Version:  sagemaker.aws.amazon.com/v1
Kind:         HostingDeployment
Metadata:
  Creation Timestamp:  2019-11-22T19:40:00Z
  Finalizers:
    sagemaker-operator-finalizer
  Generation:        1
  Resource Version:  4258134
  Self Link:         /apis/sagemaker.aws.amazon.com/v1/namespaces/default/
hostingdeployments/host-xgboost
  UID:               def0e83e-0d5f-11ea-aa45-0a3507uiduid
Spec:
  Containers:
    Container Hostname:  xgboost
    Image:               123456789012.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest
    Model Data URL:      s3://my-bucket/inference/xgboost-mnist/model.tar.gz
  Models:
    Containers:
      xgboost
    Execution Role Arn:  arn:aws:iam::123456789012:role/service-role/AmazonSageMaker-
ExecutionRole
    Name:                xgboost-model
    Primary Container:   xgboost
  Production Variants:
    Initial Instance Count:  1
    Instance Type:           ml.c5.large
    Model Name:              xgboost-model
    Variant Name:            all-traffic
  Region:                    us-east-2
Status:
  Creation Time:         2019-11-22T19:40:04Z
  Endpoint Arn:          arn:aws:sagemaker:us-east-2:123456789012:endpoint/host-xgboost-
def0e83e0d5f11eaaaexample
  Endpoint Config Name:  host-xgboost-1-def0e83e0d5f11e-e08f6c510d5f11eaaa450aexample
  Endpoint Name:         host-xgboost-def0e83e0d5f11eaaa450a350733ba06
  Endpoint Status:       Creating
  Endpoint URL:          https://runtime.sagemaker.us-east-2.amazonaws.com/endpoints/host-
xgboost-def0e83e0d5f11eaaaexample/invocations
  Last Check Time:       2019-11-22T19:43:57Z
  Last Modified Time:    2019-11-22T19:40:04Z
  Model Names:
    Name:   xgboost-model
    Value:  xgboost-model-1-def0e83e0d5f11-df5cc9fd0d5f11eaaa450aexample
Events:     <none>
```

状态字段使用以下字段提供更多信息：

- `Additional`：有关托管部署状态的更多信息。此字段是可选的，只有在出错时才填充。
- `Creation Time`：在 SageMaker 中创建终端节点的时间。
- `Endpoint ARN`: 该 SageMaker 终端节点 ARN。
- `Endpoint Config Name`: 该 SageMaker 终端节点配置的名称。

- `Endpoint Name`: 该 SageMaker 终端节点的名称。
- `Endpoint Status`：终端节点的状态。
- `Endpoint URL`：可用于访问终端节点的 HTTPS URL。有关更多信息，请参阅 。在上部署模型 SageMaker 托管服务.
- `FailureReason`：如果创建、更新或删除命令失败，则此处显示原因。
- `Last Check Time`：操作员上次检查终端节点状态的时间。
- `Last Modified Time`：上次修改终端节点的时间。
- `Model Names`：键值对 HostingDeployment 模型名称转换为 SageMaker 模型名称。

调用终端节点

终端节点状态为之后`InService`，您可以通过两种方式调用终端节点：使用AmazonCLI，它执行身份验证和 URL 请求签名，或者使用像 cURL 这样的 HTTP 客户端。如果你使用自己的客户端，你需要这样做 Amazonv4 URL 签名和身份验证自己。

使用AmazonCLI，运行以下命令。确保将区域和终端节点名称替换为终端节点的区域，并 SageMaker 终端节点名称。这些信息可以从输出中获得`kubectl describe`.

```
# Invoke the endpoint with mock input data.
aws sagemaker-runtime invoke-endpoint \
  --region us-east-2 \
  --endpoint-name <endpoint name> \
  --body $(seq 784 | xargs echo | sed 's/ /,/g') \
  >(cat) \
  --content-type text/csv > /dev/null
```

例如，如果您的区域为`us-east-2`你的终端节点配置名称是`host-xgboost-f56b6b280d7511ea824b129926example`，那么以下命令将调用终端节点：

```
aws sagemaker-runtime invoke-endpoint \
  --region us-east-2 \
  --endpoint-name host-xgboost-f56b6b280d7511ea824b1299example \
  --body $(seq 784 | xargs echo | sed 's/ /,/g') \
  >(cat) \
  --content-type text/csv > /dev/null
4.95847082138
```

此处,`4.95847082138`是模拟数据模型的预测。

更新托管部署

1. 曾经 HostingDeployment 状态为`InService`，可以更新。可能需要花费大约 10 分钟 HostingDeployment 要在服务中。验证状态是否为`InService`，使用以下命令：

   ```
   kubectl get hostingdeployments
   ```

2. 这些区域有： HostingDeployment 可以在状态为之前更新`InService`. 操作员等到 SageMaker 终端节点是`InService`在应用更新之前。

   要应用更新，请修改`hosting.yaml`文件。例如，更改`initialInstanceCount`字段从 1 到 2，如下所示：

   ```
   apiVersion: sagemaker.aws.amazon.com/v1
   kind: HostingDeployment
   metadata:
     name: host-xgboost
   spec:
       region: us-east-2
       productionVariants:
   ```

```
        - variantName: all-traffic
          modelName: xgboost-model
          initialInstanceCount: 2
          instanceType: ml.c5.large
    models:
        - name: xgboost-model
          executionRoleArn: arn:aws:iam::123456789012:role/service-role/
AmazonSageMaker-ExecutionRole
          primaryContainer: xgboost
          containers:
            - xgboost
    containers:
        - containerHostname: xgboost
          modelDataUrl: s3://my-bucket/inference/xgboost-mnist/model.tar.gz
          image: 123456789012.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest
```

3. 保存文件，然后使用kubectl按如下方式应用更新。您应该会看到状态变化InService到ReconcilingEndpoint，那么Updating.

```
$ kubectl apply -f hosting.yaml
hostingdeployment.sagemaker.aws.amazon.com/host-xgboost configured

$ kubectl get hostingdeployments
NAME           STATUS                SAGEMAKER-ENDPOINT-NAME
host-xgboost   ReconcilingEndpoint   host-xgboost-def0e83e0d5f11eaaa450a350abcdef

$ kubectl get hostingdeployments
NAME           STATUS     SAGEMAKER-ENDPOINT-NAME
host-xgboost   Updating   host-xgboost-def0e83e0d5f11eaaa450a3507abcdef
```

SageMaker 将与您的模型一起部署一组新的实例，切换流量以使用新实例，并消耗旧实例。一旦这个过程开始，状态就变成Updating. 更新完成后，终端节点将变为InService. 此过程大约需要 10 分钟。

删除托管部署

1. 使用kubectl删除 HostingDeployment 使用以下命令：

```
kubectl delete hostingdeployments host-xgboost
```

您的输出应与以下内容类似：

```
hostingdeployment.sagemaker.aws.amazon.com "host-xgboost" deleted
```

2. 要验证托管部署是否已删除，请使用以下命令：

```
kubectl get hostingdeployments
No resources found.
```

已删除的终端节点不会产生任何费用 SageMaker 资源的费用。

CrocessingJob 运算符

ProcessingJob 运营商用于启动亚马逊 SageMaker 处理作业。有关 SageMaker 请参阅处理作业CreateProcessingJob.

主题

- 创建 ProcessingJob 使用 YAML 文件 (p. 2574)

- 列出处理作业 (p. 2574)
- 描述处理作业 (p. 2575)
- 删除处理作业 (p. 2577)

创建 ProcessingJob 使用 YAML 文件

按照以下步骤创建亚马逊 SageMaker 使用 YAML 文件处理作业：

1. 下载`kmeans_preprocessing.py`预处理脚本。

```
wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
samples/kmeans_preprocessing.py
```

2. 在您的一个 Amazon Simple Storage Service (Amazon S3) 存储桶中，创建`mnist_kmeans_example/processing_code`文件夹然后将脚本上传到文件夹中。
3. 下载 `kmeans-mnist-processingjob.yaml` 文件。

```
wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
samples/kmeans-mnist-processingjob.yaml
```

4. 编辑 YAML 文件以指定`sagemaker-execution-role`并替换的所有`my-bucket`使用 S3 存储桶。

```
...
metadata:
  name: kmeans-mnist-processing
...
  roleArn: arn:aws:iam::<acct-id>:role/service-role/<sagemaker-execution-role>
  ...
  processingOutputConfig:
    outputs:
      ...
          s3Output:
            s3Uri: s3://<my-bucket>/mnist_kmeans_example/output/
  ...
  processingInputs:
    ...
        s3Input:
          s3Uri: s3://<my-bucket>/mnist_kmeans_example/processing_code/
kmeans_preprocessing.py
```

   这些区域有：`sagemaker-execution-role`这样就必须具有权限 SageMaker 可以代表您访问 S3 存储桶、Amazon CloudWatch 和其他服务。有关创建执行角色的更多信息，请参阅。SageMaker 角色.
5. 使用以下命令之一应用 YAML 文件。

   对于集群范围内的安装：

```
kubectl apply -f kmeans-mnist-processingjob.yaml
```

   对于命名空间范围内的安装：

```
kubectl apply -f kmeans-mnist-processingjob.yaml -n <NAMESPACE>
```

列出处理作业

使用以下命令之一列出所有使用 ProcessingJob 运算符。`SAGEMAKER-JOB-NAME` 来自`metadata`YAML 文件的部分。

对于集群范围内的安装：

```
kubectl get ProcessingJob kmeans-mnist-processing
```

对于命名空间范围内的安装：

```
kubectl get ProcessingJob -n <NAMESPACE> kmeans-mnist-processing
```

您的输出应类似于以下内容：

```
NAME                    STATUS     CREATION-TIME        SAGEMAKER-JOB-NAME
kmeans-mnist-processing InProgress 2020-09-22T21:13:25Z kmeans-mnist-
processing-7410ed52fd1811eab19a165ae9f9e385
```

输出列出了所有作业，无论其状态如何。要从列表中删除作业，请参阅删除处理 Job.

处理作业状态

- SynchronizingK8sJobWithSageMaker— 作业首先提交到群集。运营商已收到请求，并准备创建处理作业。
- Reconciling— 操作员正在初始化或从暂时错误中恢复，以及其他错误。如果处理作业仍处于此状态，请使用kubectl describe命令来查看原因Additional字段中返回的子位置类型。
- InProgress | Completed | Failed | Stopping | Stopped— 的状态 SageMaker 处理作业。有关更多信息，请参阅 。描述处理作业.
- Error— 操作员无法通过协调来恢复。

已完成、停止或失败的作业不会对 SageMaker 资源产生进一步的费用。

描述处理作业

使用以下命令之一获取有关处理作业的更多详细信息。这些命令通常用于调试问题或检查处理作业的参数。

对于集群范围内的安装：

```
kubectl describe processingjob kmeans-mnist-processing
```

对于命名空间范围内的安装：

```
kubectl describe processingjob kmeans-mnist-processing -n <NAMESPACE>
```

处理作业的输出应类似于以下内容。

```
$ kubectl describe ProcessingJob kmeans-mnist-processing
Name:         kmeans-mnist-processing
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"sagemaker.aws.amazon.com/
v1","kind":"ProcessingJob","metadata":{"annotations":{},"name":"kmeans-mnist-
processing",...
API Version:  sagemaker.aws.amazon.com/v1
Kind:         ProcessingJob
Metadata:
  Creation Timestamp:  2020-09-22T21:13:25Z
  Finalizers:
    sagemaker-operator-finalizer
```

```
  Generation:        2
  Resource Version:  21746658
  Self Link:         /apis/sagemaker.aws.amazon.com/v1/namespaces/default/processingjobs/
kmeans-mnist-processing
  UID:               7410ed52-fd18-11ea-b19a-165ae9f9e385
Spec:
  App Specification:
    Container Entrypoint:
      python
      /opt/ml/processing/code/kmeans_preprocessing.py
    Image Uri:  763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-training:1.5.0-cpu-
py36-ubuntu16.04
  Environment:
    Name:   MYVAR
    Value:  my_value
    Name:   MYVAR2
    Value:  my_value2
  Network Config:
  Processing Inputs:
    Input Name:  mnist_tar
    s3Input:
      Local Path:   /opt/ml/processing/input
      s3DataType:   S3Prefix
      s3InputMode:  File
      s3Uri:        s3://<s3bucket>-us-west-2/algorithms/kmeans/mnist/mnist.pkl.gz
    Input Name:     source_code
    s3Input:
      Local Path:   /opt/ml/processing/code
      s3DataType:   S3Prefix
      s3InputMode:  File
      s3Uri:        s3://<s3bucket>/mnist_kmeans_example/processing_code/
kmeans_preprocessing.py
  Processing Output Config:
    Outputs:
      Output Name:  train_data
      s3Output:
        Local Path:    /opt/ml/processing/output_train/
        s3UploadMode:  EndOfJob
        s3Uri:         s3://<s3bucket>/mnist_kmeans_example/output/
      Output Name:     test_data
      s3Output:
        Local Path:    /opt/ml/processing/output_test/
        s3UploadMode:  EndOfJob
        s3Uri:         s3://<s3bucket>/mnist_kmeans_example/output/
      Output Name:     valid_data
      s3Output:
        Local Path:    /opt/ml/processing/output_valid/
        s3UploadMode:  EndOfJob
        s3Uri:         s3://<s3bucket>/mnist_kmeans_example/output/
  Processing Resources:
    Cluster Config:
      Instance Count:     1
      Instance Type:      ml.m5.xlarge
      Volume Size In GB:  20
  Region:                 us-west-2
  Role Arn:               arn:aws:iam::<acct-id>:role/m-sagemaker-role
  Stopping Condition:
    Max Runtime In Seconds:  1800
  Tags:
    Key:    tagKey
    Value:  tagValue
Status:
  Cloud Watch Log URL:             https://us-west-2.console.aws.amazon.com/cloudwatch/
home?region=us-west-2#logStream:group=/aws/sagemaker/ProcessingJobs;prefix=kmeans-mnist-
processing-7410ed52fd1811eab19a165ae9f9e385;streamFilter=typeLogStreamPrefix
  Last Check Time:                 2020-09-22T21:14:29Z
```

```
  Processing Job Status:           InProgress
  Sage Maker Processing Job Name:  kmeans-mnist-processing-7410ed52fd1811eab19a165ae9f9e385
Events:                            <none>
```

删除处理作业

删除处理作业时， SageMaker 处理作业已从 Kubernetes 中删除，但该作业没有从 SageMaker 中删除。如果作业状态为 SageMaker 是`InProgress`作业已停止。处理停止的作业不会对 SageMaker 资源产生任何费用。使用以下命令之一删除处理作业。

对于集群范围内的安装：

```
kubectl delete processingjob kmeans-mnist-processing
```

对于命名空间范围内的安装：

```
kubectl delete processingjob kmeans-mnist-processing -n <NAMESPACE>
```

处理作业的输出应类似于以下内容。

```
processingjob.sagemaker.aws.amazon.com "kmeans-mnist-processing" deleted
```

> **Note**
>
> SageMaker 不会删除处理作业。停止的作业继续显示在 SageMaker 控制台。这些区域有：`delete`命令需要几分钟才能清除 SageMaker 的资源。

托管自动扩展策略 (HAP) 运营商

这些区域有： HostingAutoscalingPolicy (HAP) 运算符将资源 ID 列表作为输入，并将相同的策略应用于每个资源 ID。每个资源 ID 都是终端节点名称和变体名称的组合。HAP 运算符执行两个步骤：注册资源 ID，然后将扩展策略应用于每个资源 ID。`Delete`撤消这两个操作。您可以将 HAP 应用于现有 SageMaker 终端节点，也可以创建新的终端节点。 SageMaker 使用终端节点HostingDeployment 运算符. 你可以阅读更多关于 SageMaker 中的自动扩展应用程序 AutoScaling 策略.

> **Note**
>
> 在您的中`kubectl`命令，您可以使用短格式，`hap`，代替`hostingautoscalingpolicy`.

主题

- 创建 HostingAutoscalingPolicy 使用 YAML 文件 (p. 2577)
- 列出托管自动扩展策略 (p. 2579)
- 描述托管自动扩展策略 (p. 2580)
- 更新托管自动扩展策略 (p. 2580)
- 删除托管自动扩展策略 (p. 2580)
- 使用托管自动扩展策略更新或删除端点 (p. 2580)

创建 HostingAutoscalingPolicy 使用 YAML 文件

使用 YAML 文件创建 HostingAutoscalingPolicy (HAP) 将预定义或自定义指标应用于一个或多个指标 SageMaker 终端节点。

亚马逊 SageMaker 需要特定的值才能将自动缩放应用于您的变体。如果 YAML 规范中未指定这些值，HAP 运算符将应用以下默认值。

```
# Do not change
```

```
Namespace                     = "sagemaker"
# Do not change
ScalableDimension             = "sagemaker:variant:DesiredInstanceCount"
# Only one supported
PolicyType                    = "TargetTrackingScaling"
# This is the default policy name but can be changed to apply a custom policy
DefaultAutoscalingPolicyName = "SageMakerEndpointInvocationScalingPolicy"
```

使用以下示例创建 HAP，该 HAP 将预定义或自定义指标应用于一个或多个终端节点。

示例 1：将预定义指标应用于单个终端节点变体

1. 使用以下命令下载预定义指标的示例 YAML 文件：

   ```
   wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
   samples/hap-predefined-metric.yaml
   ```

2. 编辑 YAML 文件以指定`endpointName`、`variantName`, 和`region`.
3. 使用以下命令之一将预定义的指标应用于单个资源 ID（终端节点名称和变体名称组合）。

   对于集群范围内的安装：

   ```
   kubectl apply -f hap-predefined-metric.yaml
   ```

   对于命名空间范围内的安装：

   ```
   kubectl apply -f hap-predefined-metric.yaml -n <NAMESPACE>
   ```

示例 2：将自定义指标应用于单个终端节点变体

1. 使用以下命令下载自定义指标的示例 YAML 文件：

   ```
   wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
   samples/hap-custom-metric.yaml
   ```

2. 编辑 YAML 文件以指定`endpointName`、`variantName`, 和`region`.
3. 使用以下命令之一将自定义指标应用于单个资源 ID（终端节点名称和变体名称组合），而不是推荐的`SageMakerVariantInvocationsPerInstance`.

   Note

   亚马逊 SageMaker 不会检查您的 YAML 规范的有效性。

   对于集群范围内的安装：

   ```
   kubectl apply -f hap-custom-metric.yaml
   ```

   对于命名空间范围内的安装：

   ```
   kubectl apply -f hap-custom-metric.yaml -n <NAMESPACE>
   ```

示例 3：将扩展策略应用于多个终端节点和变体

您可以使用 HAP 运算符将相同的扩展策略应用于多个资源 ID。单独的`scaling_policy`将为每个资源 ID（终端节点名称和变体名称组合）创建请求。

1. 使用以下命令下载预定义指标的示例 YAML 文件：

```
wget https://raw.githubusercontent.com/aws/amazon-sagemaker-operator-for-k8s/master/
samples/hap-predefined-metric.yaml
```

2. 编辑 YAML 文件以指定`region`和多个`endpointName`和`variantName`有效值。
3. 使用以下命令之一将预定义的指标应用于多个资源 ID（终端节点名称和变体名称组合）。

   对于集群范围内的安装：

```
kubectl apply -f hap-predefined-metric.yaml
```

   对于命名空间范围内的安装：

```
kubectl apply -f hap-predefined-metric.yaml -n <NAMESPACE>
```

的注意事项 HostingAutoscalingPolicies 对于多个终端节点和变体

在使用多个资源 ID 时，请注意以下几点：

- 如果跨多个资源 ID 应用单个策略，则为每个资源 ID 创建一个 Policy ARN。五个端点有五个 Policy ARN。当你运行`describe`命令时，响应显示为一个作业，并包含单个作业状态。
- 如果将自定义指标应用于多个资源 ID，则对所有资源 ID（变体）值使用相同的维度或值。例如，如果您为实例 1-5 应用客户指标，并且终端节点变体维度映射到变体 1，则当变体 1 超出指标时，所有终端节点都将扩展或缩小。
- HAP 运算符支持更新资源 ID 列表。如果修改、添加或删除规范中的资源 ID，则自动扩缩策略将从上一个变体列表中删除，并应用于新指定的资源 ID 组合。使用`describe`命令列出策略当前应用到的资源 ID。

列出托管自动扩展策略

使用以下命令之一列出所有命令： HostingAutoscalingPolicies （HAP）使用 HAP 运营商创建。

对于集群范围内的安装：

```
kubectl get hap
```

对于命名空间范围内的安装：

```
kubectl get hap -n <NAMESPACE>
```

您的输出应类似于以下内容：

```
NAME             STATUS    CREATION-TIME
hap-predefined   Created   2021-07-13T21:32:21Z
```

使用以下命令检查您的状态 HostingAutoscalingPolicy （HAP）。

```
kubectl get hap <job-name>
```

返回以下值之一：

- `Reconciling`— 某些类型的错误将状态显示为`Reconciling`而不是`Error`. 一些示例是中的服务器端错误和终端节点`Creating`要么`Updating`状态。检查`Additional`有关更多详细信息，请参阅状态或操作符日志

- `Created`
- `Error`

查看应用策略的自动扩缩终端节点

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在左侧面板中，展开推理.
3. 选择终端节点.
4. 选择所需终端节点的名称。
5. 滚动到端点运行时设置部分。

描述托管自动扩展策略

使用以下命令获取有关 HostingAutoscalingPolicy （ HAP ）。这些命令通常用于调试问题或检查 HAP 的资源 ID（终端节点名称和变体名称组合）。

```
kubectl describe hap <job-name>
```

更新托管自动扩展策略

这些区域有： HostingAutoscalingPolicy (HAP) 操作员支持更新。您可以编辑 YAML 规范以更改值，然后重新应用策略。HAP 操作员删除现有策略并应用新策略。

删除托管自动扩展策略

使用以下命令之一删除 HostingAutoscalingPolicy （ HAP ）政策。

对于集群范围内的安装：

```
kubectl delete hap hap-predefined
```

对于命名空间范围内的安装：

```
kubectl delete hap hap-predefined -n <NAMESPACE>
```

此命令删除扩展策略并从 Kubernetes 中取消注册扩展目标。此命令将返回以下输出：

```
hostingautoscalingpolicies.sagemaker.aws.amazon.com "hap-predefined" deleted
```

使用托管自动扩展策略更新或删除端点

更新具有 HostingAutoscalingPolicy (HAP)，使用`kubectl delete`命令删除 HAP，更新终端节点，然后重新应用 HAP。

要删除具有 HAP 的终端节点，请使用`kubectl delete`命令在删除终端节点之前删除 HAP。

用于 Kubeflow 管道的 SageMaker 组件

本文档概述如何使用 SageMaker Kubeflow 管道 (KFP) 的组件。借助这些管道组件，您可以在 SageMaker 中创建和监控训练、调整、端点部署和批量转换作业。通过在 SageMaker 上运行 Kubeflow Pipeline 作业，您可以将数据处理和训练作业从 Kubernetes 集群移动到 SageMaker 的机器学习优化托管服务。本文档假设事先了解 Kubernetes 和 Kubeflow。

目录

- 什么是 Kubeflow 管道？ (p. 2581)
- Kubeflow 管道组件 (p. 2581)
- IAM 权限 (p. 2582)
- 将管道转换为使用 SageMaker (p. 2583)
- 使用 SageMaker 组件 (p. 2583)

什么是 Kubeflow 管道？

Kubeflow Pipeline (KFP) 是一个基于 Docker 容器构建和部署可移植、可扩展的机器学习 (ML) 工作流程的平台。Kubeflow Pipeline 平台包含：

- 用于管理和跟踪实验、作业和运行的用户界面 (UI)。
- 用于调度多步 ML 工作流程的引擎 (Argo)。
- 用于定义和操作管道和组件的 Python SDK。
- 用于使用 SDK 与系统进行交互的笔记本电脑。

管道是对机器学习工作流程的描述，表示为定向非循环图形如下图所示。 工作流程中的每一步都表示为 Kubeflow 管道组件，这是一个 Python 模块。

如果您的数据已经过预处理，则标准管道将获取数据的子集并运行模型的超参数优化。然后，管道使用最佳超参数训练具有完整数据集的模型。此模型用于批量推理和端点创建。

有关 Kubeflow 管道的更多信息，请参阅Kubeflow 管道文档.

Kubeflow 管道组件

Kubeflow Pipeline 组件是用于在 Kubeflow 管道中执行一个步骤的一组代码。组件由转换为 Docker 映像的 Python 模块表示。这些组件使用这些组件可以快速轻松地为实验和生产环境编写管道，而无需与底层的 Kubernetes 基础设施进行交互。

用于 Kubeflow 管道的 SageMaker 组件提供了什么？

用于 Kubeflow Pipeline 的 SageMaker 组件为在 SageMaker 中启动计算密集型作业提供了一种替代方案。这些组件集成 SageMaker 凭借 Kubeflow Pipeline 的可移植性和编排性。使用 SageMaker 组件，管道工作流程中的每个作业都在运行 SageMaker 而不是本地的 Kubernetes 集群。来自的作业参数、状态、日志和输出 SageMaker 仍然可以从 Kubeflow 管道 UI 访问。以下 SageMaker 已创建组件来集成六个关键 SageMaker 功能融入机器学习工作流程。您可以创建完全使用这些组件构建的 Kubeflow Pipeline，也可以根据需要将单个组件集成到工作流中。

使用不收取任何额外费用 SageMaker Kubeflow 管道的组件。您需要支付任何费用 SageMaker 你通过这些组件使用的资源。

训练组件

训练

训练组件允许您直接从 Kubeflow Pipeline 工作流程提交 SageMaker 培训作业。有关更多信息，请参阅 。SageMaker 培训 Kubeflow 管道组件.

超参数优化

使用超参数优化组件，您可以直接从 Kubeflow Pipeline 工作流程向 SageMaker 提交超参数调整作业。有关更多信息，请参阅 。SageMaker 超参数优化 Kubeflow 管道组件.

处理

使用 “处理” 组件，您可以将处理作业提交到 SageMaker 直接来自 Kubeflow Pipeline 工作流程。有关更多信息，请参阅 。SageMaker 处理 Kubeflow 管道组件.

推理组件

托管部署

通过部署组件，您可以通过 Kubeflow Pipeline 工作流程在 SageMaker Hosting 中部署模型。有关更多信息，请参阅 。SageMaker 托管服务-创建端点 Kubeflow 管道组件.

Batch 转换组件

使用批处理变换组件，您可以通过 Kubeflow Pipeline 工作流程为 SageMaker 中的整个数据集运行推理作业。有关更多信息，请参阅 。SageMaker Batch 转换 Kubeflow 管道组件.

组件 Ground Truth

Ground Truth使用 Ground Truth 组件，您可以直接从 Kubeflow Pipeline 工作流程提交 SageMaker Ground Truth 标签作业。有关更多信息，请参阅 。SageMaker Ground Truth Kubeflow 管道组件.

工作组

使用 Workteam 组件，您可以直接从 Kubeflow Pipeline 工作流程创建 SageMaker 私有工作组作业。有关更多信息，请参阅 。SageMaker 创建私有工作团队 Kubeflow Pipeline 组件.

IAM 权限

使用部署 Kubeflow 管道 SageMaker 组件需要以下三个级别的 IAM 权限：

- 用于访问您的IAM 用户/角色Amazon账户 (你的 _ 凭据)。注意：如果您已经有权访问 KFP Web UI 并在 Amazon S3 中拥有输入数据，或者您已经拥有带 KFP 的 Amazon Elastic Kubernetes 服务 (Amazon EKS) 集群，则根本不需要此功能。

  您可以使用网关节点（可以是本地计算机或远程实例）中的此用户/角色来执行以下操作：
  - 创建 Amazon EKS 集群并安装 KFP
  - 创建IAM 角色/用户
  - 为示例输入数据创建 Amazon S3 存储桶

  IAM 用户/角色需要以下权限：
  - CloudWatchLogsFullAccess
  - AWSCloudFormationFullAccess
  - IAMFullAccess
  - AmazonS3FullAccess
  - AmazonEC2FullAccess
  - AmazonEKSAdminPolicy（使用来自的模式创建此策略亚马逊 EKS 基于身份的策略示例）
- KFP Pod 用于访问的 IAM 角色 SageMaker (kfp 示例 pod-角色) KFP pod 使用此权限来创建 SageMaker 来自 KFP 组件的作业。 注意：如果要限制对 KFP pod 的权限，请创建自己的自定义策略并附加它。

  该角色需要以下权限：
  - AmazonSageMakerFullAccess
- 使用的 IAM 角色 SageMaker 用于访问诸如 Amazon S3 和 Amazon ECR 等资源的作业（kfp 示例-sagemaker 执行角色)。

  您的 SageMaker 作业使用此角色可以：
  - 访问 SageMaker 资源
  - 从 Amazon S3 输入数据
  - 将输出模型存储到 Amazon S3

  该角色需要以下权限：
  - AmazonSageMakerFullAccess

- AmazonS3FullAccess

这些是为 SageMaker 运行 KFP 组件所需的所有 IAM 用户/角色。

当你运行组件并创建了 SageMaker 终端节点，您还需要一个角色`sagemaker:InvokeEndpoint`查询推理端点的权限。

将管道转换为使用 SageMaker

您可以将现有管道转换为使用 SageMaker 通过移植你的通用 Python处理容器和训练容器. 如果您使用 SageMaker 为了推断，您还需要向集群附加 IAM 权限并将工件转换为模型。

使用 SageMaker 组件

在本教程中，你使用 SageMaker Kubeflow Pipeline 的组件，用于在 MNIST 数据集中使用 Kans 来训练分类模型。此工作流程使用 Kubeflow 管道作为管弦乐队，SageMaker 作为后端来运行工作流程中的步骤。有关此管道示例和其他管道示例的完整代码，请参阅示例 SageMaker Kubeflow 管道. 有关所使用组件的信息，请参阅KubeFlow 管道 GitHub 知识库.

目录

- 设置 (p. 2583)
- 运行 Kubeflow 管道 (p. 2589)

设置

要使用 Kubeflow Pipeline (KFP)，您需要一个 Amazon Elastic Kubernetes Service (Amazon EKS) 集群和一个网关节点来与该群集进行交互。以下各节显示了设置这些资源所需的步骤。

主题

- 设置网关节点 (p. 2583)
- 设置 Amazon EKS 集群 (p. 2584)
- 安装 Kubeflow 管道 (p. 2584)
- 访问 KFP 用户界面 (p. 2585)
- 为 KFP Pod 创建 IAM 用户/角色和 SageMaker 服务 (p. 2586)
- 添加对其他 IAM 用户或角色的访问 (p. 2588)

设置网关节点

网关节点用于创建 Amazon EKS 群集并访问 Kubeflow Pipeline UI。使用本地计算机或 Amazon EC2 实例作为网关节点。如果您想使用新的 Amazon EC2 实例，DLAMI 从Amazon控制台使用中的步骤启动和配置 DLAMI.

完成以下步骤以设置您的网关节点。根据您的环境，您可能已经配置了某些要求。

1. 如果您现有的 Amazon EKS 集群，则创建名为`your_credentials`使用中的步骤在您的中创建 IAM 用户Amazon账户。 如果您现有 Amazon EKS 集群，请使用 IAM 角色或具有访问权限的用户的凭证。
2. 使用中的步骤向您的用户添加以下权限：更改 IAM 用户的权限：

   - CloudWatchLogsFullAccess
   - AWSCloudFormationFullAccess
   - IAMFullAccess
   - AmazonS3FullAccess
   - AmazonEC2FullAccess
   - AmazonEKSAdminPolicy（使用来自的模式创建此策略亚马逊 EKS 基于身份的策略示例）

3. 在网关节点上安装以下内容以访问 Amazon EKS 群集和 KFP UI。

   - Amazon CLI. 如果您使用的是 IAM 用户，请配置访问密钥 ID、私有访问密钥和首选项Amazon通过运行的地区：`aws configure`
   - aws-iam-认证器版本 0.1.31 及更高版本
   - `eksctl`0.15 以上的版本。
   - `kubectl`（该版本需要在一个次要版本中与您的 Kubernetes 版本匹配）

4. 安装`boto3`.

```
pip install boto3
```

设置 Amazon EKS 集群

从网关节点的命令行运行以下步骤以设置 Amazon EKS 群集：

1. 如果您还没有现有的 Amazon EKS 集群，请完成以下子步骤。如果您已有 Amazon EKS 集群，请跳过此步骤。

   a. 从命令行运行以下命令以创建版本 1.17 或更高版本的 Amazon EKS 集群。Replace`<your-cluster-name>`有集群的任何名称。

   ```
   eksctl create cluster --name <your-cluster-name> --region us-east-1 --auto-
   kubeconfig --timeout=50m --managed --nodes=1
   ```

   b. 集群创建完成后，请使用以下命令验证您是否有权访问群集。

   ```
   kubectl get nodes
   ```

2. 验证当前`kubectl`text 是要与以下命令搭配使用的集群。当前上下文在输出中标有星号 (*)。

```
kubectl config get-contexts

CURRENT NAME     CLUSTER
*    <username>@<clustername>.us-east-1.eksctl.io    <clustername>.us-east-1.eksctl.io
```

3. 如果所需的群集未配置为当前默认集群，请使用以下命令更新默认集群。

```
aws eks update-kubeconfig --name <clustername> --region us-east-1
```

安装 Kubeflow 管道

从网关节点的命令行运行以下步骤，在集群上安装 Kubeflow Pipeline。

1. 按照以下步骤 1 在集群上安装 Kubeflow Pipeline部署 Kubeflow 管道文档. 您的 KFP 版本必须为 1.7.0 或更高版本。
2. 验证 Kubeflow Pipeline 服务和其他相关资源是否正在运行。

```
kubectl -n kubeflow get all | grep pipeline
```

您的输出应与以下内容类似。

```
pod/ml-pipeline-6b88c67994-kdtjv                          1/1     Running            0
     2d
pod/ml-pipeline-persistenceagent-64d74dfdbf-66stk         1/1     Running            0
     2d
```

```
pod/ml-pipeline-scheduledworkflow-65bdf46db7-5x9qj    1/1     Running            0
     2d
pod/ml-pipeline-ui-66cc4cffb6-cmsdb                   1/1     Running            0
     2d
pod/ml-pipeline-viewer-crd-6db65ccc4-wqlzj            1/1     Running            0
     2d
pod/ml-pipeline-visualizationserver-9c47576f4-bqmx4   1/1     Running            0
     2d
service/ml-pipeline                         ClusterIP   10.100.170.170   <none>
  8888/TCP,8887/TCP   2d
service/ml-pipeline-ui                       ClusterIP   10.100.38.71     <none>
  80/TCP              2d
service/ml-pipeline-visualizationserver      ClusterIP   10.100.61.47     <none>
  8888/TCP            2d
deployment.apps/ml-pipeline                          1/1     1            1            2d
deployment.apps/ml-pipeline-persistenceagent         1/1     1            1            2d
deployment.apps/ml-pipeline-scheduledworkflow        1/1     1            1            2d
deployment.apps/ml-pipeline-ui                       1/1     1            1            2d
deployment.apps/ml-pipeline-viewer-crd               1/1     1            1            2d
deployment.apps/ml-pipeline-visualizationserver      1/1     1            1            2d
replicaset.apps/ml-pipeline-6b88c67994                         1         1         1
 2d
replicaset.apps/ml-pipeline-persistenceagent-64d74dfdbf        1         1         1
 2d
replicaset.apps/ml-pipeline-scheduledworkflow-65bdf46db7       1         1         1
 2d
replicaset.apps/ml-pipeline-ui-66cc4cffb6                      1         1         1
 2d
replicaset.apps/ml-pipeline-viewer-crd-6db65ccc4               1         1         1
 2d
replicaset.apps/ml-pipeline-visualizationserver-9c47576f4      1         1         1
 2d
```

访问 KFP 用户界面

Kubeflow Pipeline UI 用于管理和跟踪集群上的实验、作业和运行情况。您可以使用端口转发从网关节点访问 Kubeflow Pipeline UI。

设置端口转发到 KFP UI 服务

从网关节点的命令行运行以下命令：

1. 使用以下命令验证 KFP UI 服务是否正在运行：

```
kubectl -n kubeflow get service ml-pipeline-ui

NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
ml-pipeline-ui   ClusterIP   10.100.38.71   <none>        80/TCP    2d22h
```

2. 运行以下命令以设置端口转发到 KFP UI 服务。这将 KFP UI 转发到网关节点上的 8080 端口，并允许您从浏览器访问 KFP UI。

```
kubectl port-forward -n kubeflow service/ml-pipeline-ui 8080:80
```

   如果没有活动，远程计算机转发的端口将丢弃。如果仪表板无法获取日志或更新，请再次运行此命令。如果命令返回错误，请确保您正在尝试使用的端口上没有进程正在运行。

访问 KFP UI 服务

访问 KFP UI 的方法取决于您的网关节点类型。

- 本地计算机作为网关节点

  1. 如下所示在浏览器中访问控制面板：

     ```
     http://localhost:8080
     ```

  2. 选择管道以访问管道 UI。
- 将 Amazon EC2 实例作为网关节点

  1. 您需要在 Amazon EC2 实例上设置 SSH 隧道，才能从本地计算机的浏览器访问 Kubeflow 控制面板。

     从本地计算机中的新终端会话中，运行以下命令。Replace<public-DNS-of-gateway-node>使用 Amazon EC2 控制台上找到的实例的 IP 地址。您也可以使用公有 DNS。Replace<path_to_key>使用用于访问网关节点的 pem 密钥的路径。

     ```
     public_DNS_address=<public-DNS-of-gateway-node>
     key=<path_to_key>

     on Ubuntu:
     ssh -i ${key} -L 9000:localhost:8080 ubuntu@${public_DNS_address}

     or on Amazon Linux:
     ssh -i ${key} -L 9000:localhost:8080 ec2-user@${public_DNS_address}
     ```

  2. 在浏览器中访问控制面板。

     ```
     http://localhost:9000
     ```

  3. 选择管道以访问 KFP 用户界面。

为 KFP Pod 创建 IAM 用户/角色和 SageMaker 服务

现在，您已经设置了 Kubeflow 的 Kubernetes 集群。要为 Kubeflow 管道运行 SageMaker 组件，Kubeflow 管道窗格需要访问 SageMaker。在本节中，您将创建 IAM 用户/角色，供 Kubeflow Pipeline pod 和 SageMaker 使用。

创建 KFP 执行角色

从网关节点的命令行运行以下命令：

1. 使用以下命令在 Amazon EKS 集群上启用 OIDC 支持。Replace<cluster_name>使用集群的名称和<cluster_region>使用您的集群所在的区域。

   ```
   eksctl utils associate-iam-oidc-provider --cluster <cluster-name> \
           --region <cluster-region> --approve
   ```

2. 运行以下命令以获取OIDC发布者 URL。表单为此 URLhttps://oidc.eks.<region>.amazonaws.com/id/<OIDC_ID>.

   ```
   aws eks describe-cluster --region <cluster-region> --name <cluster-name> --query
    "cluster.identity.oidc.issuer" --output text
   ```

3. 运行以下命令以创建名为trust.json. Replace<OIDC_URL>使用您的 OIDC 发行人 URL。请勿包括https://什么时候在你的 OIDC 发行人 URL 中。将 <AWS_account_number> 替换为您的 Amazon 账号。

   ```
   OIDC_URL="<OIDC-URL>"
   ```

```
AWS_ACC_NUM="<AWS-account-number>"

# Run this to create trust.json file
cat <<EOF > trust.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::${AWS_ACC_NUM}:oidc-provider/${OIDC_URL}"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "${OIDC_URL}:aud": "sts.amazonaws.com",
          "${OIDC_URL}:sub": "system:serviceaccount:kubeflow:pipeline-runner"
        }
      }
    }
  ]
}
EOF
```

4. 创建名为的 IAM 角色`kfp-example-pod-role`使用`trust.json`使用以下命令。KFP Pod 使用此角色来创建 SageMaker 来自 KFP 组件的作业。记下在输出中返回的 ARN。

```
aws iam create-role --role-name kfp-example-pod-role --assume-role-policy-document
 file://trust.json
aws iam attach-role-policy --role-name kfp-example-pod-role --policy-arn
 arn:aws:iam::aws:policy/AmazonSageMakerFullAccess
aws iam get-role --role-name kfp-example-pod-role --output text --query 'Role.Arn'
```

5. 使用以下命令编辑管道运行者服务帐户。

```
kubectl edit -n kubeflow serviceaccount pipeline-runner
```

6. 在文件中，添加以下 Amazon EKS 角色注释并替换`<role_arn>`ARN 与您的角色一起。

```
eks.amazonaws.com/role-arn: <role-arn>
```

7. 添加 Amazon EKS 角色注释后，您的文件应如下所示。保存该文件。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: <role-arn>
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":
{},"labels":{"app":"pipeline-runner","app.kubernetes.io/component":"pipelines-
runner","app.kubernetes.io/instance":"pipelines-runner-0.2.0","app.kubernetes.io/
managed-by":"kfctl","app.kubernetes.io/name":"pipelines-runner","app.kubernetes.io/
part-of":"kubeflow","app.kubernetes.io/version":"0.2.0"},"name":"pipeline-
runner","namespace":"kubeflow"}}
  creationTimestamp: "2020-04-16T05:48:06Z"
  labels:
    app: pipeline-runner
    app.kubernetes.io/component: pipelines-runner
    app.kubernetes.io/instance: pipelines-runner-0.2.0
    app.kubernetes.io/managed-by: kfctl
    app.kubernetes.io/name: pipelines-runner
    app.kubernetes.io/part-of: kubeflow
```

```
    app.kubernetes.io/version: 0.2.0
  name: pipeline-runner
  namespace: kubeflow
  resourceVersion: "11787"
  selfLink: /api/v1/namespaces/kubeflow/serviceaccounts/pipeline-runner
  uid: d86234bd-7fa5-11ea-a8f2-02934be6dc88
secrets:
- name: pipeline-runner-token-dkjrk
```

创建 SageMaker 执行角色

这些区域有：kfp-example-sagemaker-execution-roleSageMaker 作业使用 IAM 角色来访问 Amazon资源的费用。有关更多信息，请参阅 IAM 权限部分。在运行管道时，您可以将此角色作为输入参数提供。

运行以下命令以创建角色。记下输出中返回的 ARN。

```
SAGEMAKER_EXECUTION_ROLE_NAME=kfp-example-sagemaker-execution-role

TRUST="{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Effect\": \"Allow\", \"Principal
\": { \"Service\": \"sagemaker.amazonaws.com\" }, \"Action\": \"sts:AssumeRole\" } ] }"
aws iam create-role --role-name ${SAGEMAKER_EXECUTION_ROLE_NAME} --assume-role-policy-
document "$TRUST"
aws iam attach-role-policy --role-name ${SAGEMAKER_EXECUTION_ROLE_NAME} --policy-arn
 arn:aws:iam::aws:policy/AmazonSageMakerFullAccess
aws iam attach-role-policy --role-name ${SAGEMAKER_EXECUTION_ROLE_NAME} --policy-arn
 arn:aws:iam::aws:policy/AmazonS3FullAccess

aws iam get-role --role-name ${SAGEMAKER_EXECUTION_ROLE_NAME} --output text --query
 'Role.Arn'
```

添加对其他 IAM 用户或角色的访问

如果您使用像 Jupyter 这样的直观 IDE 或者希望组织中的其他人员使用您设置的集群，则还可以授予他们访问权限。在此工作流中运行以下步骤： SageMaker 笔记本。SageMaker 笔记本实例是一个完全托管的 Amazon EC2 计算实例，运行 Jupyter 笔记本应用程序。您可以使用笔记本实例创建和管理 Jupyter 笔记本以创建机器学习工作流。 您可以使用 KFP Python SDK 或 CLI 定义、编译、部署和运行管道。如果您使用的不是 SageMaker 要运行 Jupyter 的笔记本，您需要安装Amazon CLI和最新版本的kubectl.

1. 按中的步骤操作。创建 SageMaker 笔记本实例如果还没有 SageMaker 笔记本实例，请创建 SageMaker 笔记本实例。 为此实例指定 IAM 角色S3FullAccess权限。
2. Amazon EKS 集群使用 IAM 用户和角色来控制对集群的访问。这些规则在名为的配置映射中实现aws-auth. 只有拥有集群访问权限的用户/角色才能编辑此配置映射。从网关节点的命令行运行以下命令以获取您创建的笔记本实例的 IAM 角色。Replace<instance-name>使用实例的名称。

   ```
   aws sagemaker describe-notebook-instance --notebook-instance-name <instance-name> --
   region <region> --output text --query 'RoleArn'
   ```

   此命令在中输出 IAM 角色 ARNarn:aws:iam::<account-id>:role/<role-name>格式的日期和时间。记下此 ARN。
3. 运行以下命令以将策略附加到 IAM 角色。Replace<role-name>使用<role-name>在 ARN 中。

   ```
   aws iam attach-role-policy --role-name <role-name> --policy-
   arn arn:aws:iam::aws:policy/AmazonSageMakerFullAccess
   aws iam attach-role-policy --role-name <role-name> --policy-arn
    arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
   aws iam attach-role-policy --role-name <role-name> --policy-
   arn arn:aws:iam::aws:policy/AmazonS3FullAccess
   ```

4. eksctl提供了用于读取和编辑aws-auth配置映射。system:masters是默认用户组之一。将用户添加到此组。这些区域有：system:masters组对群集拥有超级用户权限。您还可以创建具有更严格权限的组，也可以直接将权限绑定到用户。Replace<IAM-Role-arn>使用 IAM 角色的 ARN。<your_username>可以是任何唯一的用户名。

```
eksctl create iamidentitymapping \
    --cluster <cluster-name> \
    --arn <IAM-Role-arn> \
    --group system:masters \
    --username <your-username> \
    --region <region>
```

5. 在 SageMaker 实例上打开 Jupyter 笔记本，然后运行以下命令以验证它是否具有对集群的访问权限。

```
aws eks --region <region> update-kubeconfig --name <cluster-name>
kubectl -n kubeflow get all | grep pipeline
```

运行 Kubeflow 管道

现在您的网关节点和 Amazon EKS 群集的设置已完成，您可以创建分类管道。要创建管道，你需要定义和编译它。然后，您可以部署它并使用它来运行工作流程。 您可以在 Python 中定义管道，然后使用 KFP 仪表板、KFP CLI 或 Python SDK 来编译、部署和运行工作流程。MNIST 分类管道示例的完整代码可在Kubeflow GitHub 存储库. 要使用它，请将示例 Python 文件克隆到网关节点。

主题

- 准备数据集 (p. 2589)
- 使用 SageMaker 组件创建 Kubeflow 管道 (p. 2589)
- 编译和部署管道 (p. 2590)
- 运行预测 (p. 2591)
- 查看结果和日志 (p. 2593)
- 清除 (p. 2593)

准备数据集

要运行管道，您需要将数据提取预处理脚本上传到 Amazon S3 存储桶。此存储桶和此示例的所有资源必须位于us-east-1亚马逊区域。如果您没有存储桶，请使用中的步骤创建存储桶。创建存储桶.

从mnist-kmeans-sagemaker在网关节点上克隆的 Kubeflow 存储库的文件夹，运行以下命令以上传kmeans_preprocessing.py将文件存储到 Amazon S3 存储桶。变更<bucket-name>到您创建的 Amazon S3 存储桶的名称。

```
aws s3 cp mnist-kmeans-sagemaker/kmeans_preprocessing.py s3://<bucket-name>/
mnist_kmeans_example/processing_code/kmeans_preprocessing.py
```

使用 SageMaker 组件创建 Kubeflow 管道

MNIST 分类管道的完整代码可在Kubeflow GitHub 存储库. 要使用它，请将示例 Python 文件克隆到网关节点。

输入参数

完整的 MNIST 分类管道具有特定于运行的参数，在创建运行时必须为其提供值。您必须为管道的每个组件提供这些参数。在使用其他管道时，也可以更新这些参数。 我们已为示例分类管道文件中的所有参数提供默认值。

以下是运行示例管道所需传递的唯一参数。要传递这些参数，请在创建新运行时更新它们的条目。

- 角色 ARN：这必须是 IAM 角色的 ARN，该角色 SageMaker 在您的中访问Amazonaccount. 使用的 ARN`kfp-example-pod-role`.
- 存储桶：这是您上传的 Amazon S3 存储桶的名称。`kmeans_preprocessing.py`文件到。

您可以使用 KFP UI 调整任何输入参数，然后再次触发运行。

编译和部署管道

在 Python 中定义管道之后，必须将管道编译为中间表示，然后才能将其提交到 Kubeflow Pipeline 服务。中间表示法是以压缩成 tar.gz 文件的 YAML 文件形式的工作流规范。 你需要 KFP SDK 来编译你的管道。

安装 KFP SDK

从网关节点的命令行运行以下命令：

1. 按照中的说明安装 KFP SDKKubeflow 管道文档.
2. 使用以下命令验证是否安装了 KFP SDK：

```
pip show kfp
```

3. 确认 。`dsl-compile`已正确安装如下：

```
which dsl-compile
```

编译您的管道

您有三种与 Kubeflow Pipeline 进行交互的选项：KFP UI、KFP CLI 或 KFP SDK。以下各节说明了使用 KFP UI 和 CLI 的工作流程。

从网关节点完成以下操作以编译管道。

1. 使用 Amazon S3 存储桶名称和 IAM 角色 ARN 修改 Python 文件。
2. 使用`dsl-compile`命令行来编译您的管道，如下所示。Replace`<path-to-python-file>`通往管道的路径以及`<path-to-output>`使用您希望 tar.gz 文件所在的位置。

```
dsl-compile --py <path-to-python-file> --output <path-to-output>
```

使用 KFP CLI 上传并运行管道

从网关节点的命令行完成以下步骤。KFP 将管道的运行作为实验组织。你可以选择指定实验名称。如果未指定一个，则运行将在下列出运行。默认值实验。

1. 按如下所述上载您的管道：

```
kfp pipeline upload --pipeline-name <pipeline-name> <path-to-output-tar.gz>
```

您的输出应与以下内容类似。记下`ID`.

```
Pipeline 29c3ff21-49f5-4dfe-94f6-618c0e2420fe has been submitted

Pipeline Details
```

```
------------------
ID           29c3ff21-49f5-4dfe-94f6-618c0e2420fe
Name         sm-pipeline
Description
Uploaded at  2020-04-30T20:22:39+00:00
...
...
```

2. 使用以下命令创建运行。KFP CLI run 命令目前不支持在创建运行时指定输入参数。在编译之前，你需要更新 Python 管道文件中的参数。Replace<experiment-name>和<job-name>有任何名字。Replace<pipeline-id>用你提交的管道的 ID。Replace<your-role-arn>使用的 ARNkfp-example-pod-role. Replace<your-bucket-name>使用您创建的 Amazon S3 存储桶的名称。

```
kfp run submit --experiment-name <experiment-name> --run-name <job-name> --pipeline-
id <pipeline-id> role_arn="<your-role-arn>" bucket_name="<your-bucket-name>"
```

   您还可以使用编译后的管道软件包直接提交运行，作为dsl-compile命令。

```
kfp run submit --experiment-name <experiment-name> --run-name <job-name> --package-
file <path-to-output> role_arn="<your-role-arn>" bucket_name="<your-bucket-name>"
```

   您的输出应与以下内容类似：

```
Creating experiment aws.
Run 95084a2c-f18d-4b77-a9da-eba00bf01e63 is submitted
+--------------------------------------+--------+----------+--------------------------
+
| run id                               | name   | status   | created at
  |
+======================================+========+==========+==========================
+
| 95084a2c-f18d-4b77-a9da-eba00bf01e63 | sm-job |          | 2020-04-30T20:36:41+00:00
 |
+--------------------------------------+--------+----------+--------------------------
+
```

3. 导航到 UI 以检查作业的进度。

使用 KFP UI 上传并运行管道

1. 在左侧面板中，选择管道选项卡。
2. 在右上角，选择+ 上传管道.
3. 输入管道名称和描述。
4. 选择上传文件然后输入您使用 CLI 或 Python SDK 创建的 tar.gz 文件的路径。
5. 在左侧面板中，选择管道选项卡。
6. 找到你创建的管道。
7. 选择+ 创意运行.
8. 输入你的输入参数。
9. 选择 Run（运行）。

运行预测

部署分类管道后，您可以对 Deploy 组件创建的终端节点运行分类预测。使用 KFP UI 检查输出工件sagemaker-deploy-model-endpoint_name. 下载 .tgz 文件以提取终端节点名称或在您使用的区域中检查 SageMaker 控制台。

配置权限以运行预测

如果要从网关节点运行预测，请跳过此部分。

1. 要使用任何其他计算机运行预测，请分配sagemaker:InvokeEndpoint对客户端计算机使用的 IAM 角色或 IAM 用户的权限。此权限用于运行预测。
2. 在网关节点上，运行以下命令以创建策略文件：

```
cat <<EoF > ./sagemaker-invoke.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:InvokeEndpoint"
            ],
            "Resource": "*"
        }
    ]
}
EoF
```

3. 将策略附加到客户端节点的 IAM 角色或 IAM 用户。
4. 如果您的客户端机器附加了 IAM 角色，请运行以下命令。Replace<your-instance-IAM-role>使用客户端节点的 IAM 角色的名称。Replace<path-to-sagemaker-invoke-json>使用您创建的策略文件的路径。

```
aws iam put-role-policy --role-name <your-instance-IAM-role> --policy-name sagemaker-
invoke-for-worker --policy-document file://<path-to-sagemaker-invoke-json>
```

5. 如果您的客户端计算机配置了 IAM 用户凭证，请运行以下命令。Replace<your_IAM_user_name>使用客户端节点的 IAM 用户的名称。Replace<path-to-sagemaker-invoke-json>使用您创建的策略文件的路径。

```
aws iam put-user-policy --user-name <your-IAM-user-name> --policy-name sagemaker-
invoke-for-worker --policy-document file://<path-to-sagemaker-invoke-json>
```

运行预测

1. 从你的客户端计算机创建名为的 Python 文件mnist-predictions.py包含以下内容。替换ENDPOINT_NAME变量。此脚本加载 MNIST 数据集，然后根据这些数字创建 CSV 并将其发送到终端节点进行预测。然后它输出结果。

```
import boto3
import gzip
import io
import json
import numpy
import pickle

ENDPOINT_NAME='<endpoint-name>'
region = boto3.Session().region_name

# S3 bucket where the original mnist data is downloaded and stored
downloaded_data_bucket = f"jumpstart-cache-prod-{region}"
downloaded_data_prefix = "1p-notebooks-datasets/mnist"

# Download the dataset
```

```
s3 = boto3.client("s3")
s3.download_file(downloaded_data_bucket, f"{downloaded_data_prefix}/mnist.pkl.gz",
 "mnist.pkl.gz")

# Load the dataset
with gzip.open('mnist.pkl.gz', 'rb') as f:
    train_set, valid_set, test_set = pickle.load(f, encoding='latin1')

# Simple function to create a csv from our numpy array
def np2csv(arr):
    csv = io.BytesIO()
    numpy.savetxt(csv, arr, delimiter=',', fmt='%g')
    return csv.getvalue().decode().rstrip()

runtime = boto3.Session(region).client('sagemaker-runtime')

payload = np2csv(train_set[0][30:31])

response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
                                   ContentType='text/csv',
                                   Body=payload)
result = json.loads(response['Body'].read().decode())
print(result)
```

2. 按如下方式运行 Python 文件：

```
python mnist-predictions.py
```

查看结果和日志

管道运行时，您可以选择任何组件来检查执行详细信息，例如输入和输出。这列出了已创建资源的名称。

如果成功处理了 KFP 请求并且 SageMaker 作业已创建，KFP UI 中的组件日志将提供指向 SageMaker 中创建的作业的链接。如果成功创建任务，也会提供 CloudWatch 日志。

如果你在同一群集上运行过多的管道作业，你可能会看到一条错误消息，指出你没有足够的 Pod 可用。要解决此问题，请登录网关节点，然后删除您未使用的管道创建的 Pod，如下所示：

```
kubectl get pods -n kubeflow
kubectl delete pods -n kubeflow <name-of-pipeline-pod>
```

清除

完成使用管道后，您需要清除您的资源。

1. 在 KFP 控制面板中，如果管道运行不能正确退出，请通过选择将其终止终止.
2. 如果终止选项不起作用，请登录网关节点并手动终止管道创建的所有 pod 运行，如下所示：

```
kubectl get pods -n kubeflow
kubectl delete pods -n kubeflow <name-of-pipeline-pod>
```

3. 使用Amazon帐户，请登录 SageMaker 服务。手动停止所有训练、批量转换和 HPO 作业。删除模型、数据存储桶和终端节点以避免产生任何额外成本。 终止管道运行并不会停止 SageMaker 中的作业。

亚马逊 SageMaker 常见问题

使用以下常见问题解答项目查找有关 mLOPS 工作流程的常见问题的答案。

问：我是否需要使用 SageMaker Python 开发工具包来创建 SageMaker 管道？

不， SageMaker 不需要 Python SDK 即可创建 SageMaker 管道。您还可以使用boto3要么Amazon CloudFormation. 创建管道需要管道定义，该定义是定义管道每个步骤的 JSON 对象。这些区域有：SageMaker SDK 提供了一种简单的方法来构建管道定义，您可以将其与之前提到的任何 API 一起使用，以创建管道本身。在不使用 SDK 的情况下，用户必须编写原始 JSON 定义才能创建管道，而不需要 SageMaker Python 开发工具包。要查看管道 JSON 定义的架构，请参阅 SageMaker JSON 管道定义 JSON 架构. 以下代码示例显示了一个示例。 SageMaker 管道定义 JSON 对象：

```
{'Version': '2020-12-01',
 'Metadata': {},
 'Parameters': [{'Name': 'ProcessingInstanceType',
   'Type': 'String',
   'DefaultValue': 'ml.m5.xlarge'},
  {'Name': 'ProcessingInstanceCount', 'Type': 'Integer', 'DefaultValue': 1},
  {'Name': 'TrainingInstanceType',
   'Type': 'String',
   'DefaultValue': 'ml.m5.xlarge'},
  {'Name': 'ModelApprovalStatus',
   'Type': 'String',
   'DefaultValue': 'PendingManualApproval'},
  {'Name': 'ProcessedData',
   'Type': 'String',
   'DefaultValue': 'S3_URL',
{'Name': 'InputDataUrl',
   'Type': 'String',
   'DefaultValue': 'S3_URL',
 'PipelineExperimentConfig': {'ExperimentName': {'Get': 'Execution.PipelineName'},
  'TrialName': {'Get': 'Execution.PipelineExecutionId'}},
 'Steps': [{'Name': 'ReadTrainDataFromFS',
   'Type': 'Processing',
   'Arguments': {'ProcessingResources': {'ClusterConfig': {'InstanceType': 'ml.m5.4xlarge',
      'InstanceCount': 2,
      'VolumeSizeInGB': 30}},
    'AppSpecification': {'ImageUri': 'IMAGE_URI',
     'ContainerArguments': [....]},
    'RoleArn': 'ROLE',
      'ProcessingInputs': [...],
    'ProcessingOutputConfig': {'Outputs': [.....]},
    'StoppingCondition': {'MaxRuntimeInSeconds': 86400}},
   'CacheConfig': {'Enabled': True, 'ExpireAfter': '30d'}},
   ...
   ...
   ...
  }
```

问：为什么我在我的中看到重新包装步骤？ SageMaker 管道？

当管道需要在压缩模型文件 (model.tar.gz) 中包含自定义脚本以上传到 Amazon S3 并用于将模型部署到 SageMaker 终端节点。何时 SageMaker 管道训练模型并将其注册到模型注册表中，它引入了重新包装步骤如果训练作业的训练模型输出需要包括自定义推理脚本。重新包装步骤解压模型，添加新脚本，然后重新压缩模型。运行管道将重新包装步骤添加为训练作业。

问：是否可以使用 SageMaker 使用实验 SageMaker 管道？

是. SageMaker 管道与原生集成 SageMaker 实验。您可以使用`PipelineExperimentConfig`创建管道并设置自己的管道时 SageMaker 实验名称。管道的每次运行都会创建一个试验，管道中的每个步骤都对应于`TrialComponent`在审判中。如果在实验配置中没有指定试用名称，则管道执行 ID 将用作试验名称。

```
pipeline = Pipeline(
   name=pipeline_name,
    parameters=[...],
    steps=[...],
    sagemaker_session=sagemaker_session,
    pipeline_experiment_config=PipelineExperimentConfig(
        ExecutionVariables.PIPELINE_NAME,
        ExecutionVariables.PIPELINE_EXECUTION_ID
    )
)
```

Q。 SageMaker 项目模板有一个模型部署存储库，它使用 CloudFormation (CFN) 来创建终端节点。有没有办法在不使用 CloudFormation 的情况下部署模型？

您可以在项目模板中自定义部署存储库，以便根据自己喜欢的方式从模型注册表中部署模型。例如，该模板使用 CloudFormation 创建实时终端节点。您可以更新部署以使用 SageMaker SDK、boto3 或任何其他可以创建终端节点而不是 CFN 的 API。如果您需要更新 CodeBuild 步骤作为部署管道的一部分，则可以创建自定义模板。

问：我们如何将模型文件 Amazon S3 URL 从火车步骤传递到模型注册步骤 SageMaker 在运行时管道？

您可以将模型位置引用为训练步骤的属性，如 end-to-end 示例CustomerChurn 管道在 Github 中。

问：如果我正在扩展预构建的容器来训练估计器还是为了`ProcessingStep`上 SageMaker 管道，是否有必要将脚本复制到 Dockerfile 中的容器中？

不可以，您可以将脚本复制到容器中，也可以通过`entry_point`参数（你的估计器实体的）或`code`参数（您的处理器实体），如以下代码示例所示。

```
step_process = ProcessingStep(
    name="PreprocessAbaloneData",
    processor=sklearn_processor,
    inputs = [
        ProcessingInput(
            input_name='dataset',
            source=...,
            destination="/opt/ml/processing/code",
        )
    ],
    outputs=[
        ProcessingOutput(output_name="train", source="/opt/ml/processing/train",
 destination = processed_data_path),
        ProcessingOutput(output_name="validation", source="/opt/ml/processing/validation",
 destination = processed_data_path),
        ProcessingOutput(output_name="test", source="/opt/ml/processing/test", destination
 = processed_data_path),
    ],
    code=os.path.join(BASE_DIR, "process.py"), ## Code is passed through an argument
    cache_config = cache_config,
    job_arguments = ['--input', 'arg1']
)

sklearn_estimator = SKLearn(
```

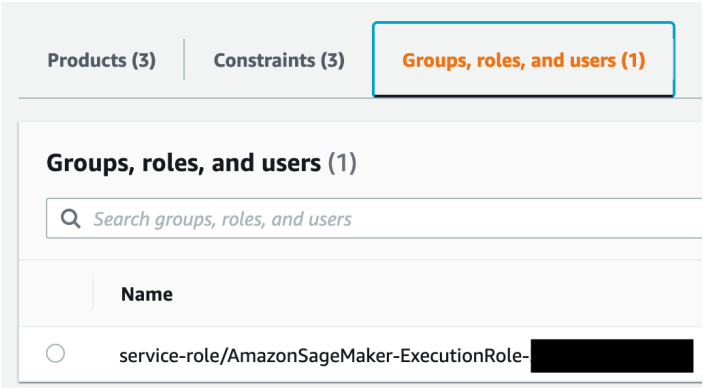
```
    entry_point=os.path.join(BASE_DIR, "train.py"), ## Code is passed through the
 entry_point
    framework_version="0.23-1",
    instance_type=training_instance_type,
    role=role,
    output_path=model_path, # New
    sagemaker_session=sagemaker_session, # New
    instance_count=1, # New
    base_job_name=f"{base_job_prefix}/pilot-train",
    metric_definitions=[
        {'Name': 'train:accuracy', 'Regex': 'accuracy_train=(.*?);'},
        {'Name': 'validation:accuracy', 'Regex': 'accuracy_validation=(.*?);'}
    ],
)
```

问：推荐使用哪种方法来管理不同的依赖关系 SageMaker 管道步骤？

您可以使用 SageMaker 用于实施映像构建 CI/CD 的项目模板。使用此模板，您可以自动执行构建并推送到 Amazon ECR 的映像的 CI/CD。项目的源代码控制存储库中容器文件的更改会启动 ML 管道并为容器部署最新版本。有关更多信息，请参阅博客。创建 Amazon SageMaker 带有图像构建CI/CD 管道的项目.

问：如何提供 SageMaker 项目访问亚马逊中特定用户配置文件 SageMaker Studio？

此后 SageMaker 项目由Amazon Service Catalog，必须将需要访问 SageMaker 项目的每个角色添加到亚马逊 SageMaker 解决方案和 ML Ops 产品服务目录中的产品组合。您可以在组、角色和用户选项卡，如下图所示。如果 Studio 中的每个用户配置文件都有不同的角色，则应将其中的每个角色添加到服务目录中。您也可以在 Studio 中创建用户配置文件时执行此操作。



问：我在哪里可以看到与每个属性相关的属性 SageMaker 管道步骤以便我可以在后续步骤中使用它们？

管道中的每个步骤都使用底层 SageMaker 适用于相应作业的 API。例如，`TrainingStep`调用`CreateTrainingJob`API 和步骤属性与来自的响应对应`DescribeTrainingJob`. 可以在以下方面的 API 参考链接中找到响应输出：DescribeTrainingJob. 你可以按照同样的过程获取属性TransSstep、处理步骤、TuningStep, 和创建模型步骤. 有关管道步骤的更多信息，请参阅管道步骤.

问：在 SageMaker 中重现我的模型的最佳方法是什么？

SageMaker 的世系跟踪服务在后端工作，以跟踪与模型训练和部署工作流程相关的所有元数据。这包括训练作业、使用的数据集、管道、端点和实际模型。您可以随时查询世系服务，以查找用于训练模型的确切工

件。使用这些工件，只要您可以访问已使用的确切数据集，就可以重新创建相同的机器学习工作流来重现模型。试用组件跟踪培训作业。此试用组件包含作为培训作业的一部分使用的所有参数。如果您不需要重新运行整个工作流程，则可以重现训练作业以获得相同的模型。

使用亚马逊 Augmented AI 进行人工审核

此功能在中国区域中不可用。

当您使用 Amazon Rekognition、Amazon Textract 或自定义机器学习 (ML) 模型等 AI 应用程序时，您可以使用 Amazon Audation Augmented AI 获得低置信度预测或随机预测样本的人工审核。

什么是 Amazon Augmented AI？

Amazon A2I 是一项服务，通过消除与构建人 Augmented AI 审核系统或管理大量人工审核人员相关的繁重工作，为所有开发人员带来人工学习预测的人工审核。

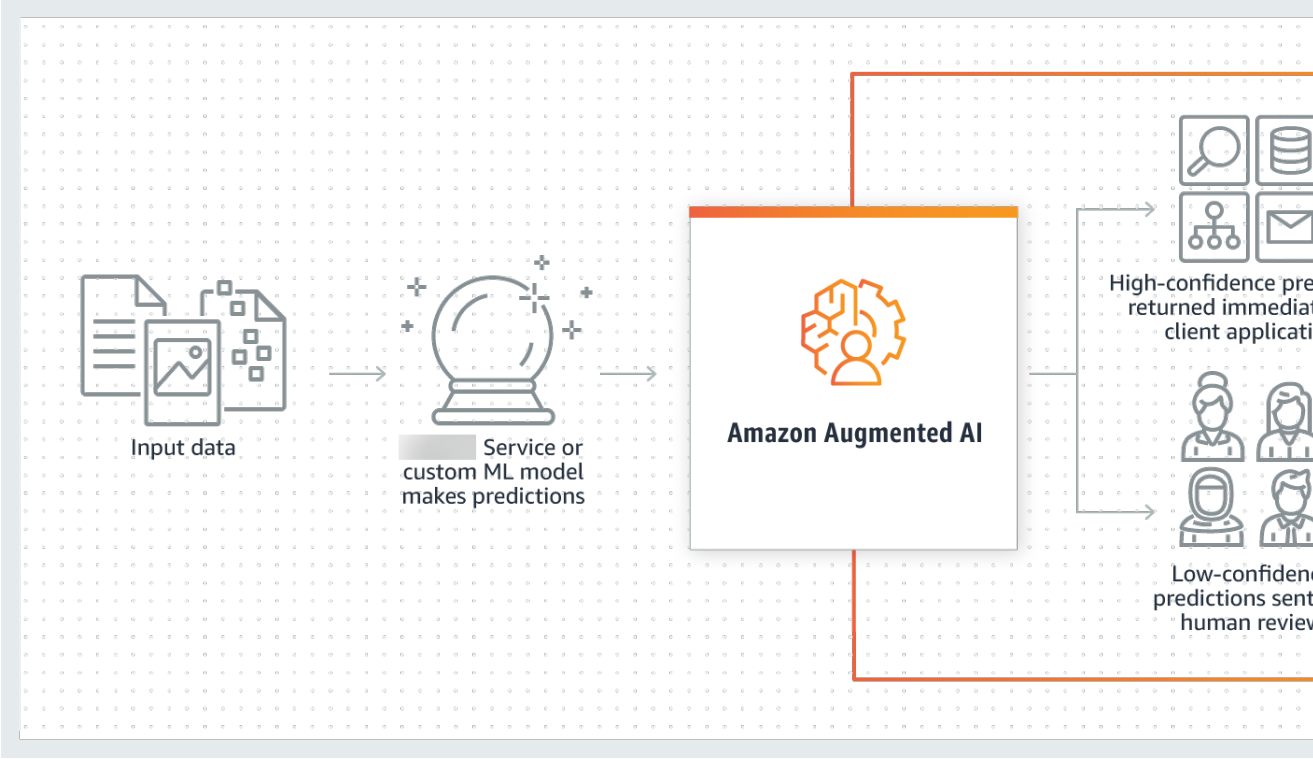
许多 ML 应用程序都要求人工审核低置信度预测，以确保结果正确无误。例如，由于扫描质量差或笔迹字迹差，从扫描的按揭申请表格中提取信息可能需要人工审核。建立人工审核系统既费时又成本高昂，因为它涉及实施复杂的流程或工作流，编写自定义软件来管理审阅任务和结果，以及管理大批审阅者。

Amazon A2I 简化了机器学习应用程序的人工评论的构建和管理。Amazon A2I 为常见 ML 用例提供内置的人工审核工作流程，例如内容审核和文档中的文本提取。您还可以为基于 SageMaker 或任何其他工具构建的 ML 模型创建自己的工作流程。借助 Amazon A2I，当模型无法进行高置信度预测或持续审计其预测时，您可以允许人工审核人员介入。

Amazon A2I 用例示例

以下示例演示了如何使用 Amazon A2I 将人工评论循环集成到机器学习应用程序中。对于这些示例中的每个例子，你都可以找到一个 Jupyter 笔记本展示了使用 Amazon A2I 使用案例和示例 (p. 2616).

- 将 Amazon A2I 与 Amazon Textract 一起使用— 让人类查看单页文档中的重要键值对，或者让 Amazon Textract 随机抽样并将数据集中的文档发送给人类进行审查。
- 使用亚马逊 A2I 与 Amazon Rekognition 一起使用— 如果 Amazon Rekognition 返回信心较低的分数，或者让 Amazon Rekognition 随机采样并将数据集中的图像发送给人类进行审查，则让人类查看不安全图片是否有明确的成人或暴力内容。
- 使用亚马逊 A2I 查看实时机器学习推理— 使用 Amazon A2I 来查看部署到 SageMaker 托管终端节点的模型所做的实时低信心推断，并使用 Amazon A2I 输出数据以增量方式训练模型。
- 将 Amazon A2I 与 Amazon Comprehend 一起使用— 让人类查看 Amazon Comprehend 关于文本数据的推论，例如情绪分析、文本语法和实体检测。
- 将 Amazon A2I 与 Amazon Transcribe 一起使用— 让人类查看 Amazon Transcribe 视频或音频文件的转录。使用转录人工评论循环的结果来创建自定义词汇并改善未来类似视频或音频内容的转录。
- 将 Amazon A2I 与 Amazon Translate 一起使用— 让人们审查从 Amazon Translate 返回的低信心翻译。
- 使用 Amazon A2I 审查表格数据— 使用 Amazon A2I 将人工评论循环集成到使用表格数据的机器学习应用程序中。



主题

- 开始使用亚马逊 Augmented AI (p. 2599)
- 使用 Amazon A2I 使用案例和示例 (p. 2616)
- 创建人工审核工作流程 (p. 2622)
- 删除人工审核工作流程 (p. 2639)
- 创建和启动人工循环 (p. 2640)
- 删除人工循环 (p. 2644)
- 创建和管理工作人员任务模板 (p. 2647)
- 监控和管理您的人工循环 (p. 2653)
- Amazon A2I 输出数据 (p. 2654)
- Amazon Augmented AI 中的权限和安全性 (p. 2662)
- 使用Amazon CloudWatch Events在 Amazon Augmented AI (p. 2667)
- 在 Amazon Augmented AI 中使用 API (p. 2669)

开始使用亚马逊 Augmented AI

要开始使用亚马逊 Augmented AI，请查看Amazon A2I 的核心组件 (p. 2600)和使用 Augmented AI 的先决条件 (p. 2603). 然后，使用以下文档了解如何使用 Amazon A2I 控制台和 API。

- 教程：从 Amazon A2I 控制台开始使用 (p. 2603)
- 教程：开始使用 Amazon A2I API (p. 2608)

您还可以通过遵循 Jupyter 笔记本教程来开始使用 Amazon A2I API。请参阅使用 Amazon A2I 使用案例和示例 (p. 2616)查看笔记本电脑和使用案例的列表。

Amazon A2I 的核心组件

查看以下术语以熟悉 Amazon A2I 的核心组件。

任务类型

您将 Amazon A2I 集成到的 AI/ML 工作流程定义了亚马逊 A2I任务类型.

Amazon A2I 支持：

- 两个内置任务类型：Amazon Textract 键值对提取和Amazon Rekognition 图像审核.
- 一个自定义任务类型：使用自定义任务类型将人工评论循环集成到任何机器学习工作流程。您可以使用自定义任务类型将 Amazon A2I 与其他任务类型集成AmazonAmazon Comprehend、Amazon Transcribe 和 Amazon Translate 等服务，以及您自己的自定义机器学习工作流程。要了解更多信息，请参阅 使用 Amazon A2I 使用案例和示例 (p. 2616)。

在下表中选择一个选项卡，以查看说明 Amazon A2I 如何处理每种任务类型的图表。使用上述列表中的链接选择任务类型页面以了解有关该任务类型的详细信息。

Amazon Textract – Key-value pair extraction

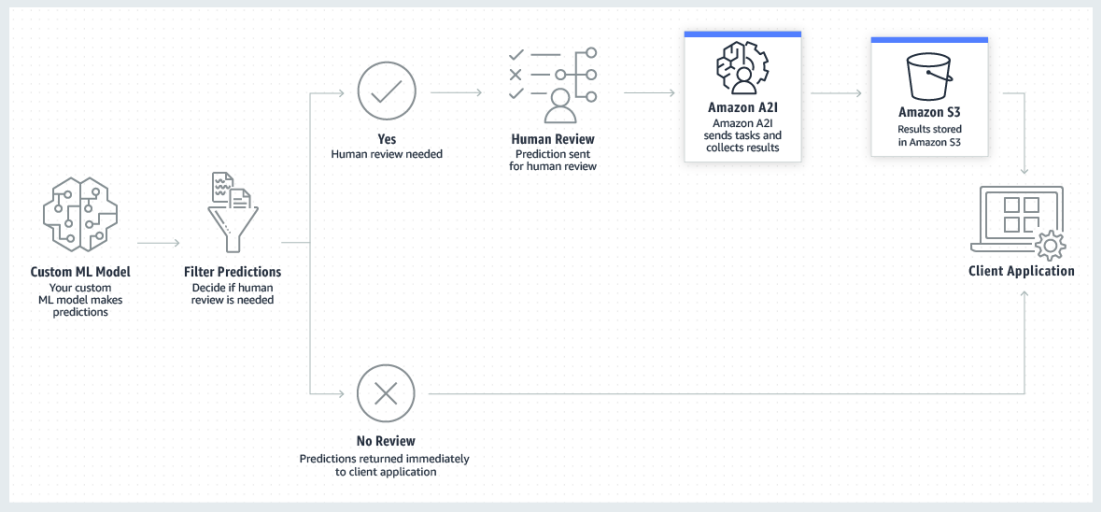
此图片描述了亚马逊 A2I 内置的 Amazon Textract 工作流程。左侧描述了创建 Amazon Textract 人工审核工作流程所需的资源：Amazon S3 存储桶、激活条件、工作人员任务模板和工作团队。这些资源用于创建人工审核工作流，或流定义。一个箭头指向工作流程的下一步：使用 Amazon Textract 通过人工审核工作流程配置人工循环。第二个箭头指向从此步骤直接指向满足人工审核工作流程中指定的激活条件的步骤。这启动了创建人工循环。在图像右侧，人类循环分三个步骤描述：1) 生成工作人员 UI 和工具，任务可供工作人员使用，2) 工作人员查看输入数据，最后 3) 结果保存在 Amazon S3 中。

Amazon Rekognition – Image moderation

此图片描述了 Amazon Rekognition 的亚马逊 A2I 内置工作流程。左侧描述了创建 Amazon Rekognition 人工审核工作流程所需的资源：Amazon S3 存储桶、激活条件、工作人员任务模板和工作团队。这些资源用于创建人工审核工作流，或流定义。一个箭头指向工作流程的下一步：使用 Amazon Rekognition 通过人工审核工作流程配置人工循环。第二个箭头指向从此步骤直接指向满足人工审核工作流程中指定的激活条件的步骤。这启动了创建人工循环。在图像右侧，人类循环分三个步骤描述：1) 生成工作人员 UI 和工具，任务可供工作人员使用，2) 工作人员查看输入数据，最后 3) 结果保存在 Amazon S3 中。

Custom Task Type

下图描述了 Amazon A2I 自定义工作流程。自定义机器学习模型用于生成预测。客户端应用程序使用用户定义的标准过滤这些预测，并确定是否需要人工审查。如果是，这些预测将发送给亚马逊 A2I 进行人工审查。Amazon A2I 在 Amazon S3 中收集人工审核结果，客户端应用程序可以访问该结果。如果筛选器确定不需要人工评论，则可以直接向客户应用程序提供预测。



人工审核工作流程（流程定义）

你可以使用人工审查工作流程来指定你的人工作团队，以使用工作人员界面人员任务模板，并提供有关工作人员应如何完成审查任务的信息。

对于内置任务类型，您还可以使用人工审核工作流来标识启动人工循环的条件。例如，Amazon Rekognition 可以使用机器学习执行图像内容审核。如果 Amazon Rekognition 的信度过低，您可以使用人工审核工作流指定向人员发送图像以进行内容审核。

您可以使用人工审核工作流创建多个人工循环。

您可以在 SageMaker 控制台中或使用 SageMaker API 创建流定义。要了解有关这些选项的更多信息，请参阅创建人工审核工作流程 (p. 2622)。

工作团队

一个工作团队是一群人类工作者，你将人工审查任务发送给他们。

在创建人工审核工作流程时，您需要指定单个工作组。

你的工作团队可以来自Amazon Mechanical Turk 员工，a供应商管理的员工，或者您自己的私有人力. 当你使用私人劳动力时，你可以创建多个工作团队。每个工作团队都可以在多个人工审核工作流程中使用。要了解如何创建人力和工作团队，请参阅创建和管理人力 (p. 463).

工作人员任务模板和人工任务 UI

你使用人员任务模板创建工作人员 UI (a)人工任务 UI) 为您的人工审核任务。

人工任务 UI 显示输入数据，例如文档或图像，以及工作人员的说明。它还提供了工作人员可用于完成任务的交互式工具。

对于内置任务类型，您必须使用为该任务类型提供的 Amazon A2I 工作人员任务模板。

人类循环

一个人类循环用于创建单个人类评论作业。对于每个人类评审工作，您可以选择发送的员工人数任务以查看单个数据对象。例如，如果将每个对象的工作人员数设置为3对于图像分类标记作业，有三个工作人员将分类每个输入图像 增加每个对象的工作人员数量可以提高标签准确性。

人类循环是使用人工审阅工作流程创建的，如下所示：

- 对于内置任务类型，人工审核工作流中指定的条件决定人工循环的创建时间。
- 人工审核任务将发送到人工审核工作流程中指定的工作团队。
- 人工审阅工作流程中指定的工作人员任务模板用于呈现人工任务 UI。

什么时候创建人类循环？

当你使用其中一个内置任务类型，相应的Amazon如果满足人工审核工作流中指定的条件，则服务会代表您创建并启动人工循环。 例如：

- 当您将 Augmented AI 与 Amazon Textract 结合使用时，您可以使用 API 操作将 Amazon A2I 集成到文档审核任务中`AnalyzeDocument`. 每当 Amazon Textract 返回关于符合您在人工审核工作流程中指定条件的键值对的推断时，都会创建人类循环。
- 当您将 Augmented AI 与 Amazon Rekognition 结合使用时，您可以使用 API 操作将 Amazon A2I 集成到图像审核任务中`DetectModerationLabels`. 每当 Amazon Rekognition 返回有关符合您在人工审核工作流程中指定条件的图片内容的推断时，都会创建人类循环。

使用自定义任务类型，您可以使用Amazon Augmented AI 运行时 API. 当您在自定义应用程序中调用 `StartHumanLoop` 时，任务将发送给人工审核员。

要了解如何创建和启动人工循环，请参阅创建和启动人工循环 (p. 2640)。

为了生成这些资源并创建人工审核工作流，Amazon A2I 集成了多个 API，包括 Amazon Augmented AI 运行时模型、SageMaker API 以及与您的任务类型关联的 API。要了解更多信息，请参阅 在 Amazon Augmented AI 中使用 API (p. 2669)。

> **Note**
>
> Amazon将 Augmented AI 能与其他配合使用时，区域可用性可Amazon服务，例如 Amazon Textract。同时创建 Augmented AI 资源Amazon您用于与其互的区域Amazon服务。适用于Amazon所有服务的区域可用性，请参阅区域表.

使用 Augmented AI 的先决条件

Amazon A2I 使用 IAM、SageMaker 和 Amazon S3 中的资源来创建和运行人工审核工作流程。创建人工审核工作流程时，您可以在 Amazon A2I 控制台中创建其中一些资源。要了解如何操作，请参阅教程：从 Amazon A2I 控制台开始使用 (p. 2603)。

要使用 Amazon A2I，您需要以下资源：

- 同一存储桶中的一个或多个 Amazon S3 存储桶Amazon区域作为输入和输出数据的工作流。要创建存储桶，请按照中的说明操作创建存储桶中的Amazon Simple Storage Storage Service 控制台.
- 具有创建人工审核工作流所需的权限的 IAM 角色，以及具有访问 Augmented AI 权限的 IAM 用户或角色。有关更多信息，请参阅 Amazon Augmented AI 中的权限和安全性 (p. 2662)。
- 针对人工审核工作流的公有、私有或供应商人力。如果您计划使用私有人力，则需要提前设置私有人力。Amazon区域为您的 Amazon A2I 工作流程。要了解有关这些人力类型的更多信息，请参阅 创建和管理人力 (p. 463)。

  Important

  要了解目前涵盖 Amazon Augmented AI 的合规性计划，请参阅Amazon合规性计划范围内的服务. 如果您与其他人一起使用亚马逊 Augmented AIAmazon服务（例如 Amazon Rekognition 和 Amazon Textract），请注意，亚马逊 Augmented AI 可能与其他服务不在同一合规性计划的范围内。您应对如何使用 Amazon Advanced Augmented AI 责，包括了解服务如何处理或存储客户数据以及对数据环境合规性的任何影响。你应该与你的讨论你的工作负载目标和目标Amazon客户团队；他们可以帮助您评估该服务是否适合您提议的使用案例和架构。

教程：从 Amazon A2I 控制台开始使用

以下教程向您展示了如何在亚马逊 A2I 控制台中开始使用 Amazon A2I。

本教程让您可以选择将 Augmented AI 与 Amazon Textract 结合使用进行文档审阅，或者使用 Amazon Rekognition 进行图片内容审核。

先决条件

要开始使用 Amazon A2I，请满足以下先决条件。

- 在同一个存储桶中创建 Amazon S3 存储桶Amazon区域作为输入和输出数据的工作流。例如，如果您在 us-east-1 中将 Amazon A2I 与 Amazon Textract 结合使用，请在 us-east-1 中创建存储桶。要创建存储桶，请按照中的说明操作创建存储桶中的Amazon Simple Storage Storage Service 控制台.
- 请执行下列操作之一：
  - 如果您想使用 Amazon Textract 完成教程，请下载此示例文档将其放在 Amazon S3 存储桶中。
  - 如果您想使用 Amazon Rekognition 完成教程，请下载映像将其放在 Amazon S3 存储桶中。

  Note

  Amazon A2I 控制台嵌入在 SageMaker 控制台中。

第 1 步：创建工作团队

首先，在 Amazon A2I 控制台中创建一个工作组，然后将自己添加为工作人员，以便您可以预览工作人员审核任务。

Important

本教程使用私人工作团队。Amazon A2I 私有劳动力在 SageMaker 控制台的 Ground Truth 区域中配置，并在 Amazon A2I 和 Ground Truth 之间共享。

使用工作人员电子邮件创建私有人力

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中，选择为人力添加标签下Ground Truth.
3. 选择 Private (私有)，然后选择 Create private team (创建私有团队)。
4. 选择 Invite new workers by email (通过电子邮件邀请新工作人员)。
5. 对于本教程，请输入您的电子邮件和任何其他您希望能够预览人工任务 UI 的其他电子邮件。在电子邮件地址框中粘贴或键入包含最多 50 个电子邮件的列表，以逗号分隔。
6. 输入组织名称和联系人电子邮件。
7. （可选）选择团队订阅到的 Amazon SNS 主题，这样在有新的 Ground Truth 标记作业可用时，其工作人员将收到通知。Amazon SNS 受 Ground Truth 支持，但不受支持，但不 Augmented AI 支持。如果您为工作人员订阅 Amazon SNS 通知，则他们仅收到有关 Ground Truth 标记作业的通知。而不会收到有关 Augmented AI 任务的通知。
8. 选择 Create private team (创建私有团队)。

如果您将自己添加到私人工作团队，则会收到来自的电子邮件`no-reply@verificationemail.com`带登录信息。使用此电子邮件中的链接重置密码并登录工作人员门户。当你创建人工循环时，你的人工审查任务就会出现在这里。

第 2 步：创建人工审核工作流程

在此步骤中，您将创建人工审核工作流程。每个人工审核工作流程都是针对特定的任务类型. 在本教程中，您可以选择内置任务类型：Amazon Rekognition 和 Amazon Textract。

要创建人工审核工作流程：

1. 在以下位置打开 Augmented AI 控制台https://console.aws.amazon.com/a2i访问人工审核工作流页.
2. Select创建人工审核工作流.
3. In工作流设置中，输入工作流程名称、S3 bucket，以及IAM 角色您为本教程创建的，使用Amazon管理的策略`AmazonAugmentedAIIntegratedAPIAccess`已附加。
4. 适用于任务类型，请选择Textract — 键值对提取要么Rekognition — 图像审核.
5. 选择从下表中选择的任务类型，以获取该任务类型的说明。

   Amazon Textract – Key-value pair extraction

   1. Select根据表单键置信度分数或特定表单键缺失时触发针对特定表单键的人工审查。.

   2. 适用于键名称，输入`Mail Address`.

   3. 在之间设置识别置信度阈值`0`和`99`.

   4. 设置资格置信度阈值`0`和`99`.

   5. Select触发器针对由 Amazon Textract 标识的所有表单键的人工审查，并在特定范围内的置信度分数。.

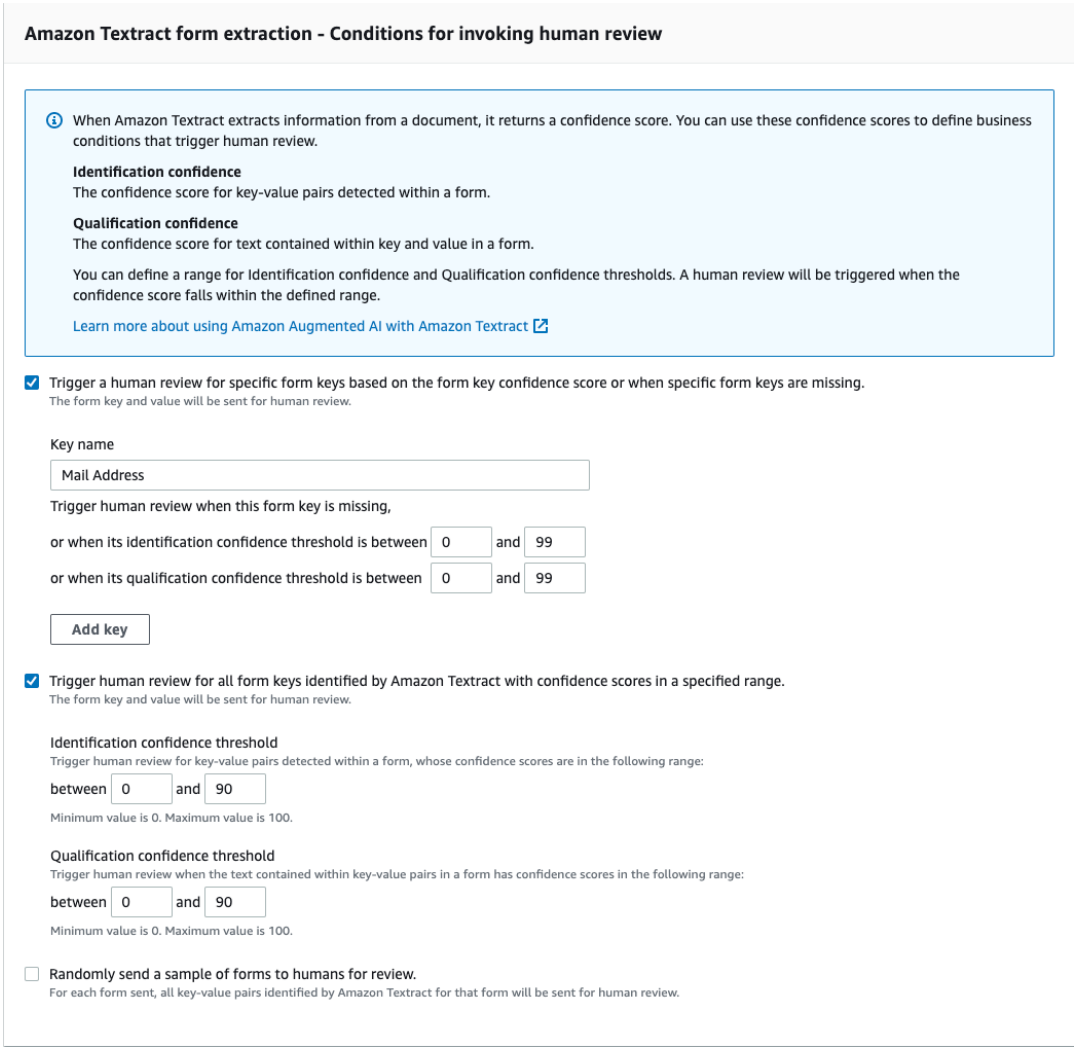
   6. 在之间设置识别置信度阈值`0`和`90`.

   7. 设置资格置信度阈值`0`和`90`.

   如果 Amazon Textract 返回的信心分数低于`99`为了`Mail Address`和它的关键，或者如果它返回的信心分数低于`90`用于文档中检测到的任何密钥值对。

   下图显示了 Amazon Textract 表单提取-亚马逊 A2I 控制台中调用人工评论的条件部分。在图片中，选中了上一段中解释的两种触发器类型的复选框，并且`Mail Address`被用作键名称对于第一个触发器。标识置信度阈值是使用表单中检测到的键值对的置信度分数来定义的，并设置在 0 到 99 之

间。资格置信度阈值是使用键中包含的文本的置信度分数和表单中的值来定义的，并设置在 0 到 99 之间。



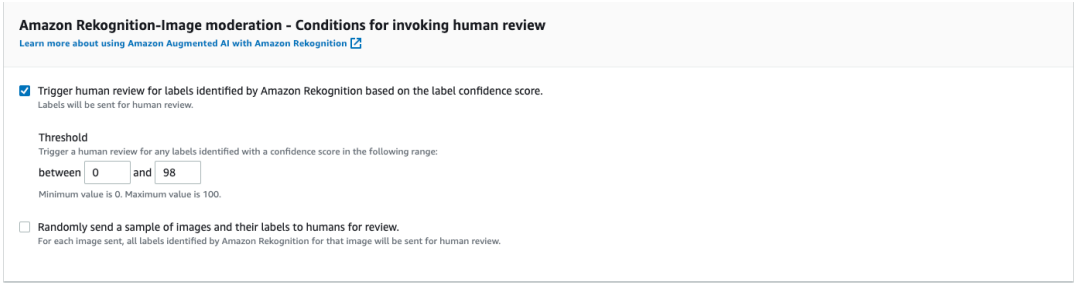
Amazon Rekognition – Image moderation

1. Select触发器根据标签置信度分数，针对由 Amazon Rekognition 标识的标签的人工审查.

2. 设置Threshold之间0和98.

如果 Amazon Rekognition 返回的信心分数低于98用于图像审核工作。

下图说明了如何选择根据标签置信度分数触发针对由 Amazon Rekognition 标识的标签的人工审查。选项然后输入Threshold在亚马逊 A2I 控制台中的 0 到 98 之间。



6. UDER创建员工任务模板，请选择从默认模板创建.
7. 输入模板名称.
8. In任务描述字段中，输入以下文本：

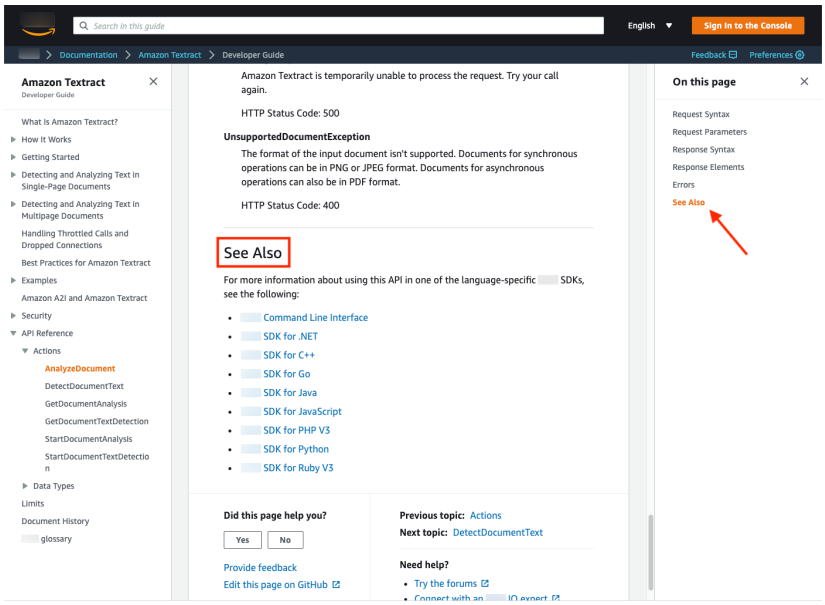
```
Read the instructions carefully and complete the task.
```

9. UDER员工，请选择私密.
10. 选择您创建的私人团队。
11. 选择创建。

创建人工审核工作流程后，它将显示在人工审核工作流页. 当状态是Active中，复制并保存工作流 ARN。下一个步骤中，您需要用到它。

第 3 步：启动人工循环

您必须使用 API 操作启动人工循环。您可以使用各种特定于语言的 SDK 与这些 API 操作进行交互。要查看每个 SDK 的文档，请参阅另请参阅部分，如下图所示。



在本教程中，您将使用以下 API 之一：

- 如果您选择了 Amazon Textract 任务类型，则使用AnalyzeDocumentoperation.
- 如果您选择了 Amazon Rekognition 任务类型，则使用DetectModerationLabelsoperation.

您可以使用 SageMaker 笔记本实例（建议新用户使用）或Amazon Command Line Interface(Amazon CLI)。选择下列选项之一，以了解有关这些选项的更多信息：

- 要了解有关并设置笔记本实例的更多信息，请参阅使用 Amazon SageMaker 笔记本实例 (p. 147).
- 要详细了解相关内容并开始使用Amazon CLI，请参阅什么是？Amazon命令行界面？中的Amazon Command Line Interface用户指南.

在下表中选择您的任务类型，以查看 Amazon Textract 和 Amazon Rekognition 的示例请求Amazon SDK for Python (Boto3).

Amazon Textract – Key-value pair extraction

以下示例使用Amazon SDK for Python (Boto3)调用`analyze_document`(在 us-west-2 中)。将斜体红色文本替换为斜体的红色文本。加入`DataAttributes`参数如果您使用的是 Amazon Mechanical Turk 员队伍。有关更多信息，请参阅 。`analyze_document`(在文档中)Amazon SDK for Python (Boto)API 参考.

```
   response = client.analyze_document(
          Document={
                 "S3Object": {
                     "Bucket": "AWSDOC-EXAMPLE-BUCKET",
                     "Name": "document-name.pdf"
                 }
          },
          HumanLoopConfig={
             "FlowDefinitionArn":"arn:aws:sagemaker:us-west-2:111122223333:flow-
definition/flow-definition-name",
             "HumanLoopName":"human-loop-name",
             "DataAttributes" : {
                 "ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation","FreeOfAdultContent"]
             }
          },
          FeatureTypes=["TABLES", "FORMS"])
```

Amazon Rekognition – Image moderation

以下示例使用Amazon SDK for Python (Boto3)调用`detect_moderation_labels`(在 us-west-2 中)。将斜体红色文本替换为斜体的红色文本。加入`DataAttributes`参数如果您使用的是 Amazon Mechanical Turk 员队伍。有关更多信息，请参阅 。`detect_moderation_labels`(在) 中的文档Amazon SDK for Python (Boto)API 参考.

```
   response = client.detect_moderation_labels(
             Image={
                 "S3Object":{
                     "Bucket": "AWSDOC-EXAMPLE-BUCKET",
                     "Name": "image-name.png"
                 }
             },
             HumanLoopConfig={
                "FlowDefinitionArn":"arn:aws:sagemaker:us-west-2:111122223333:flow-
definition/flow-definition-name",
                "HumanLoopName":"human-loop-name",
                "DataAttributes":{
                     ContentClassifiers:
["FreeOfPersonallyIdentifiableInformation"|"FreeOfAdultContent"]
                 }
              })
```

第 4 步：在控制台中查看人类循环状态

当您启动人类循环时，您可以在 Amazon A2I 控制台中查看其状态。

查看你的人类循环状态

1. 在以下位置打开 Augmented AI 控制台https://console.aws.amazon.com/a2i访问人工审核工作流页.
2. 选择用于启动人工循环的人工审查工作流程。

3. 在人类循环部分，你可以看到你的人类循环。在状态column.

第 5 步：下载输出数据

输出数据存储在您创建人工审核工作流程时指定的 Amazon S3 存储桶中。

要查看您的 Amazon A2I 输出数据

1. 打开 Amazon S3 控制台。
2. 在本示例的步骤 2 中，选择您在创建人工审核工作流程时指定的 Amazon S3 存储桶。
3. 从以人工审核工作流程命名的文件夹开始，通过选择具有以下命名约定的文件夹来导航到输出数据：

```
s3://output-bucket-specified-in-human-review-workflow/human-review-workflow-
name/YYYY/MM/DD/hh/mm/ss/human-loop-name/output.json
```

4. Select`output.json`然后选择下载.

教程：开始使用 Amazon A2I API

本教程介绍了您可以用来开始使用 Amazon A2I 的 API 操作。

要使用 Jupyter 笔记本运行这些操作，请从使用 Amazon A2I 使用案例和示例 (p. 2616)使用将 SageMaker 笔记本实例与 Amazon A2I Jupyter 笔记本电脑一起使用 (p. 2617)了解如何在 SageMaker 笔记本实例中使用它。

要了解有关可用于 Amazon A2I 的 API 操作的更多信息，请参阅在 Amazon Augmented AI 中使用 API (p. 2669).

创建私有人工作团队

您可以创建一个私人工作团队并将自己添加为员工，以便您可以预览 Amazon A2I。

如果您不熟悉 Amazon Cognito，我们建议您使用 SageMaker 控制台创建私人劳动力并将自己添加为私人员工。有关说明，请参阅第 1 步：创建工作团队 (p. 2603)。

如果您熟悉 Amazon Cognito，可以使用以下说明使用 SageMaker API 创建私有工作团队。创建工作团队后，请注意工作团队 ARN (`WorkteamArn`)。

要了解有关私有人力和其他可用配置的更多信息，请参阅使用私有人力 (p. 467).

创建私有人力

如果您尚未创建私有人力，则可以使用Amazon Cognito 用户池. 确保您已将自己添加到此用户池。您可以使用Amazon SDK for Python (Boto3) `create_workforce`function. 有关其他特定于语言的 SDK，请参阅中的列表创建劳动力.

```
response = client.create_workforce(
    CognitoConfig={
        "UserPool": "Pool_ID",
        "ClientId": "app-client-id"
    },
    WorkforceName="workforce-name"
)
```

创建私人工作团队

在中创建私人劳动力之后Amazon用于配置和启动人工循环的区域，您可以使用Amazon SDK for Python (Boto3) `create_workteam`function. 有关其他特定于语言的 SDK，请参阅中的列表`CreateWorkteam`.

```
response = client.create_workteam(
    WorkteamName="work-team-name",
    WorkforceName= "workforce-name",
    MemberDefinitions=[
        {
            "CognitoMemberDefinition": {
                "UserPool": "<aws-region>_ID",
                "UserGroup": "user-group",
                "ClientId": "app-client-id"
            },
        }
    ]
)
```

按如下方式访问您的工作团队 ARN：

```
workteamArn = response["WorkteamArn"]
```

列出账户中的私人工作团队

如果您已创建私有工作组，则可以列出指定的所有工作团队。Amazon使用账户中的区域Amazon SDK for Python (Boto3) `list_workteams`function. 有关其他特定于语言的 SDK，请参阅中的列表`ListWorkteams`.

```
response = client.list_workteams()
```

如果您的帐户中有许多工作团队，则可能需要使用`MaxResults`、`SortBy`, 和`NameContains`来筛选结果。

创建人工审核工作流程

您可以使用 Amazon A2I 创建人工审核工作流。`CreateFlowDefinition`operation. 在创建人工审核工作流程之前，您需要创建人工任务 UI。您可使用`CreateHumanTaskUi`operation.

如果您将亚马逊 A2I 与 Amazon Textract 或 Amazon Rekognition 集成一起使用，则可以使用 JSON 指定激活条件。

创建人工任务 UI

如果您正在创建人工审核工作流程以与 Amazon Textract 或 Amazon Rekognition 集成一起使用，则需要使用和修改预制的工作人员任务模板。对于所有自定义集成，您可以使用自己的自定义工作人员任务模板。使用下表了解如何使用两个内置集成的工作人员任务模板创建人工任务 UI。将模板替换为您自己的模板，以自定义此请求。

Amazon Textract – Key-value pair extraction

要详细了解相关此模板，请参阅Amazon Textract 的自定义模板示例 (p. 2649).

Amazon Rekognition – Image moderation

要详细了解相关此模板，请参阅Amazon Rekognition 的自定义模板示例 (p. 2650).

Custom Integration

以下是可在自定义集成中使用的示例模板。在此模板中使用了此模板。笔记本，展示了与 Amazon Comprehend 的自定义集成。

使用上面指定的模板，您可以使用Amazon SDK for Python (Boto3) `create_human_task_ui`function. 有关其他特定于语言的 SDK，请参阅中的列表`CreateHumanTaskUi`.

```
response = client.create_human_task_ui(
    HumanTaskUiName="human-task-ui-name",
    UiTemplate={
        "Content": template
    }
)
```

此响应元素包含人工任务 UI ARN。按如下方式保存此内容：

```
humanTaskUiArn = response["HumanTaskUiArn"]
```

创建 JSON 以指定激活条件

对于 Amazon Textract 和 Amazon Rekognition 内置集成，您可以将激活条件保存在 JSON 对象中，然后在`CreateFlowDefinition`请求.

接下来，选择一个选项卡以查看可用于这些内置集成的示例激活条件。有关激活条件选项的其他信息，请参阅亚马逊中用于人 Augmented AI 循环激活条件的 JSON 架构 (p. 2628).

Amazon Textract – Key-value pair extraction

此示例指定了特定密钥的条件（例如`Mail address`) (在文档中)。如果 Amazon Textract 的信心超出此处设定的阈值，则会将文档发送给人员进行审核，启动人工循环的特定密钥将提示给工作人员。

```
     import json

     humanLoopActivationConditions = json.dumps(
       {
           "Conditions": [
               {
                 "Or": [

                   {
                       "ConditionType": "ImportantFormKeyConfidenceCheck",
                       "ConditionParameters": {
                           "ImportantFormKey": "Mail address",
                           "ImportantFormKeyAliases": ["Mail Address:","Mail
address:", "Mailing Add:","Mailing Addresses"],
                           "KeyValueBlockConfidenceLessThan": 100,
                           "WordBlockConfidenceLessThan": 100
                       }
                   },
                   {
                       "ConditionType": "MissingImportantFormKey",
                       "ConditionParameters": {
                           "ImportantFormKey": "Mail address",
                           "ImportantFormKeyAliases": ["Mail Address:","Mail
address:","Mailing Add:","Mailing Addresses"]
                       }
                   },
```

```
                    {
                        "ConditionType": "ImportantFormKeyConfidenceCheck",
                        "ConditionParameters": {
                            "ImportantFormKey": "Phone Number",
                            "ImportantFormKeyAliases": ["Phone number:", "Phone No.:",
"Number:"],
                            "KeyValueBlockConfidenceLessThan": 100,
                            "WordBlockConfidenceLessThan": 100
                        }
                    },
                    {
                      "ConditionType": "ImportantFormKeyConfidenceCheck",
                      "ConditionParameters": {
                        "ImportantFormKey": "*",
                        "KeyValueBlockConfidenceLessThan": 100,
                        "WordBlockConfidenceLessThan": 100
                      }
                    },
                    {
                      "ConditionType": "ImportantFormKeyConfidenceCheck",
                      "ConditionParameters": {
                        "ImportantFormKey": "*",
                        "KeyValueBlockConfidenceGreaterThan": 0,
                        "WordBlockConfidenceGreaterThan": 0
                      }
                    }
            ]
        }
            ]
        }
    )
```

Amazon Rekognition – Image moderation

此处使用的人类循环激活条件是针对 Amazon Rekognition 内容审核量身定制的；它们基于`Suggestive`和`Female Swimwear Or Underwear`审核标签。

```
        import json

        humanLoopActivationConditions = json.dumps(
        {
            "Conditions": [
                {
                  "Or": [
                    {
                        "ConditionType": "ModerationLabelConfidenceCheck",
                        "ConditionParameters": {
                            "ModerationLabelName": "Suggestive",
                            "ConfidenceLessThan": 98
                        }
                    },
                    {
                        "ConditionType": "ModerationLabelConfidenceCheck",
                        "ConditionParameters": {
                            "ModerationLabelName": "Female Swimwear Or Underwear",
                            "ConfidenceGreaterThan": 98
                        }
                    }
                  ]
                }
            ]
        }
    )
```

创建人工审核工作流程

本节提供了一个示例，说明`CreateFlowDefinition` Amazon SDK for Python (Boto3)请求使用前面章节中创建的资源。有关其他特定于语言的 SDK，请参阅中的列表CreateFlowDefinition. 使用下表中的选项卡查看为 Amazon Textract 和 Amazon Rekognition 内置集成创建人工审核工作流程的请求。

Amazon Textract – Key-value pair extraction

如果您使用与 Amazon Textract 的内置集成，则必须指定`"AWS/Textract/AnalyzeDocument/Forms/V1"`为了`"AwsManagedHumanLoopRequestSource"`在`HumanLoopRequestSource`.

```
    response = client.create_flow_definition(
        FlowDefinitionName="human-review-workflow-name",
        HumanLoopRequestSource={
            "AwsManagedHumanLoopRequestSource": "AWS/Textract/AnalyzeDocument/Forms/V1"
        },
        HumanLoopActivationConfig={
            "HumanLoopActivationConditionsConfig": {
                "HumanLoopActivationConditions": humanLoopActivationConditions
            }
        },
        HumanLoopConfig={
            "WorkteamArn": workteamArn,
            "HumanTaskUiArn": humanTaskUiArn,
            "TaskTitle": "Document entry review",
            "TaskDescription": "Review the document and instructions. Complete the
 task",
            "TaskCount": 1,
            "TaskAvailabilityLifetimeInSeconds": 43200,
            "TaskTimeLimitInSeconds": 3600,
            "TaskKeywords": [
                "document review",
            ],
        },
        OutputConfig={
            "S3OutputPath": "s3://DOC-EXAMPLE-BUCKET/prefix/",
        },
        RoleArn="arn:aws:iam::<account-number>:role/<role-name>",
        Tags=[
            {
                "Key": "string",
                "Value": "string"
            },
        ]
    )
```

Amazon Rekognition – Image moderation

如果您使用与 Amazon Rekognition 的内置集成，则必须指定`"AWS/Rekognition/DetectModerationLabels/Image/V3"`为了`"AwsManagedHumanLoopRequestSource"`在`HumanLoopRequestSource`.

```
    response = client.create_flow_definition(
        FlowDefinitionName="human-review-workflow-name",
        HumanLoopRequestSource={
            "AwsManagedHumanLoopRequestSource": "AWS/Rekognition/
DetectModerationLabels/Image/V3"
        },
        HumanLoopActivationConfig={
            "HumanLoopActivationConditionsConfig": {
```

```
                "HumanLoopActivationConditions": humanLoopActivationConditions
            }
        },
        HumanLoopConfig={
            "WorkteamArn": workteamArn,
            "HumanTaskUiArn": humanTaskUiArn,
            "TaskTitle": "Image content moderation",
            "TaskDescription": "Review the image and instructions. Complete the task",
            "TaskCount": 1,
            "TaskAvailabilityLifetimeInSeconds": 43200,
            "TaskTimeLimitInSeconds": 3600,
            "TaskKeywords": [
                "content moderation",
            ],
        },
        OutputConfig={
            "S3OutputPath": "s3://DOC-EXAMPLE-BUCKET/prefix/",
        },
        RoleArn="arn:aws:iam::<account-number>:role/<role-name>",
        Tags=[
            {
                "Key": "string",
                "Value": "string"
            },
        ]
    )
```

Custom Integration

如果使用自定义集成，请排除以下参数：HumanLoopRequestSource、HumanLoopActivationConfig.

```
response = client.create_flow_definition(
    FlowDefinitionName="human-review-workflow-name",
    HumanLoopConfig={
        "WorkteamArn": workteamArn,
        "HumanTaskUiArn": humanTaskUiArn,
        "TaskTitle": "Image content moderation",
        "TaskDescription": "Review the image and instructions. Complete the task",
        "TaskCount": 1,
        "TaskAvailabilityLifetimeInSeconds": 43200,
        "TaskTimeLimitInSeconds": 3600,
        "TaskKeywords": [
            "content moderation",
        ],
    },
    OutputConfig={
        "S3OutputPath": "s3://DOC-EXAMPLE-BUCKET/prefix/",
    },
    RoleArn="arn:aws:iam::<account-number>:role/<role-name>",
    Tags=[
        {
            "Key": "string",
            "Value": "string"
        },
    ]
)
```

在创建人工审核工作流程后，您可以从响应中检索流定义 ARN：

```
    humanReviewWorkflowArn = response["FlowDefinitionArn"]
```

创建人工循环

您用来启动人工循环的 API 操作取决于您使用的 Amazon A2I 集成。

- 如果您使用 Amazon Textract 内置集成，则使用AnalyzeDocumentoperation.
- 如果您使用 Amazon Rekognition 内置集成，则使用DetectModerationLabelsoperation.
- 如果您使用自定义集成，可以使用StartHumanLoopoperation.

在下表中选择您的任务类型，以查看 Amazon Textract 和 Amazon Rekognition 的示例请求Amazon SDK for Python (Boto3).

Amazon Textract – Key-value pair extraction

以下示例使用Amazon SDK for Python (Boto3)调用`analyze_document`(在 us-west-2 中)。将斜体红色文本替换为您的资源。加入`DataAttributes`参数如果您使用的是 Amazon Mechanical Turk 员队伍。有关更多信息，请参阅 。分析 _文档(在文档中)Amazon SDK for Python (Boto)API 参考.

```
   response = client.analyze_document(
         Document={"S3Object": {"Bucket": "AWSDOC-EXAMPLE-BUCKET", "Name": "document-
name.pdf"},
         HumanLoopConfig={
            "FlowDefinitionArn":"arn:aws:sagemaker:us-west-2:111122223333:flow-
definition/flow-definition-name",
            "HumanLoopName":"human-loop-name",
            "DataAttributes" : {ContentClassifiers:
["FreeOfPersonallyIdentifiableInformation"|"FreeOfAdultContent"]}
         }
         FeatureTypes=["FORMS"]
    )
```

只有当 Amazon Textract 对文档分析任务的信心满足您在人工审核工作流程中指定的激活条件时，才会创建人类循环。你可以查看`response`元素来确定是否创建了人类循环。要查看此响应中包含的所有内容，请参阅`HumanLoopActivationOutput`.

```
    if "HumanLoopArn" in analyzeDocumentResponse["HumanLoopActivationOutput"]:
        # A human loop has been started!
        print(f"A human loop has been started with ARN:
 {analyzeDocumentResponse["HumanLoopActivationOutput"]["HumanLoopArn"]}"
```

Amazon Rekognition – Image moderation

以下示例使用Amazon SDK for Python (Boto3)调用`detect_moderation_labels`(在 us-west-2 中)。将斜体红色文本替换为您的资源。加入`DataAttributes`参数如果您使用的是 Amazon Mechanical Turk 员队伍。有关更多信息，请参阅 。detect_moderation_label(在文档中)Amazon SDK for Python (Boto)API 参考.

```
   response = client.detect_moderation_labels(
            Image={"S3Object":{"Bucket": "AWSDOC-EXAMPLE-BUCKET", "Name": "image-
name.png"}},
            HumanLoopConfig={
               "FlowDefinitionArn":"arn:aws:sagemaker:us-west-2:111122223333:flow-
definition/flow-definition-name",
```

```
                "HumanLoopName":"human-loop-name",
                "DataAttributes":{ContentClassifiers:
["FreeOfPersonallyIdentifiableInformation"|"FreeOfAdultContent"]}
            }
    )
```

只有当 Amazon Rekognition 对图像审核任务的信心满足您在人工审核工作流程中指定的激活条件时，才会创建人类循环。你可以查看`response`元素来确定是否创建了人类循环。要查看此响应中包含的所有内容，请参阅`HumanLoopActivationOutput`.

```
    if "HumanLoopArn" in response["HumanLoopActivationOutput"]:
        # A human loop has been started!
        print(f"A human loop has been started with ARN:
 {response["HumanLoopActivationOutput"]["HumanLoopArn"]}")
```

Custom Integration

以下示例使用Amazon SDK for Python (Boto3)调用`start_human_loop`(在 us-west-2 中)。将斜体红色文本替换为您的资源。加入`DataAttributes`参数如果您使用的是 Amazon Mechanical Turk 员队伍。有关更多信息，请参阅 。start_human_LOOP(在文档中)Amazon SDK for Python (Boto)API 参考.

```
   response = client.start_human_loop(
        HumanLoopName= "human-loop-name",
        FlowDefinitionArn= "arn:aws:sagemaker:us-west-2:111122223333:flow-definition/
flow-definition-name",
        HumanLoopInput={"InputContent": inputContentJson},
        DataAttributes={"ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation"|"FreeOfAdultContent"]}
   )
```

此示例将输入内容存储在变量中*inputContentJson*. 假设输入内容包含两个元素：文本模板和情绪（例如`Positive`、`Negative`，或者`Neutral`），其格式如下所示：

```
    inputContent = {
        "initialValue": sentiment,
         "taskObject": blurb
     }
```

钥匙`initialValue`和`taskObject`必须与工作人员任务模板的液体元素中使用的密钥相对应。请参阅中的自定义模板创建人工任务 UI (p. 2609)请参阅一个示例。

创建*inputContentJson*中，执行以下操作：

```
    import json

    inputContentJson = json.dumps(inputContent)
```

每次打电话时都会开始人类循环`start_human_loop`. 要检查人类循环的状态，请使用描述了_human_LOOP：

```
    human_loop_info = a2i.describe_human_loop(HumanLoopName="human_loop_name")
    print(f"HumanLoop Status: {resp["HumanLoopStatus"]}")
```

```
print(f"HumanLoop Output Destination: {resp["HumanLoopOutput"]}")
```

使用 Amazon A2I 使用案例和示例

您可以使用亚马逊 Augmented AI 将人工审核集成到您的工作流程中内置任务类型、Amazon Textract 和 Amazon Rekognition，或者使用自己的自定义任务自定义任务类型.

当您使用内置任务类型之一创建人工审核工作流时，您可以指定条件（如置信度阈值）来启动人工审核。满足这些条件时，该服务（Amazon Rekognition 或 Amazon Textract）代表您创建人工循环，并将您的输入数据直接提供给 Amazon A2I 以发送到人工审核者。要了解内置任务类型的更多信息，请使用以下命令：

- 在 Amazon Textract 中使用亚马逊 Augmented AI (p. 2617)
- 通过 Amazon Rekognition 使用亚马逊 Augmented AI (p. 2619)

当您使用自定义任务类型时，您可以使用 Amazon A2I 运行时 API 创建并启动人工循环。使用自定义任务类型，将人工审核工作流与其他Amazon服务或您自己的自定义 ML 应用程序。

- 有关更多详细信息，请参阅 在自定义任务类型中使用 Amazon Augmented AI (p. 2621)

下表概述了您可以使用 SageMaker Jupyter 笔记本探索的各种 Amazon A2I 使用案例。要开始使用 Jupyter 笔记本，请使用中的说明将 SageMaker 笔记本实例与 Amazon A2I Jupyter 笔记本电脑一起使用 (p. 2617). 有关更多示例，请参阅该 GitHub 存储库。

使用案例	说明	任务类型
使用亚马逊 A2I 和 Amazon Textract	让人类查看单页文档以查看重要的表单键值对，或者让 Amazon Textract 随机抽样并将数据集中的文档发送给人类进行审查。	内置
使用亚马逊 A2I 和 Amazon Rekognition	如果 Amazon Rekognition 返回的置信度较低，或者让 Amazon Rekognition 随机抽样并将数据集中的图像发送给人类进行审查，则让人类审核不安全图片是否存在明确的成人或暴力内容。	内置
使用 Amazon A2I 和 Amazon A2I 理解	让人们查看 Amazon Comprehend 关于文本数据的推论，例如情绪分析、文本语法和实体检测。	自定义
使用亚马逊 A2I 和 Amazon Transcribe	让人们查看 Amazon Transcribe 视频或音频文件的转录。使用转录人工评论循环的结果来创建自定义词汇并改善未来类似视频或音频内容的转录。	自定义
使用亚马逊 A2I 与 Amazon Translate	让人们审查从 Amazon Translate 返回的低信心翻译。	自定义
使用亚马逊 A2I 查看实时机器学习推理	使用 Amazon A2I 来查看部署到 SageMaker 托管终端节点的模型所做的实时低信心推断，并使用	自定义

使用案例	说明	任务类型
	Amazon A2I 输出数据以增量方式训练模型。	
使用 Amazon A2I 审查表格数据	使用 Amazon A2I 将人工评论循环集成到使用表格数据的机器学习应用程序中。	自定义

主题

- 将 SageMaker 笔记本实例与 Amazon A2I Jupyter 笔记本电脑一起使用 (p. 2617)
- 在 Amazon Textract 中使用亚马逊 Augmented AI (p. 2617)
- 通过 Amazon Rekognition 使用亚马逊 Augmented AI (p. 2619)
- 在自定义任务类型中使用 Amazon Augmented AI (p. 2621)

将 SageMaker 笔记本实例与 Amazon A2I Jupyter 笔记本电脑一起使用

对于演示如何将 Amazon A2I 人工审阅循环集成到机器学习工作流程中的端到端示例，您可以从中使用 Jupyter 笔记本GitHub 存储库在 SageMaker 笔记本实例中。

要在 Amazon SageMaker 笔记本实例中使用 Amazon A2I 自定义任务类型示例笔记本：

1. 如果您还没有活动的 SageMaker 笔记本实例，请按照中的说明创建一个第 1 步：创建 Amazon SageMaker 笔记本实例 (p. 59).
2. 当笔记本实例处于活动状态时，选择打开 JupyterLab位于笔记本实例名称的右侧。加载 JupyterLab 可能需要一些时间。
3. 选择 图标以将 GitHub 存储库克隆到您的工作区。
4. 输入 amazon-a2i-sample-jupyter-notebooks 存储库 HTTPS URL。
5. 选择 CLONE (克隆)。
6. 打开要运行的笔记本。
7. 按照笔记本中的说明配置人工审核工作流和人工循环并运行单元格。
8. 为了避免产生不必要的费用，当您完成演示时，请停止并删除笔记本实例以及在演练过程中创建的任何 Amazon S3 存储桶、IAM 角色和 CloudWatch Events 资源。

在 Amazon Textract 中使用亚马逊 Augmented AI

借助 Amazon Textract，您可以向应用程序添加文档文本检测和分析功能。Amazon Augmented AI (Amazon A2I) 直接与 Amazon Textract 集成`AnalyzeDocument`API 操作。可以使用 `AnalyzeDocument` 分析文档以了解检测到的项目之间的关系。当您将 Amazon A2I 人工评论循环添加到`AnalyzeDocument`请求，Amazon A2I 会监控 Amazon Textract 结果，并在满足流程定义中指定的条件时将文档发送给一个或多个工作人员以进行审核。例如，如果你希望人类查看一个特定的密钥，例如`Full name:`以及它们相关的输入值，你可以创建一个激活条件，随时启动人工评论`Full name:`检测到 key，或者当该键的推断置信度在您指定的范围内时。

下图描述了亚马逊 A2I 内置的 Amazon Textract 工作流程。左侧描述了创建 Amazon Textract 人工审核工作流程所需的资源：以及 Amazon S3 存储桶、激活条件、工作人员任务模板和工作团队。这些资源用于创建人工审核工作流，或流定义。一个箭头指向工作流程的下一步：使用 Amazon Textract 通过人工审核工作流

程配置人工循环。第二个箭头指向从此步骤直接指向满足人工审核工作流程中指定的激活条件的步骤。这启动了人工循环的创建。在图像右侧，人类循环分三个步骤描述：1) 生成工作人员 UI 和工具，任务可供员工使用 2) 工作人员查看输入数据，最后 3) 结果保存在 Amazon S3 中。

您可以通过指定指定在创建人工审核工作流程或流程定义时指定 Amazon Textract 将任务发送给工作人员以进行审核的时间。激活条件.

在使用 Amazon Textract 任务类型时，您可以设置以下激活条件：

- 根据表单键置信度分数启动对特定表单键的人工审查。
- 在特定表单键丢失时启动人工审查。
- 对 Amazon Textract 识别的所有表单键启动人工审查，并在指定范围内的置信度分数。
- 随机将表单示例发送给工作人员以进行审查。

当激活条件依赖于表单键置信度分数时，您可以使用两种类型的预测置信度来启动人工循环：

- 识别置信度— 在表单中检测到的键值对的置信度分数。
- 资格认证信— 表单中键和值中包含的文本的置信度分数。

在下一节的图片中，全名：Jane Doe是键值对，全名是关键，而且Jane Doe是值。

您可以在创建人工审核工作流程时使用 Amazon SageMaker 控制台设置这些激活条件，也可以通过为人工循环激活条件创建 JSON 并将其指定为`HumanLoopActivationConditions`的参数`CreateFlowDefinition`API 操作。要了解如何以 JSON 格式指定激活条件，请参阅 亚马逊中用于人 Augmented AI 循环激活条件的 JSON 架构 (p. 2628) 和将人工循环激活条件 JSON 架构与 Amazon Textract 一起使用 (p. 2629)。

> Note
>
> 将 Augmented AI 与 Amazon Textract 结合使用时，请在相同的内容创建 Augmented AI 资源 Amazon你用来打电话的地区`AnalyzeDocument`.

开始使用：将人工审核集成到 Amazon Textract 分析文档 Job 中

要将人工审核集成到 Amazon Textract 文本检测和分析作业中，您需要创建流程定义，然后使用 Amazon Textract API 将该流程定义集成到您的工作流程中。要了解如何使用 SageMaker 控制台或 Augmented AI API 创建流程定义，请参阅以下主题：

- 创建人工审核工作流程（控制台）(p. 2623)
- 创建人工审核工作流程 (API) (p. 2624)

创建流程定义后，请参阅将 Augmented AI 与 Amazon Textract 结合使用以了解如何将流程定义集成到 Amazon Textract 任务中。

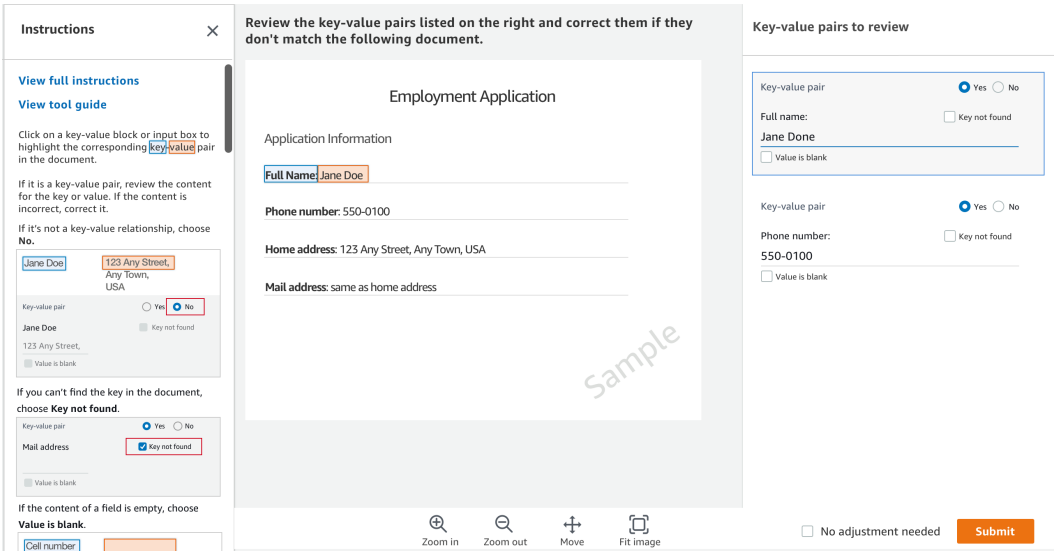
使用 Amazon Textract 和 Amazon A2I 的端到端示例

要了解如何使用控制台将 Amazon Textract 与 Amazon A2I 结合使用的端到端示例，请参阅教程：从 Amazon A2I 控制台开始使用 (p. 2603).

要了解如何使用 Amazon A2I API 创建并启动人工审核，您可以使用Amazon AAugmented AI (Amazon A2I) 与 Amazon Textract 的分析文档集成 [示例]在 SageMaker 笔记本实例中。要了解其用法，请参阅 将 SageMaker 笔记本实例与 Amazon A2I Jupyter 笔记本电脑一起使用 (p. 2617)。

A2I Textract 工作人员控制台预览

在 Amazon Textract 工作流程中为工作人员分配审核任务时，工作人员可能会看到与以下内容类似的用户界面：



您可以在创建人工审核定义时，或通过创建并使用自定义模板，来在 SageMaker 控制台中自定义此界面。要了解更多信息，请参阅“创建和管理工作人员任务模板 (p. 2647)”。

通过 Amazon Rekognition 使用亚马逊 Augmented AI

Amazon Rekognition 让您可以向应用程序轻松添加图像分析功能。Amazon Rekognition`DetectModerationLabels`API 操作直接与 Amazon A2I 集成，因此您可以轻松创建人工循环来审核不安全的图像，例如明显的成人或暴力内容。您可以使用 `DetectModerationLabels`，通过流定义 ARN 配置人工循环。这使 Amazon A2I 能够分析由 Amazon Rekognition 做出的预测，并将结果发送给人员以便审查，以确保它们符合流程定义中设置的条件。

下图描述了 Amazon Rekognition 的亚马逊 A2I 内置工作流程。左侧描述了创建 Amazon Rekognition 人工审核工作流程所需的资源：以及 Amazon S3 存储桶、激活条件、工作人员任务模板和工作团队。这些资源用于创建人工审核工作流，或流定义流程。一个箭头指向工作流程的下一步：使用 Amazon Rekognition 通过人工审核工作流程配置人工循环。第二个箭头指向从此步骤直接指向满足人工审核工作流程中指定的激活条件的步骤。这启动了人工循环的创建。在图像右侧，人类循环分三个步骤描述：1) 生成工作人员 UI 和工具，任务可供工作人员使用，2) 工作人员查看输入数据，最后 3) 结果保存在 Amazon S3 中。

在使用 Amazon Rekognition 任务类型时，可以设置以下激活条件：

- 根据标签置信度分数，启动针对由 Amazon Rekognition 标识的标签的人工审查。
- 随机将图像示例发送给人员以进行审查。

您可以在创建人工审核工作流程时使用 Amazon SageMaker 控制台设置这些激活条件，也可以通过为人工循环激活条件创建 JSON 并将其指定为输入来设置这些激活条件。`HumanLoopActivationConditions`的参数`CreateFlowDefinition`API 操作。要了解如何以 JSON 格式指定激活条件，请参阅 亚马逊中用于人 Augmented AI 循环激活条件的 JSON 架构 (p. 2628) 和将人工循环激活条件 JSON 架构与 Amazon Rekognition 结合使用 (p. 2635)。

> Note
>
> 将 Augmented AI 与 Amazon Rekognition 一起使用时，请在同一个方面创建 Augmented AI 资源 Amazon你用来打电话的地区`DetectModerationLabels`.

开始使用：将人工审核集成到 Amazon Rekognition 图像审核 Job 中

要将人工审核集成到 Amazon Rekognition 中，请参阅以下主题：

- 创建人工审核工作流程（控制台） (p. 2623)
- 创建人工审核工作流程 (API) (p. 2624)

创建流程定义之后，请参阅在 Amazon Rekognition 中使用 Augmented AI以了解如何将流程定义集成到 Amazon Rekognition 任务中。

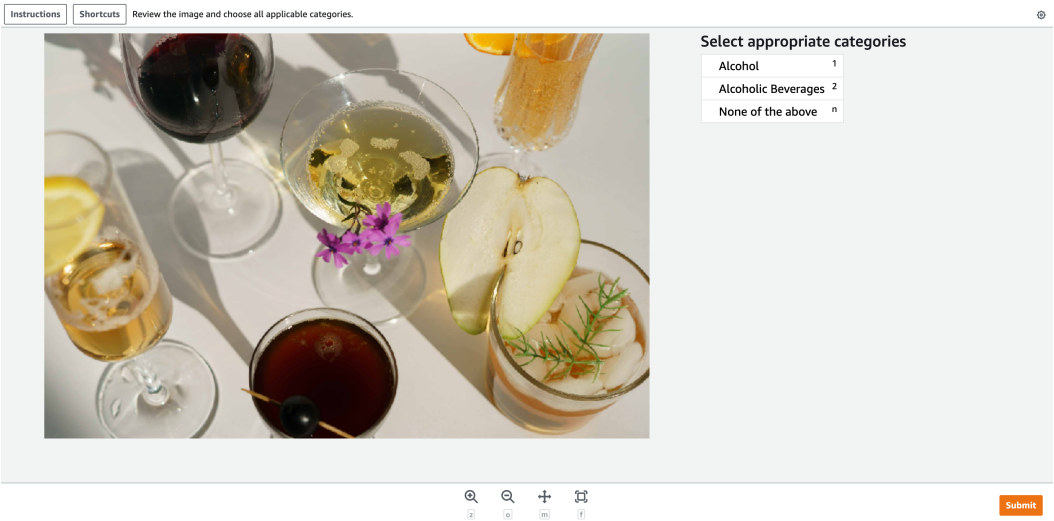
使用 Amazon Rekognition 和 Amazon A2I 使用 Amazon A2I 的端到端演示

有关演示如何使用控制台将 Amazon Rekognition 与 Amazon A2I 结合使用的端到端示例，请参阅教程：从 Amazon A2I 控制台开始使用 (p. 2603).

要了解如何使用 Amazon A2I API 创建并启动人工审核，可以使用Amazon Augmented AI (Amazon A2I) 与 Amazon Rekognition 集成 [示例]在 SageMaker 笔记本实例中。要了解其用法，请参阅 将 SageMaker 笔记本实例与 Amazon A2I Jupyter 笔记本电脑一起使用 (p. 2617)。

A2I Rekognition 工作人员控制台预览

在 Amazon Rekognition 工作流程中为工作人员分配审核任务时，工作人员可能会看到与以下内容类似的用户界面：



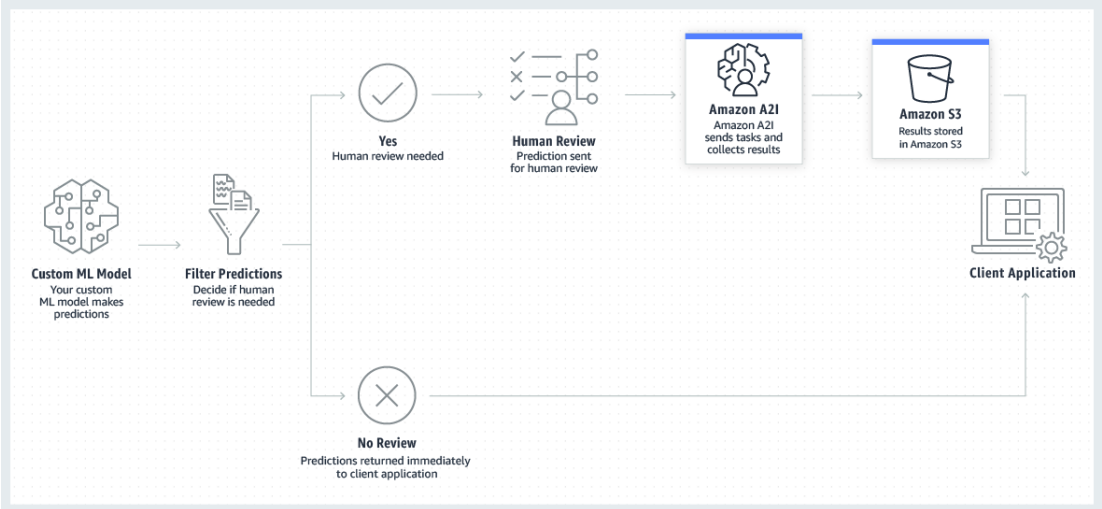
可以在创建人工审核定义时，或通过创建并使用自定义模板，来在 SageMaker 控制台中自定义此界面。要了解更多信息，请参阅“创建和管理工作人员任务模板 (p. 2647)”。

在自定义任务类型中使用 Amazon Augmented AI

您可以使用 Amazon Augmented AI (Amazon A2I) 将人工审核（人工循环）集成到任何机器学习工作流程使用自定义任务类型. 此选项为您提供了最大的灵活性，可以自定义将数据对象发送给人类进行审阅的条件，以及工作人员用户界面的外观。

使用自定义任务类型时，您可以创建自定义人工审核工作流程，并指定直接在应用程序中发送数据对象以供人工审阅的条件。

下图描述了 Amazon A2I 自定义工作流程。自定义机器学习模型用于生成预测。客户端应用程序使用用户定义的标准过滤这些预测，并确定是否需要人工审查。如果是，这些预测将发送给亚马逊 A2I 进行人工审查。Amazon A2I 在 Amazon S3 中收集人工审核结果，客户端应用程序可以访问该结果。如果筛选器确定不需要人工评论，则可以直接向客户应用程序提供预测。



使用本页上的步骤了解如何使用自定义任务类型将 Amazon A2I 集成到任何机器学习工作流程中。

要使用流定义创建人工循环，请将其集成到应用程序中并监控结果。

1. 完成 Amazon A2I使用 Augmented AI 的先决条件 (p. 2603). 请注意以下几点：
   - Amazon Simple Storage Service (Amazon S3) 存储桶的路径。
   - 的 Amazon 资源名称 (ARN)Amazon Identity and Access Management附加了所需权限的 (IAM) 角色。
   - （可选）私有人力的 ARN（如果您计划使用）。
2. 通过使用 HTML 元素，创建自定义工作人员模板，Amazon A2I 将使用该模板生成工作人员任务 UI。要了解如何创建自定义模板，请参阅 创建自定义 Worker 任务模板 (p. 2648)。
3. 使用步骤 2 中的自定义工作人员模板在 Amazon SageMaker 控制台中生成工作人员任务模板。要了解如何操作，请参阅创建工作人员任务模板 (p. 2647)。

   在下一步，您将创建流定义：
   - 如果要使用 SageMaker API 创建流定义，请记下此工作人员任务模板的 ARN 以在下一步中使用。
   - 如果要使用控制台创建流定义，则模板将自动显示在工作者任务模板选择时的部分创建人工审核工作流.
4. 创建流定义时，请提供 S3 存储桶的路径、IAM 角色 ARN 和工作人员模板。
   - 要了解如何使用 SageMaker 创建流定义`CreateFlowDefinition`API，请参阅创建人工审核工作流程 (API) (p. 2624).
   - 要了解如何使用 SageMaker 控制台创建流定义，请参阅创建人工审核工作流程（控制台） (p. 2623).
5. 使用Amazon A2I 运行时 API. 要了解如何操作，请参阅创建和启动人工循环 (p. 2640)。
6. 要控制何时在应用程序中启动人工审核，请指定在应用程序中调用 `StartHumanLoop` 的条件。在将 Amazon A2I 与自定义任务类型结合使用时，启动人工循环的置信度阈值等人工循环激活条件将不可用。每次 `StartHumanLoop` 调用都会导致人工审核。

启动人工循环后，您可以使用 Amazon Augmented AI 运行时 API 和 Amazon EventBridge（也称为 Amazon CloudWatch Events）管理和监控循环。要了解更多信息，请参阅 监控和管理您的人工循环 (p. 2653)。

使用 Amazon A2I 自定义任务类型的端到端教程

有关演示如何将 Amazon A2I 集成到各种 ML 工作流程中的端到端示例，请参阅中的表格使用 Amazon A2I 使用案例和示例 (p. 2616). 要开始使用其中一台笔记本电脑，请参阅将 SageMaker 笔记本实例与 Amazon A2I Jupyter 笔记本电脑一起使用 (p. 2617).

创建人工审核工作流程

使用 Amazon Augmented AI (Amazon A2I)人工审核工作流，或者流定义对于指定以下内容：

- 对于 Amazon Textract 和 Amazon Rekognition 内置任务类型，指定调用您人工循环的条件。
- 将任务发送到的员工
- 您的工作人员收到的一组说明，称为作人员任务模板
- 工作人员任务的配置，包括接收任务的工作人员数量和完成任务的时间限制。
- 存储输出数据的位置

您可以在 SageMaker 控制台中或使用 SageMaker 创建人工审核工作流程。`CreateFlowDefinition`operation. 创建流定义时，您可以使用控制台构建工作人员任务模板以及 Amazon Rekognition 任务类型。

Important

启动人类循环的人类循环激活条件（例如，置信阈值）不适用于 Amazon A2I 自定义任务类型。使用控制台为自定义任务类型创建流定义时，您无法指定激活条件。使用 Amazon A2I API 为自定义任务类型创建流定义时，您无法将`HumanLoopActivationConditions`的属性`HumanLoopActivationConditionsConfig`参数。要控制何时启动人工审核，请指定在自定义应用程序中调用 `StartHumanLoop` 的条件。在这种情况下，每次 `StartHumanLoop` 调用都会导致人工审核。有关更多信息，请参阅 在自定义任务类型中使用 Amazon Augmented AI (p. 2621)。

先决条件

要创建人工审核工作流定义，您必须已完成中所述的先决条件。使用 Augmented AI 的先决条件 (p. 2603).

如果您使用 API 为任何任务类型创建流定义，或者在控制台中创建流定义时使用自定义任务类型，请首先创建工作人员任务模板。有关更多信息，请参阅 创建和管理工作人员任务模板 (p. 2647)。

如果您在控制台中为内置任务类型创建流定义时，希望预览工作人员任务模板，请确保使用类似于所述的策略，向您用于创建流定义的角色授予访问 Amazon S3 存储桶的权限（其中包含您的模板构件）。在启用工作人员任务模板预览 (p. 2666).

主题

- 创建人工审核工作流程（控制台） (p. 2623)
- 创建人工审核工作流程 (API) (p. 2624)
- 亚马逊中用于人 Augmented AI 循环激活条件的 JSON 架构 (p. 2628)

创建人工审核工作流程（控制台）

使用此过程可以使用 SageMaker 控制台创建 Amazon Augmented AI (Amazon A2I) 人工审核工作流程。如果您刚开始使用 Amazon A2I，我们建议您使用组织中的人员创建一个私有工作团队，并在创建流定义时使用该工作团队的 ARN。要了解如何设置私有人力和创建工作团队，请参阅创建私有人力（Amazon SageMaker 控制台） (p. 468)。如果您已设置私有人力，请参阅使用 SageMaker 控制台创建工作团队 (p. 471)以了解如何将工作团队添加到该人力中。

如果您将 Amazon A2I 与某个内置任务类型结合使用，则在控制台中创建人工审查工作流程时，可以使用 Augmented AI 提供的默认工作人员任务模板来创建工作人员说明。要查看 Augmented AI 提供的默认模板示例，请参阅中的内置任务类型。使用 Amazon A2I 使用案例和示例 (p. 2616).

创建流定义（控制台）

1. 从打开 SageMaker 控制台https://console.amazonaws.cn/sagemaker/.
2. 在导航窗格的 Augmented AI 部分中，选择人工审核工作流，然后选择创建人工审核工作流。
3. 在 Overview (概述) 中，执行以下操作：
   a. 在名称中，输入唯一工作流名称。该名称必须使用小写字母，在您账户所在的 Amazon 区域内唯一，并且最多可以包含 63 个字符。有效字符包括：a-z、0-9 和 -（连字符）。
   b. 在输出的 S3 位置中，输入将人工审核结果存储到的 S3 存储桶。该存储桶必须与工作流位于同一 Amazon 区域中。
   c. 适用于IAM 角色中，选择具有所需权限的角色。如果您选择内置任务类型并希望在控制台中预览工作人员模板，请提供附加了启用工作人员任务模板预览 (p. 2666)中所描述策略类型的角色。
4. 在任务类型中，选择您希望人工执行的任务类型。
5. 如果您选择 Amazon Rekognition 或 Amazon Textract 任务类型，请指定调用人工审核的条件。

- 对于 Amazon Rekognition 图像审核任务，选择一个推理置信度评分阈值区间，以启动人工审核。
- 对于 Amazon Textract 任务，当特定表单密钥缺失或表单密钥检测置信度较低时，您可以启动人工审核。如果在对文本中的所有表单密钥进行评估后，置信度低于任何表单密钥所需的阈值，您也可以启动人工审核。两个变量指定了信心阈值：识别信心和资格认证信. 要了解有关这些变量的更多信息，请参阅在 Amazon Textract 中使用亚马逊 Augmented AI (p. 2617)。
- 对于这两种任务类型，您可以随机将一定比例的数据对象（图像或表单）及其标签发送给工作人员以供审查。

6. 配置并指定工作人员任务模板：

   a. 如果您使用的是 Amazon Rekognition 或 Amazon Textract 任务类型：

      - 在创建模板部分：

        - 要使用 Amazon Rekognition 和 Amazon Textract 任务类型的 Amazon A2I 默认模板为您的工作人员创建说明，请选择从默认模板构建.
          - 如果您选择从默认模板构建，请在工作人员任务设计中创建您的说明。
            - 提供模板名称这是独一无二的Amazon你所在的地区。
            - 在说明部分中，提供有关如何完成您的任务的详细说明。为了帮助工作人员达到更好的准确性，请提供好示例和坏示例。
            - （可选）在其他说明中，向您的工作人员提供其他信息和说明。

              有关创建有效说明的信息，请参阅创建良好的工作人员说明 (p. 2652)。
        - 要选择您已创建的自定义模板，请从模板菜单中选择该模板，并提供一个任务描述来简要描述工作人员的任务。要了解如何创建自定义模板，请参阅 创建工作人员任务模板 (p. 2647)。

   b. 使用自定义任务类型时：

      - 在工作者任务模板在部分中，从列表中选择您的模板。您在 SageMaker 控制台中创建的所有模板都将显示在此列表中。要了解如何为自定义任务类型创建模板，请参阅创建和管理工作人员任务模板 (p. 2647)。

7. （可选）预览工作人员模板：

   对于 Amazon Rekognition 和 Amazon Textract 任务类型，您可以选择选择查看示例工作人员任务以预览你的工作人员任务 UI。

   如果要为自定义任务类型创建流定义，您可以使用 `RenderUiTemplate` 操作预览工作人员任务 UI。有关更多信息，请参阅 预览工作人员任务模板 (p. 2652)。

8. 在工作人员中，选择人力类型。
9. 选择创建。

后续步骤

创建人工审核工作流后，该工作流将显示在控制台中的人工审核工作流下。要查看流定义的 Amazon 资源名称 (ARN) 和配置详细信息，请通过选择名称来选择工作流。

如果您使用的是内置任务类型，则可以使用流定义 ARN，通过该类型启动人工循环。Amazon服务的 API（例如，Amazon Textract API）。对于自定义任务类型，您可以使用 ARN，通过 Amazon Augmented AI 运行时 API 启动人工循环。要了解有关这两个选项的更多信息，请参阅创建和启动人工循环 (p. 2640)。

创建人工审核工作流程 (API)

要使用 SageMaker API 创建流定义，请使用`CreateFlowDefinition`operation. 完成后，执行使用 Augmented AI 的先决条件 (p. 2603)，请按照以下步骤了解如何使用此 API 操作。

有关的概述CreateFlowDefinition操作和有关每个参数的详细信息，请参阅CreateFlowDefinition.

创建流定义 (API)

1. 对于 FlowDefinitionName，输入唯一名称。该名称在您账户所在的 Amazon 区域中必须唯一，并且最多可以包含 63 个字符。有效字符包括：a-z、0-9 和 - (连字符) 。
2. 对于 RoleArn，输入您为授予对数据源访问权限而配置的角色的 ARN。
3. 对于 HumanLoopConfig，输入有关工作人员及其所见内容的信息。有关 HumanLoopConfig 中每个参数的信息，请参阅 HumanLoopConfig。
4. （可选）如果您使用内置任务类型，请在中提供启动人工循环的条件。HumanLoopActivationConfig. 要了解如何创建 HumanLoopActivationConfig 参数所需的输入，请参阅亚马逊中用于人 Augmented AI 循环激活条件的 JSON 架构 (p. 2628)。如果您没有在此指定条件，则当您向Amazon与内置任务类型（例如 Amazon Textract 或 Amazon Rekognition）关联的服务（例如 Amazon Textract 或 Amazon Rekognition），该服务将每个任务发送给工作人员以进行审查。

   如果您使用自定义任务类型，则将禁用 HumanLoopActivationConfig。要了解如何使用自定义任务类型控制何时将任务发送给工作人员，请参阅在自定义任务类型中使用 Amazon Augmented AI (p. 2621)。
5. （可选）如果您使用的是内置任务类型，Amazon Rekognition 在HumanLoopRequestSource参数。
6. 适用于OutputConfig中，指示在 Amazon Simple Storage Service (Amazon S3) 中存储人工循环输出的位置。
7. （可选）使用Tags以输入键值对以帮助您分类和组织流定义。每个标签都由键 和值组成，这两个参数都由您定义。

Amazon Textract – Key-value pair extraction

以下是使用Amazon SDK for Python (Boto3). 您必须使用'AWS/Textract/AnalyzeDocument/Forms/V1'创建 Amazon Textract 人工循环。只包括PublicWorkforceTaskPrice如果你使用的是 Mechanical Turk 员队伍。

```
sagemaker_client = boto3.client('sagemaker', aws_region)

response = sagemaker_client.create_flow_definition(
    FlowDefinitionName='ExampleFlowDefinition',
    HumanLoopRequestSource={
         'AwsManagedHumanLoopRequestSource': 'AWS/Textract/AnalyzeDocument/Forms/V1'
    },
    HumanLoopActivationConfig={
        'HumanLoopActivationConditionsConfig': {
            'HumanLoopActivationConditions': '{...}'
        }
    },
    HumanLoopConfig={
        'WorkteamArn': 'arn:aws:sagemaker:aws_region:aws_account_number:workteam/
private-crowd/workteam_name',
        'HumanTaskUiArn': 'arn:aws:sagemaker:aws_region:aws_account_number:human-task-
ui/template_name',
        'TaskTitle': 'Example task title',
        'TaskDescription': 'Example task description.',
        'TaskCount': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskKeywords': [
            'Keyword1','Keyword2'
        ],
        'PublicWorkforceTaskPrice': {
            'AmountInUsd': {
                'Dollars': 123,
                'Cents': 123,
```

```
                'TenthFractionsOfACent': 123
            }
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/',
        'KmsKeyId': '1234abcd-12ab-34cd-56ef-1234567890ab'
    },
    RoleArn='arn:aws:iam::aws_account_number:role/role_name',
    Tags=[
        {
            'Key': 'KeyName',
            'Value': 'ValueName'
        },
    ]
)
```

Amazon Rekognition – Image moderation

以下是使用Amazon SDK for Python (Boto3). 您必须使用`'AWS/Rekognition/DetectModerationLabels/Image/V3'`以创建 Amazon Rekognition 流定义。只包括`PublicWorkforceTaskPrice`如果你使用的是 Mechanical Turk 员队伍。

```
sagemaker_client = boto3.client('sagemaker', aws_region)

response = sagemaker_client.create_flow_definition(
    FlowDefinitionName='ExampleFlowDefinition',
    HumanLoopRequestSource={
         'AwsManagedHumanLoopRequestSource': 'AWS/Rekognition/DetectModerationLabels/
Image/V3'
    },
    HumanLoopActivationConfig={
        'HumanLoopActivationConditionsConfig': {
            'HumanLoopActivationConditions': '{...}'
        }
    },
    HumanLoopConfig={
        'WorkteamArn': 'arn:aws:sagemaker:aws_region:aws_account_number:workteam/
private-crowd/workteam_name',
        'HumanTaskUiArn': 'arn:aws:sagemaker:aws_region:aws_account_number:human-task-
ui/template_name',
        'TaskTitle': 'Example task title',
        'TaskDescription': 'Example task description.',
        'TaskCount': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskKeywords': [
            'Keyword1','Keyword2'
        ],
        'PublicWorkforceTaskPrice': {
            'AmountInUsd': {
                'Dollars': 123,
                'Cents': 123,
                'TenthFractionsOfACent': 123
            }
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/',
        'KmsKeyId': '1234abcd-12ab-34cd-56ef-1234567890ab'
    },
    RoleArn='arn:aws:iam::aws_account_number:role/role_name',
    Tags=[
        {
```

```
            'Key': 'KeyName',
            'Value': 'ValueName'
        },
    ]
)
```

Custom Workflow

以下是为自定义集成创建人工审核工作流程（流定义）的请求示例。要创建这种类型的人工审核工作流程，请省略HumanLoopRequestSource来自流程定义请求。您只需要包含PublicWorkforceTaskPrice如果你使用的是 Mechanical Turk 员队伍。

```
sagemaker_client = boto3.client('sagemaker', aws_region)

response = sagemaker_client.create_flow_definition(
    FlowDefinitionName='ExampleFlowDefinition',
    HumanLoopActivationConfig={
        'HumanLoopActivationConditionsConfig': {
            'HumanLoopActivationConditions': '{...}'
        }
    },
    HumanLoopConfig={
        'WorkteamArn': 'arn:aws:sagemaker:aws_region:aws_account_number:workteam/
private-crowd/workteam_name',
        'HumanTaskUiArn': 'arn:aws:sagemaker:aws_region:aws_acount_number:human-task-
ui/template_name',
        'TaskTitle': 'Example task title',
        'TaskDescription': 'Example task description.',
        'TaskCount': 123,
        'TaskAvailabilityLifetimeInSeconds': 123,
        'TaskTimeLimitInSeconds': 123,
        'TaskKeywords': [
            'Keyword1','Keyword2'
        ],
        'PublicWorkforceTaskPrice': {
            'AmountInUsd': {
                'Dollars': 123,
                'Cents': 123,
                'TenthFractionsOfACent': 123
            }
        }
    },
    OutputConfig={
        'S3OutputPath': 's3://bucket/path/',
        'KmsKeyId': '1234abcd-12ab-34cd-56ef-1234567890ab'
    },
    RoleArn='arn:aws:iam::account_number:role/role_name',
    Tags=[
        {
            'Key': 'KeyName',
            'Value': 'ValueName'
        },
    ]
)
```

后续步骤

成功调用 CreateFlowDefinition API 操作的返回值是流定义的 Amazon 资源名称 (ARN)。

如果您使用的是内置任务类型，则可以使用流定义 ARN，通过该类型启动人工循环。Amazon服务的 API（即 Amazon Textract API）。对于自定义任务类型，您可以使用 ARN，通过 Amazon Augmented AI 运行时 API 启动人工循环。要了解有关这些选项的更多信息，请参阅创建和启动人工循环 (p. 2640)。

亚马逊中用于人 Augmented AI 循环激活条件的 JSON 架构

`HumanLoopActivationConditions` 是 `CreateFlowDefinition` API 的输入参数。此参数是 JSON 格式的字符串。当根据集成 AI 服务 API（例如）的响应评估这些条件时，JSON 对人工循环的创建条件进行建模。`Rekognition.DetectModerationLabels`要么`Textract.AnalyzeDocument`)。此响应称为推理。例如，Amazon Rekognition 发送审核标签的推理，并带有关联的置信度得分。在此示例中，推理是模型对适合图像的标签的最佳估计值。对于 Amazon Textract，推断文本块之间的关联（键/值对），例如之间的关联`Name:`和`Sue`在表单中以及文本块中的内容，或者单词块，例如“姓名”。

下面是 JSON 的架构。在顶层，`HumanLoopActivationConditions` 具有 JSON 数组 `Conditions`。此数组的每个成员都是一个独立的条件，如果评估为`true`，导致亚马逊 A2I 创造了人类循环。每个这样的独立条件可以是简单条件或复杂条件。简单条件具有以下属性：

- `ConditionType`：此属性标识条件的类型。EACHAmazon与 Amazon A2I 集成的 AI 服务 API 定义了自己的一套允许的`ConditionTypes`.
  - Rekognition`DetectModerationLabels`— 此 API 支持`ModerationLabelConfidenceCheck`和`Sampling` `ConditionType`有效值。
  - Textract`AnalyzeDocument`— 此 API 支持`ImportantFormKeyConfidenceCheck`、`MissingImportantFormKey`, 和`Sampling` `ConditionType`有效值。
- `ConditionParameters`— 这是一个 JSON 对象，用于将条件参数化。此对象的允许属性集取决于 `ConditionType` 的值。每个 `ConditionType` 定义它自己的 `ConditionParameters` 集。

`Conditions` 数组的成员可以对复杂条件进行建模。这是通过使用逻辑连接简单条件来实现的`And`和`Or`逻辑运算符和嵌套基本的简单条件。最多支持两层嵌套。

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "definitions": {
        "Condition": {
            "type": "object",
            "properties": {
                "ConditionType": {
                    "type": "string"
                },
                "ConditionParameters": {
                    "type": "object"
                }
            },
            "required": [
                "ConditionType"
            ]
        },
        "OrConditionArray": {
            "type": "object",
            "properties": {
                "Or": {
                    "type": "array",
                    "minItems": 2,
                    "items": {
                        "$ref": "#/definitions/ComplexCondition"
                    }
                }
            }
        },
        "AndConditionArray": {
            "type": "object",
```

```
            "properties": {
                "And": {
                    "type": "array",
                    "minItems": 2,
                    "items": {
                        "$ref": "#/definitions/ComplexCondition"
                    }
                }
            }
        },
        "ComplexCondition": {
            "anyOf": [
                {
                    "$ref": "#/definitions/Condition"
                },
                {
                    "$ref": "#/definitions/OrConditionArray"
                },
                {
                    "$ref": "#/definitions/AndConditionArray"
                }
            ]
        }
    },
    "type": "object",
    "properties": {
        "Conditions": {
            "type": "array",
            "items": {
                "$ref": "#/definitions/ComplexCondition"
            }
        }
    }
}
```

Note

人工循环激活条件不适用于与自定义任务类型集成的人工审核工作流。对于自定义任务类型，`HumanLoopActivationConditions` 参数禁用。

主题

- 将人工循环激活条件 JSON 架构与 Amazon Textract 一起使用 (p. 2629)
- 将人工循环激活条件 JSON 架构与 Amazon Rekognition 结合使用 (p. 2635)

将人工循环激活条件 JSON 架构与 Amazon Textract 一起使用

与 Amazon A2I 一起使用时，`AnalyzeDocument`操作支持以下输入：`ConditionType`参数：

- `ImportantFormKeyConfidenceCheck`— 使用此条件可在推理置信度在文档表单键和单词块的指定范围内时创建人工循环。表单键 是文档中与输入关联的任何单词。输入称为值。表单键和值一起称为键/值对。一个单词块指 Amazon Textract 在检测到的文本块中识别的单词。要了解有关 Amazon Textract 文档块的更多信息，请参阅文档和阻止对象中的Amazon Textract 开发人员指南.
- `MissingImportantFormKey`— 使用此条件可在 Amazon Textract 未在文档中标识键或其关联的别名时创建人工循环。
- `Sampling`— 使用此条件指定发送进行人工审核的表单百分比，而不考虑推理置信度得分。使用此条件来执行以下操作：
  - 通过对模型分析的所有表单随机抽样并发送指定百分比的表单进行人工审核，对 ML 模型进行审计。

- 使用 ImportantFormKeyConfidenceCheck 条件随机抽样满足 ImportantFormKeyConfidenceCheck 中指定的条件的一定百分比的推理以启动人工循环，并仅发送指定的百分比进行人工审核。

  Note

  如果你将同样的请求发送到AnalyzeDocument多次，结果Sampling不会因为该输入的推断而改变。例如，如果你做了AnalyzeDocument请求一次，然后Sampling不会启动人类循环，随后请求AnalyzeDocument使用相同的配置不会启动人类循环。

ImportantFormKeyConfidenceCheck输入和结果

ImportantFormKeyConfidenceCheck ConditionType 支持以下 ConditionParameters：

- ImportantFormKey— 一个字符串，表示 Amazon Textract 检测到的键/值对中的键，需要人工审核。如果此参数的值是包罗万象的特殊值 (*)，则所有键都被视为与条件匹配。您可以使用此项，对满足特定置信度阈值的任何键/值对都需要人工审核的情况进行建模。
- ImportantFormKeyAliases— 表示重要表单键的替代拼写或逻辑等效物的数组。
- KeyValueBlockConfidenceEquals
- KeyValueBlockConfidenceLessThan
- KeyValueBlockConfidenceLessThanEquals
- KeyValueBlockConfidenceGreaterThan
- KeyValueBlockConfidenceGreaterThanEquals
- WordBlockConfidenceEquals
- WordBlockConfidenceLessThan
- WordBlockConfidenceLessThanEquals
- WordBlockConfidenceGreaterThan
- WordBlockConfidenceGreaterThanEquals

当您使用ImportantFormKeyConfidenceCheck ConditionType中，Amazon A2I 发送键/值块和键/值块推理和键/值块推理以及您在中指定的关联别名。ImportantFormKey和ImportantFormKeyAliases供人类评论。

在创建流程定义时，如果您使用中提供的默认工作人员任务模板人工审核工作流Amazon SageMaker 控制台的部分、按此激活条件发送给人类审查的键值和区块推断都包含在工作人员 UI 中。如果您使用自定义工作人员任务模板，则需要包含{{ task.input.selectedAiServiceResponse.blocks }}元素，包含来自 Amazon Textract 的初始值输入数据（推理）。有关使用此输入元素的自定义模板的示例，请参阅Amazon Textract 的自定义模板示例 (p. 2649)。

MissingImportantFormKey输入和结果

MissingImportantFormKey ConditionType 支持以下 ConditionParameters：

- ImportantFormKey— 一个字符串，表示 Amazon Textract 检测到的键/值对中的键，需要人工审核。
- ImportantFormKeyAliases— 表示重要表单键的替代拼写或逻辑等效物的数组。

当您使用MissingImportantFormKey ConditionType，如果钥匙进入ImportantFormKey或者中的别名ImportantFormKeyAliases不包含在 Amazon Textract 推理中，将发送表单供人工审核，且不包含任何预测的键/值对。例如，如果 Amazon Textract 仅确定Address和Phone在表格中，但错过了ImportantFormKey Name（在MissingImportantFormKey条件类型）将发送表单供人工审核，而不检测到任何表单键（Address和Phone)。

如果您使用 SageMaker 控制台中提供的默认工作人员任务模板，则会创建一个任务，要求工作人员在中标识中的键。`ImportantFormKey`以及相关的价值。如果您使用自定义工作人员任务模板，则需要包含 `<task.input.humanLoopContext>` 自定义 HTML 元素以配置此任务。

采样输入和结果

`Sampling ConditionType` 现在支持 `RandomSamplingPercentage ConditionParameters`。`RandomSamplingPercentage` 的输入必须是 0.01 到 100 之间的实数。此数字表示符合人工审核条件并发送进行人工审核的数据百分比。如果您将`Sampling`在没有任何其他条件的情况下，此数字表示所得出的所有推理中，`AnalyzeDocument`操作来自发送给人员进行审查的单个请求。

如果您指定 `Sampling` 条件而没有任何其他条件类型，则所有键/值和块推理都会发送给工作人员审核。

在创建流程定义时，如果您使用中提供的默认工作人员任务模板人工审核工作流SageMaker 控制台的部分，通过此激活条件发送给人类审查的所有键值和区块推断都包含在工作人员 UI 中。如果您使用自定义工作人员任务模板，则需要包含`{{ task.input.selectedAiServiceResponse.blocks }}`元素，包含来自 Amazon Textract 的初始值输入数据（推理）。有关使用此输入元素的自定义模板的示例，请参阅Amazon Textract 的自定义模板示例 (p. 2649)。

示例

虽然只有一个条件需要评估`true`为了启动人工循环，亚马逊 A2I 评估 Amazon Textract 分析的每个对象的所有条件。人工审核者需要审查评估到的所有条件的重要表单键。`true`.

示例 1：在指定范围内使用置信度分数检测重要表单键，以启动人工循环。

以下示例显示了一个`HumanLoopActivationConditions`如果满足以下三个条件中的任何一个，则启动人工循环：

- Amazon Textract`AnalyzeDocument`API 返回键/值对的键/值对，其键为`Employee Name`、`Name`，或者`EmployeeName`，键/值块的置信度小于 60，组成键和值的每个单词块的置信度小于 85。
- Amazon Textract`AnalyzeDocument`API 返回键/值对的键/值对，其键为`Pay Date`、`PayDate`、`DateOfPay`，或者`pay-date`，键/值块的置信度小于 65，组成键和值的每个单词块的置信度小于 85。
- Amazon Textract`AnalyzeDocument`API 返回键/值对的键/值对，其键为`Gross Pay`、`GrossPay`，或者`GrossAmount`，键/值块的置信度小于 60，组成键和值的每个单词块的置信度小于 85。

```
{
    "Conditions": [
        {
            "ConditionType": "ImportantFormKeyConfidenceCheck",
            "ConditionParameters": {
                "ImportantFormKey": "Employee Name",
                "ImportantFormKeyAliases": [
                    "Name",
                    "EmployeeName"
                ],
                "KeyValueBlockConfidenceLessThan": 60,
                "WordBlockConfidenceLessThan": 85
            }
        },
        {
            "ConditionType": "ImportantFormKeyConfidenceCheck",
            "ConditionParameters": {
                "ImportantFormKey": "Pay Date",
                "ImportantFormKeyAliases": [
                    "PayDate",
                    "DateOfPay",
                    "pay-date"
                ],
```

```
                "KeyValueBlockConfidenceLessThan": 65,
                "WordBlockConfidenceLessThan": 85
            }
        },
        {
            "ConditionType": "ImportantFormKeyConfidenceCheck",
            "ConditionParameters": {
                "ImportantFormKey": "Gross Pay",
                "ImportantFormKeyAliases": [
                    "GrossPay",
                    "GrossAmount"
                ],
                "KeyValueBlockConfidenceLessThan": 60,
                "WordBlockConfidenceLessThan": 85
            }
        }
    ]
}
```

示例 2：使用 **`ImportantFormKeyConfidenceCheck`**

在以下示例中，如果 Amazon Textract 检测到一个键值对，其键/值块的置信度小于 60，并且任何基础单词块的置信度小于 90，则会创建人工循环。人工审查者被要求审查与置信值比较公式匹配的所有表单键值对。

```
{
    "Conditions": [
        {
            "ConditionType": "ImportantFormKeyConfidenceCheck",
            "ConditionParameters": {
                "ImportantFormKey": "*"
                "KeyValueBlockConfidenceLessThan": 60,
                "WordBlockConfidenceLessThan": 90
            }
        }
    ]
}
```

示例 3：使用采样

在以下示例中，5% 的推理是由 Amazon Textract 得出的`AnalyzeDocument`请求将发送给人类工作人员进行审查。Amazon Textract 返回的所有检测到的键/值对都将发送给工作人员以供审核。

```
{
  "Conditions": [
    {
      "ConditionType": "Sampling",
      "ConditionParameters": {
        "RandomSamplingPercentage": 5
      }
    }
  ]
}
```

示例 4：使用 **`MissingImportantFormKey`**

在以下示例中，如果`Mailing Address`或者它的别名，`Mailing Address:`，在 Amazon Textract 检测到的键中缺少，启动人工审核。在使用默认工作人员任务模板时，工作人员 UI 将要求工作人员标识键 `Mailing Address` 或 `Mailing Address:` 及其关联值。

```
{
    "ConditionType": "MissingImportantFormKey",
```

```
    "ConditionParameters": {
        "ImportantFormKey": "Mailing Address",
        "ImportantFormKeyAliases": ["Mailing Address:"]
    }
}
```

示例 5：使用采样和**ImportantFormKeyConfidenceCheck**使用**And**操作者

在此示例中，Amazon Textract 检测到的密钥值对中有 5%，其密钥是Pay Date、PayDate、DateOfPay，或者pay-date对于键/值块的置信度小于 65，组成键和值的每个单词块的置信度小于 85，则发送给工作人员进行审核。

```
{
  "Conditions": [
    {
      "And": [
        {
          "ConditionType": "Sampling",
          "ConditionParameters": {
            "RandomSamplingPercentage": 5
          }
        },
        {
          "ConditionType": "ImportantFormKeyConfidenceCheck",
            "ConditionParameters": {
                "ImportantFormKey": "Pay Date",
                "ImportantFormKeyAliases": [
                    "PayDate",
                    "DateOfPay",
                    "pay-date"
                ],
                "KeyValueBlockConfidenceLessThan": 65,
                "WordBlockConfidenceLessThan": 85
            }
        }
      ]
    }
  ]
}
```

示例 6：使用采样和**ImportantFormKeyConfidenceCheck**使用**And**操作者

使用此示例可以配置人工审核工作流，始终将指定键/值对的低置信度推理发送进行人工审核，并按指定比率抽样键/值对的高置信度推理。

在以下示例中，通过以下方式之一启动人工审核：

- 检测到哪个密钥是其中之一的键值对Pay Date、PayDate、DateOfPay，或者pay-date对于键/值和单词块置信度小于 60，则会发送供人工审核。仅将 Pay Date 表单键（及其别名）和关联值发送给工作人员进行审查。
- 5% 的键值对检测到哪些密钥是其中之一Pay Date、PayDate、DateOfPay，或者pay-date，键/值和单词块置信度大于 90，将发送供人工审核。仅将 Pay Date 表单键（及其别名）和关联值发送给工作人员进行审查。

```
{
  "Conditions": [
    {
      "Or": [
       {
          "ConditionType": "ImportantFormKeyConfidenceCheck",
```

```
            "ConditionParameters": {
                "ImportantFormKey": "Pay Date",
                "ImportantFormKeyAliases": [
                    "PayDate",
                    "DateOfPay",
                    "pay-date"
                ],
                "KeyValueBlockConfidenceLessThan": 60,
                "WordBlockConfidenceLessThan": 60
            }
        },
        {
            "And": [
                {
                    "ConditionType": "Sampling",
                    "ConditionParameters": {
                        "RandomSamplingPercentage": 5
                    }
                },
                {
                    "ConditionType": "ImportantFormKeyConfidenceCheck",
                        "ConditionParameters": {
                            "ImportantFormKey": "Pay Date",
                            "ImportantFormKeyAliases": [
                                "PayDate",
                                "DateOfPay",
                                "pay-date"
                        ],
                        "KeyValueBlockConfidenceLessThan": 90
                        "WordBlockConfidenceGreaterThan": 90
                    }
                }
            ]
        }
      ]
    }
  ]
}
```

示例 7：使用采样和**ImportantFormKeyConfidenceCheck**使用**Or**操作者

在下面的示例中，Amazon TextractAnalyzeDocument操作返回键/值对的键/值对，其键为Pay Date、PayDate、DateOfPay，或者pay-date，键/值块的置信度小于 65，组成键和值的每个单词块的置信度小于 85。此外，所有其他表单中有 5% 的表单启动了人工循环。对于随机选择的每个表单，为该表单检测到的所有键/值对都会发送给人工审核。

```
{
  "Conditions": [
    {
      "Or": [
        {
          "ConditionType": "Sampling",
          "ConditionParameters": {
            "RandomSamplingPercentage": 5
          }
        },
        {
           "ConditionType": "ImportantFormKeyConfidenceCheck",
            "ConditionParameters": {
                "ImportantFormKey": "Pay Date",
                "ImportantFormKeyAliases": [
                    "PayDate",
                    "DateOfPay",
                    "pay-date"
```

```
                ],
                "KeyValueBlockConfidenceLessThan": 65,
                "WordBlockConfidenceLessThan": 85
            }
          }
        }
      ]
    }
  ]
}
```

将人工循环激活条件 JSON 架构与 Amazon Rekognition 结合使用

与 Amazon A2I 一起使用时，Amazon Rekognition 将与结合使用的 Amazon RekognDetectModerationLabels操作支持以下输入：ConditionType参数：

- ModerationLabelConfidenceCheck— 使用此条件类型可在一个或多个指定标签的推理置信度低时创建人工循环。
- Sampling— 使用此条件指定发送进行人工审核的所有推理的百分比。使用此条件来执行以下操作：
  - 通过对模型的所有推理随机抽样并将指定的百分比发送进行人工审核，对 ML 模型进行审计。
  - 使用 ModerationLabelConfidenceCheck 条件随机抽样满足 ModerationLabelConfidenceCheck 中指定的条件的一定百分比的推理以启动人工循环，并仅发送指定的百分比进行人工审核。

  Note

  如果你将同样的请求发送到DetectModerationLabels多次，结果Sampling不会因为该输入的推断而改变。例如，如果你做了DetectModerationLabels请求一次，然后Sampling不会启动人类循环，随后请求DetectModerationLabels使用相同的配置不要启动人类循环。

在创建流程定义时，如果您使用中提供的默认工作人员任务模板人工审核工作流在 Amazon SageMaker 控制台的部分，当工作人员打开任务时，按照这些激活条件发送的供人员审阅的推论将包含在工作人员 UI 中。如果您使用自定义工作人员任务模板，则需要包含 <task.input.selectedAiServiceResponse.blocks> 自定义 HTML 元素来访问这些推理。有关使用此 HTML 元素的自定义模板示例，请参阅 Amazon Rekognition 的自定义模板示例 (p. 2650)。

ModerationLabelConfidenceCheck 输入

对于 ModerationLabelConfidenceCheck ConditionType，支持以下 ConditionParameters：

- ModerationLabelName— 一个的确切（区分大小写）名称ModerationLabelAmazon Rekognition 检测到DetectModerationLabelsoperation. 您可以指定包罗万象的特殊值 (*) 来表示任何审核标签。
- ConfidenceEquals
- ConfidenceLessThan
- ConfidenceLessThanEquals
- ConfidenceGreaterThan
- ConfidenceGreaterThanEquals

当您使用ModerationLabelConfidenceCheck ConditionType中，亚马逊 A2I 为您在中指定的标签发送标签推理ModerationLabelName供人类评论。

采样输入

Sampling ConditionType 现在支持 RandomSamplingPercentage ConditionParameters。RandomSamplingPercentage 参数的输入应是 0.01 到 100 之间的实数。此

数字表示符合人工审核条件并将发送进行人工审核的推理百分比。如果您在不使用任何其他条件的情况下使用 `Sampling` 条件，则此数字表示单个 `DetectModerationLabel` 请求生成的所有推理中，发送供人工审核的百分比。

示例

示例 1：使用**`ModerationLabelConfidenceCheck`**使用**`And`**操作者

以下示例为`HumanLoopActivationConditions`在满足以下一个或多个条件时，条件启动人工循环：

- Amazon Rekognition 检测到`Graphic Male Nudity`置信度介于 90 到 99 之间的审核标签。
- Amazon Rekognition 检测到`Graphic Female Nudity`有信度介于 80 到 99 之间的审核标签。

请注意使用`Or`和`And`为此逻辑建模的逻辑运算符。

虽然在这两个条件中只有一个`Or`运营商需要评估才能`true`为了创建人类循环，亚马逊 Augmented AI 会评估所有条件。人工审核者需要审查评估到的所有条件的审核标签`true`.

```
{
     "Conditions": [{
         "Or": [{
                 "And": [{
                         "ConditionType": "ModerationLabelConfidenceCheck",
                         "ConditionParameters": {
                             "ModerationLabelName": "Graphic Male Nudity",
                             "ConfidenceLessThanEquals": 99
                         }
                     },
                     {
                         "ConditionType": "ModerationLabelConfidenceCheck",
                         "ConditionParameters": {
                             "ModerationLabelName": "Graphic Male Nudity",
                             "ConfidenceGreaterThanEquals": 90
                         }
                     }
                 ]
             },
             {
                 "And": [{
                         "ConditionType": "ModerationLabelConfidenceCheck",
                         "ConditionParameters": {
                             "ModerationLabelName": "Graphic Female Nudity",
                             "ConfidenceLessThanEquals": 99
                         }
                     },
                     {
                         "ConditionType": "ModerationLabelConfidenceCheck",
                         "ConditionParameters": {
                             "ModerationLabelName": "Graphic Female Nudity",
                             "ConfidenceGreaterThanEquals": 80
                         }
                     }
                 ]
             }
         ]
     }]
}
```

示例 2：使用**`ModerationLabelConfidenceCheck`**具有全部捕获值 (*)

在以下示例中，如果检测到置信度大于或等于 75 的任何审核标签，则会启动人工循环。人工审核者需要审核置信度得分大于或等于 75 的所有审核标签。

```
{
    "Conditions": [
        {
            "ConditionType": "ModerationLabelConfidenceCheck",
            "ConditionParameters": {
                "ModerationLabelName": "*",
                "ConfidenceGreaterThanEquals": 75
            }
        }
    ]
}
```

示例 3：使用采样

在下面的示例中，5% 的 Amazon Rekognition 推理来自`DetectModerationLabels`请求将发送给工作人员。使用 SageMaker 控制台中提供的默认工作人员任务模板时，Amazon Rekognition 返回的所有审核标签都将发送给工作人员以供审核。

```
{
  "Conditions": [
    {
      "ConditionType": "Sampling",
      "ConditionParameters": {
        "RandomSamplingPercentage": 5
      }
    }
  ]
}
```

示例 4：使用采样和**`ModerationLabelConfidenceCheck`**使用**`And`**操作者

在这个例子中，有 5% 的 Amazon Rekognition 推断`Graphic Male Nudity`将置信度大于 50 的审核标签发送给工作人员进行审核。使用 SageMaker 控制台中提供的默认工作人员任务模板时，只有`Graphic Male Nudity`标签将发送给工作人员以进行审查。

```
{
  "Conditions": [
    {
      "And": [
        {
          "ConditionType": "Sampling",
          "ConditionParameters": {
            "RandomSamplingPercentage": 5
          }
        },
        {
          "ConditionType": "ModerationLabelConfidenceCheck",
          "ConditionParameters": {
            "ModerationLabelName": "Graphic Male Nudity",
            "ConfidenceGreaterThan": 50
          }
        }
      ]
    }
  ]
}
```

示例 5：使用采样和**`ModerationLabelConfidenceCheck`**使用**`And`**操作者

使用此示例可以配置人工审核工作流，始终将指定标签的低置信度推理发送进行人工审核，并按指定比率抽样标签的高置信度推理。

在以下示例中，通过以下方式之一启动人工审核：

- 对于 `Graphic Male Nudity` 审核标签的推理，在置信度分数小于 60 时将始终发送供人工审核。仅将 `Graphic Male Nudity` 标签发送给工作人员以进行审查。
- 所有推论中的 5%`Graphic Male Nudity`将置信度分数大于 90 的审核标签发送给工人工审核。仅将 `Graphic Male Nudity` 标签发送给工作人员以进行审查。

```
{
  "Conditions": [
    {
      "Or": [
        {
          "ConditionType": "ModerationLabelConfidenceCheck",
          "ConditionParameters": {
            "ModerationLabelName": "Graphic Male Nudity",
            "ConfidenceLessThan": 60
          }
        },
        {
            "And": [
                {
                    "ConditionType": "Sampling",
                    "ConditionParameters": {
                        "RandomSamplingPercentage": 5
                    }
                },
                {
                    "ConditionType": "ModerationLabelConfidenceCheck",
                    "ConditionParameters": {
                        "ModerationLabelName": "Graphic Male Nudity",
                        "ConfidenceGreaterThan": 90
                    }
                }
            ]
        }
      ]
    }
  ]
}
```

示例 6：使用采样和**`ModerationLabelConfidenceCheck`**使用**`Or`**操作者

在以下示例中，如果 Amazon Rekognition 推理响应包含 “图形男性裸体” 标签且推理置信度大于 50，则会创建人工循环。此外，所有其他推理中有 5% 的推理启动了人工循环。

```
{
  "Conditions": [
    {
      "Or": [
        {
          "ConditionType": "Sampling",
          "ConditionParameters": {
            "RandomSamplingPercentage": 5
          }
        },
        {
          "ConditionType": "ModerationLabelConfidenceCheck",
          "ConditionParameters": {
            "ModerationLabelName": "Graphic Male Nudity",
            "ConfidenceGreaterThan": 50
          }
        }
```

```
      ]
    }
  ]
}
```

删除人工审核工作流程

当你删除人工评论工作流程或删除你的Amazon账户在进行人工循环时，您的人工审核工作流程状态更改为`Deleting`. 如果工作人员尚未启动由这些人工循环创建的任务，则 Amazon A2I 将自动停止和删除所有关联的人工循环。如果工作人员已在处理某个任务，则该任务将继续可用，直到完成或过期。只要工作人员仍在处理某个任务，则人工审核工作流程的状态为`Deleting`. 如果完成这些任务，则结果将存储在流定义中指定的 Amazon S3 存储桶中。

删除流定义并不会从 S3 存储桶中删除任何工作人员应答。如果任务已完成，但是您删除了Amazon账户，结果将存储在 Augmented AI 服务存储桶中 30 天，然后永久删除。

删除所有人工循环后，人工审核工作流程将被永久删除。删除人工审核工作流程后，您可以重复使用其名称来创建新的人工审核工作流程。

出于以下任一原因，您可能需要删除人工审核工作流程：

- 您已将数据发送给一组人工审查人员，并希望删除所有未启动的人工循环，因为您希望这些工作人员不再处理这些任务。
- 用于生成工作人员 UI 的工作人员任务模板无法正常呈现或无法按预期运行。

删除人工审核工作流程后，将发生以下更改：

- 人工审核工作流程不再显示在人工审核工作流在 Amazon SageMaker 控制台的 Augmented AI 区域中的页面。
- 当您使用人工审核工作流程名称作为 API 操作的输入时`DescribeFlowDefinition`要么`DeleteFlowDefinition`，Augmented AI 返回`ResourceNotFound`错误消息。
- 当您使用`ListFlowDefinitions`，删除的人工审核工作流程不包括在结果中。
- 使用人工审核工作流程 ARN 作为 Augmented AI 运行时 API 操作的输入时`ListHumanLoops`，Augmented AI 返回`ResourceNotFoundException`.

使用控制台或 SageMaker API 删除流定义

您可以删除人工审核工作流程人工审核工作流页面位于 SageMaker 控制台的 Augmented AI 区域中或使用 SageMaker API。

只有当流定义的状态为 `Active` 时才能将其删除。

删除人工审核工作流程（控制台）

1. 导航到 Augmented AI 控制台https://console.aws.amazon.com/a2i/.
2. 在导航窗格中的下Augmented AI部分，选择人工审核工作流.
3. 选择要删除的人工审核工作流程的超链接名称。
4. 在存储库的摘要人工审查工作流程的页面，选择Delete.
5. 在要求您确认删除人工审查工作流程的对话框中，选择 Delete (删除)。

系统会自动将您重定向到 Human review workflows (人工审查工作流程) 页面。在删除人工审查工作流程时，该工作流程的状态列中将显示 Deleting (正在删除) 状态。删除该工作流程后，它不会显示在此页面上的工作流程列表中。

删除人工审核工作流程 (API)

使用 SageMaker 可删除人工审核工作流程（流定义）DeleteFlowDefinitionAPI 操作。通过 Amazon CLI 和各种语言特定的开发工具包支持此 API 操作。下表显示了使用 SDK for Python (Boto3) 和Amazon CLI要删除人工审查工作流程，*example-flow-definition*.

Amazon SDK for Python (Boto3)

以下请求示例使用 SDK for Python (Boto3) 删除人工审核工作流程。有关更多信息，请参阅 。delete_flow_定义中的AmazonSDK for Python (Boto) API 参考.

```
import boto3

sagemaker_client = boto3.client('sagemaker')
response = sagemaker_client.delete_flow_definition(FlowDefinitionName='example-flow-definition')
```

Amazon CLI

以下请求示例使用Amazon用于删除人工审核工作流程的 CLI。有关更多信息，请参阅 。删除flow-flow-flow-flow-e-中的Amazon CLI命令参考.

```
$ aws sagemaker delete-flow-definition --flow-definition-name 'example-flow-definition'
```

如果此操作成功，则 Augmented AI 会发送回带有空 HTTP 正文的 HTTP 200 响应。

创建和启动人工循环

人工循环 启动人工审核工作流，并将数据审核任务发送给工作人员。如果您使用某种 Amazon A2I 内置任务类型，则将显示相应的Amazon如果满足流定义中指定的条件，则服务会代表您创建并启动人工循环。如果未在流定义中指定任何条件，则为每个对象创建一个人工循环。在将 Amazon A2I 用于自定义任务时，应用程序调用时会启动人工循环。`StartHumanLoop`.

按照以下说明操作，使用 Amazon Rekognition 或 Amazon Textract 内置任务类型以及自定义任务类型配置人工循环。

先决条件

要创建和启动人工循环，您必须附加`AmazonAugmentedAIFullAccess`的策略Amazon Identity and Access Management(IAM) 用户或角色，用于配置或启动人工循环。这是您用来通过使用配置人工循环的身份。`HumanLoopConfig`对于内置任务类型。对于自定义任务类型，这是您用于调用的身份。`StartHumanLoop`.

此外，在使用内置任务类型时，您的 IAM 用户或角色必须有权调用Amazon与您的任务类型相关联的服务。例如，如果您使用带 Augmented AI 的 Amazon Rekognition，您必须附加调用所需的权限。`DetectModerationLabels`. 有关可用于授予这些权限的基于身份的策略的示例，请参阅 Amazon Rekognition 基于身份的策略示例和 Amazon Textract 基于身份的策略示例。您也可以使用更一般的策略`AmazonAugmentedAIIntegratedAPIAccess`以授予这些权限。有关更多信息，请参阅 创建具有调用 Amazon A2I、Amazon Textract 和 Amazon Rekognition API 操作权限的 IAM 用户 (p. 2665)。

要创建和启动人工循环，您需要提供流定义 ARN。要了解如何创建流定义（或人工审核工作流），请参阅创建人工审核工作流程 (p. 2622)。

Important

Amazon A2I 要求所有包含人工循环输入映像数据的 S3 存储桶都必须附加 CORS 策略。要详细了解有关此更改的信息，请参CORS 权限要求 (p. 2663).

为内置任务类型创建和启动人工循环

要使用内置任务类型启动人工循环，请使用相应服务的 API 来提供输入数据并配置人工循环。对于 Amazon Textract，您可以使用`AnalyzeDocument`API 操作。对于 Amazon Rekognition，您可以使用`DetectModerationLabels`API 操作。您可以使用Amazon CLI或特定于语言的开发工具包以使用这些 API 操作来创建请求。

> Important
>
> 当使用内置任务类型创建人工循环时，您可以使用`DataAttributes`指定一组`ContentClassifiers`与向提供的输入相关`StartHumanLoop`operation. 使用内容分类器声明您的内容不含个人身份信息或成人内容。
> 要使用 Amazon Mechanical Turk，请确保您的数据中没有个人身份信息，包括 HIPAA 规定的受保护的健康信息。加入`FreeOfPersonallyIdentifiableInformation`内容分类器。如果你不使用此内容分类器， SageMaker 不会将你的任务发送给 Mechanical Turk。如果您的数据不含成人内容，请包含`'FreeOfAdultContent'`分类器。如果您不使用这些内容分类器， SageMaker 可能会限制可以查看您的任务的 Mechanical Turk 员。

在使用内置任务类型启动 ML 作业后Amazon服务 API，Amazon A2I 监控该服务的推理结果。例如，在使用 Amazon Rekognition 运行作业时，Amazon A2I 会检查每个图像的推理置信度分数，并将其与流定义中指定的置信度阈值进行比较。如果满足启动人工审核任务的条件，或者如果您未在流定义中指定条件，则人工审核任务将发送到工作人员。

创建一个 Amazon Textract 人工循环

Amazon A2I 将与 Amazon Textract 集成，以便您能够使用 Amazon Textract API 配置和启动人工循环。要将文档文件发送到 Amazon Textract 以进行文档分析，您可以使用 Amazon Textract`AnalyzeDocument`API 操作. 要向此文档分析作业添加人工循环，必须配置参数`HumanLoopConfig`.

当配置人工循环时，您在中指定的流定义`FlowDefinitionArn`的`HumanLoopConfig`必须位于相同的位置Amazon区域作为中标识的存储桶`Bucket`的`Document`参数。

下表显示了如何将此操作与Amazon CLI和Amazon SDK for Python (Boto3).

Amazon SDK for Python (Boto3)

以下请求示例使用 SDK for Python (Boto3)。有关更多信息，请参阅 。分析 _文档中的Amazon适用于 Python 的开发工具包 (Boto) API 参考.

```
import boto3

textract = boto3.client('textract', aws_region)

response = textract.analyze_document(
            Document={'S3Object': {'Bucket': bucket_name, 'Name': document_name}},
            FeatureTypes=["TABLES", "FORMS"],
            HumanLoopConfig={
                'FlowDefinitionArn':
 'arn:aws:sagemaker:aws_region:aws_account_number:flow-definition/flow_def_name',
                'HumanLoopName': 'human_loop_name',
                'DataAttributes': {'ContentClassifiers':
 ['FreeOfPersonallyIdentifiableInformation','FreeOfAdultContent']}
            }
          )
```

Amazon CLI

以下请求示例使用 Amazon CLI。有关更多信息，请参阅 。文档分析中的Amazon CLI命令参考.

```
$ aws textract analyze-document \
```

```
    --document '{"S3Object":{"Bucket":"bucket_name","Name":"document_name"}}' \
    --human-loop-config
 HumanLoopName="human_loop_name",FlowDefinitionArn="arn:aws:sagemaker:aws-
region:aws_account_number:flow-
definition/
flow_def_name",DataAttributes='{ContentClassifiers=["FreeOfPersonallyIdentifiableInformation",
 "FreeOfAdultContent"]}' \
    --feature-types '["TABLES", "FORMS"]'
```

```
$ aws textract analyze-document \
    --document '{"S3Object":{"Bucket":"bucket_name","Name":"document_name"}}' \
    --human-loop-config \

 '{"HumanLoopName":"human_loop_name","FlowDefinitionArn":"arn:aws:sagemaker:aws_region:aws_account_
definition/flow_def_name","DataAttributes": {"ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation","FreeOfAdultContent"]}}' \
    --feature-types '["TABLES", "FORMS"]'
```

跑完之后AnalyzeDocument配置了人类循环后，Amazon A2I 会监控来自AnalyzeDocument并根据流定义的激活条件检查该内容。如果一个或多个键值对的 Amazon Textract 推理置信度分数满足审核条件，将启动人工审核循环，并包含HumanLoopActivationOutput在中的对象执行AnalyzeDocument响应。

创建 Amazon Rekognition 人类循环

Amazon A2I 将与 Amazon Rekognition 集成，以便您能够使用 Amazon Rekognition API 配置和启动人工循环。要将图像发送到 Amazon Rekognition 以进行内容审核，您可以使用 Amazon RekognitionDetectModerationLabelsAPI 操作. 要配置人工循环，请在配置 DetectModerationLabels 时设置 HumanLoopConfig 参数。

当配置人工循环时，您在中指定的流定义FlowDefinitionArn的HumanLoopConfig必须位于相同的位置Amazon区域为中标识的 S3 存储桶Bucket的Image参数。

下表显示了如何将此操作与Amazon CLI和Amazon SDK for Python (Boto3).

Amazon SDK for Python (Boto3)

以下请求示例使用 SDK for Python (Boto3)。有关更多信息，请参阅 。detect_moderation_label中的Amazon适用于 Python 的开发工具包 (Boto) API 参考.

```
import boto3

rekognition = boto3.client("rekognition", aws_region)

response = rekognition.detect_moderation_labels( \
        Image={'S3Object': {'Bucket': bucket_name, 'Name': image_name}}, \
        HumanLoopConfig={ \
            'HumanLoopName': 'human_loop_name', \
            'FlowDefinitionArn': ,
 "arn:aws:sagemaker:aws_region:aws_account_number:flow-definition/flow_def_name" \
            'DataAttributes': {'ContentClassifiers':
 ['FreeOfPersonallyIdentifiableInformation','FreeOfAdultContent']}
         })
```

Amazon CLI

以下请求示例使用 Amazon CLI。有关更多信息，请参阅 。检测适用标签中的Amazon CLI命令参考.

```
$ aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket='bucket_name',Name='image_name'}" \
```

```
    --human-loop-config
 HumanLoopName="human_loop_name",FlowDefinitionArn="arn:aws:sagemaker:aws_region:aws_account_number
definition/
flow_def_name",DataAttributes='{ContentClassifiers=["FreeOfPersonallyIdentifiableInformation",
 "FreeOfAdultContent"]}'
```

```
$ aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket='bucket_name',Name='image_name'}" \
    --human-loop-config \
        '{"HumanLoopName": "human_loop_name", "FlowDefinitionArn":
 "arn:aws:sagemaker:aws_region:aws_account_number:flow-definition/flow_def_name",
 "DataAttributes": {"ContentClassifiers": ["FreeOfPersonallyIdentifiableInformation",
 "FreeOfAdultContent"]}}'
```

跑完之后DetectModerationLabels配置了人类循环后，Amazon A2I 会监控来自DetectModerationLabels并根据流定义的激活条件检查该内容。如果一个图像的 Amazon Rekognition 推理置信度分数满足审核条件，将启动人工审核循环，并包含响应元素。HumanLoopActivationOutput中的DetectModerationLabels响应。

为自定义任务类型创建和启动人工循环

要为自定义人工审核任务配置人工循环，请在应用程序中使用 StartHumanLoop 操作。此部分提供了一个使用 Amazon SDK for Python (Boto3) 和 Amazon Command Line Interface (Amazon CLI) 的人工循环请求的示例。有关支持的其他特定于语言的 SDK 的文档StartHumanLoop，使用另请参阅部分StartHumanLoop在亚马逊 Augmented AI 运行时 API 文档中。请参阅使用 Amazon A2I 使用案例和示例 (p. 2616)以查看演示如何将 Amazon A2I 与自定义任务类型结合使用的示例。

先决条件

要完成此过程，您需要：

- 格式设置为 JSON 格式文件的字符串表示形式的输入数据
- 流定义的 Amazon 资源名称 (ARN)

配置人工循环

1. 对于 DataAttributes，指定一组与提供给 StartHumanLoop 操作的输入相关的 ContentClassifiers。使用内容分类器声明您的内容不含个人身份信息或成人内容。

   要使用 Amazon Mechanical Turk，请确保您的数据中没有个人身份信息，包括 HIPAA 下的受保护的健康信息，并包括FreeOfPersonallyIdentifiableInformation内容分类器。如果你不使用此内容分类器， SageMaker 不会将你的任务发送给 Mechanical Turk。如果您的数据不含成人内容，请包含'FreeOfAdultContent'分类器。如果您不使用这些内容分类器， SageMaker 可能会限制可以查看您的任务的 Mechanical Turk 员。
2. 对于 FlowDefinitionArn，输入流定义的 Amazon 资源名称 (ARN)。
3. 对于 HumanLoopInput，以 JSON 格式文件的字符串表示形式输入您的输入数据。构建输入数据和自定义工作人员任务模板，以便在您启动人工循环时，向工作人员正确显示输入数据。请参阅预览工作人员任务模板 (p. 2652)以了解如何预览自定义工作人员任务模板。
4. 对于 HumanLoopName，输入人工循环的名称。该名称在您账户所在的区域中必须唯一，并且最多可以包含 63 个字符。有效字符为 a-z、0-9 和- (连字符) 。

启动人工循环

- 要启动人工循环，请使用首选的特定于语言的开发工具包提交与以下示例类似的请求。

Amazon SDK for Python (Boto3)

以下请求示例使用 SDK for Python (Boto3)。有关更多信息，请参阅 。Boto 3 Augmented AI 运行时中的Amazon适用于 Python 的开发工具包 (Boto) API 参考.

```
import boto3

a2i_runtime_client = boto3.client('sagemaker-a2i-runtime')

response = a2i_runtime_client.start_human_loop(
    HumanLoopName='human_loop_name',
    FlowDefinitionArn='arn:aws:sagemaker:aws-region:xyz:flow-definition/flow_def_name',
    HumanLoopInput={
        'InputContent': '{"InputContent": {\"prompt\":\"What is the answer?\"}}'
    },
    DataAttributes={
        'ContentClassifiers': [
            'FreeOfPersonallyIdentifiableInformation'|'FreeOfAdultContent',
        ]
    }
)
```

Amazon CLI

以下请求示例使用 Amazon CLI。有关更多信息，请参阅 。开始人类循环中的Amazon CLI命令参考.

```
$ aws sagemaker-a2i-runtime start-human-loop
        --flow-definition-arn 'arn:aws:sagemaker:aws_region:xyz:flow-
definition/flow_def_name' \
        --human-loop-name 'human_loop_name' \
        --human-loop-input '{"InputContent": "{\"prompt\":\"What is the answer?\"}"}' \
        --data-attributes
 ContentClassifiers="FreeOfPersonallyIdentifiableInformation","FreeOfAdultContent" \
```

当通过调用成功启动人工循环时StartHumanLoop直接，响应包括HumanLoopARN和HumanLoopActivationResults对象设置为NULL. 您可以使用此人工循环名称来监控和管理您的人工循环。

后续步骤:

在启动人工循环后，您可以使用亚马逊 Augmented AI 运行时 API 和亚马逊来管理和监控该循环。CloudWatch 事件。要了解更多信息，请参阅“监控和管理您的人工循环 (p. 2653)”。

删除人工循环

当你删除人类循环时，状态更改为Deleting. 删除人类循环后，工作人员不再可以执行相关的人工审查任务。在以下情况之一中，您可能需要删除人工循环：

- 用于生成工作人员用户界面的工作人员任务模板无法正常呈现或无法按预期运行。
- 单个数据对象被意外多次发送给工作人员。
- 您不再需要由人类审核的数据对象。

如果人类循环的状态是InProgress，你必须在删除人类循环之前停止它。当您停止人工循环时，该状态将变为Stopping在停止时。当状态更改为Stopped，可以删除人工循环。

当停止关联人工循环时，如果人工工已在处理某个任务，则该任务将继续可用，直到完成或过期。只要工作人员仍在处理某个任务，人工循环的状态就是Stopping. 如果完成这些任务，结果将存储在人工审查工作流程中指定的 Amazon S3 存储桶 URI 中。如果工作人员在没有提交工作的情况下离开任务，则该任务将停止，工作人员无法返回任务。如果没有工作人员开始处理任务，则会立即停止该任务。

如果删除Amazon用于创建人类循环的帐户，它会自动停止并删除。

人类循环数据保留和删除

当人工完成人工审查任务时，结果将存储在您在用于创建人工循环的人工审核工作流程中指定的 Amazon S3 输出存储桶中。删除或停止人工循环不会从 S3 存储桶中删除任何工作人员应答。

此外，出于以下原因，Amazon A2I 在内部暂时存储人工循环输入和输出数据：

- 如果您配置人工循环以便将单个数据对象发送给多个工作程序进行审核，那么在所有工作人员完成审核任务之前，Amazon A2I 不会将输出数据写入 S3 存储桶。Amazon A2I 在内部存储部分答案（来自单个员工的答案），以便它可以将完整的结果写入 S3 存储桶。
- 如果您报告了低质量的人工评论结果，亚马逊 A2I 可以调查并回应您的问题。
- 如果您无法访问或删除用于创建人类循环的人工审阅工作流程中指定的输出 S3 存储桶，并且任务已发送给一个或多个工作人员，Amazon A2I 需要一个地方来临时存储人工审查结果。

在人类循环状态更改为以下状态之一后 30 天内，Amazon A2I 会在内部删除此数据：Deleted、Stopped，或者Completed. 换句话说，数据将在人类循环完成、停止或删除 30 天后删除。此外，如果您关闭了，此数据将在 30 天后删除Amazon用于创建关联的人类循环的帐户。

使用控制台或 Amazon A2I API 停止和删除流定义

您可以在 Augmented AI 控制台中或使用 SageMaker API 停止和删除人类循环。当人工循环被删除后，该状态将变为Deleted.

删除人类循环（控制台）

1. 导航到 Augmented AI 控制台https://console.aws.amazon.com/a2i/.
2. 在导航窗格中的Augmented AI部分中，选择人工审核工作流.
3. 选择用于创建要删除的人工循环的人工审查工作流程的超链接名称。
4. 在人类循环在页面底部的部分中，选择要停止和删除的人工循环。
5. 如果人工循环状态为Completed、Stopped，或者Failed，选择Delete.

   如果人工循环状态是InProgress，选择停止. 当状态更改为Stopped，选择Delete.

删除人类循环 (API)

1. 使用 Augmented AI 运行时 API 操作检查人工循环的状态DescribeHumanLoop. 请参阅下表中的使用此操作的示例。

   Amazon SDK for Python (Boto3)

   以下示例使用 SDK for Python (Boto3) 描述名为##-####. 有关更多信息，请参阅 。描述了_human_LOOP中的Amazon适用于 Python 的开发工具包 (Boto) API 参考.

```
import boto3

a2i_runtime_client = boto3.client('sagemaker-a2i-runtime')
```

```
response = a2i_runtime_client.describe_human_loop(HumanLoopName='example-human-loop')
human_loop_status = response['HumanLoopStatus']
print(f'example-human-loop status is: {human_loop_status}')
```

Amazon CLI

以下示例使用AmazonCLI 来描述名为的人类循环*##-####*. 有关更多信息，请参阅 。描述人类循环中的Amazon CLI命令参考.

```
$ aws sagemaker-a2i-runtime describe-human-loop --human-loop-name 'example-human-loop'
```

2. 如果流程定义状态为`Completed`、`Stopped`，或者`Failed`中，使用 Augmented AI 运行时 API 操作删除流程定义`DeleteHumanLoop`.

   Amazon SDK for Python (Boto3)

   以下示例使用 SDK for Python (Boto3) 删除名为*##-####*. 有关更多信息，请参阅 。delete_human_LOOP中的Amazon适用于 Python 的开发工具包 (Boto) API 参考.

   ```
   import boto3

   a2i_runtime_client = boto3.client('sagemaker-a2i-runtime')
   response = a2i_runtime_client.delete_human_loop(HumanLoopName='example-human-loop')
   ```

   Amazon CLI

   以下示例使用AmazonCLI 删除名为的人类循环*##-####*. 有关更多信息，请参阅 。删除人类循环中的Amazon CLI命令参考.

   ```
   $ aws sagemaker-a2i-runtime delete-human-loop --human-loop-name 'example-human-loop'
   ```

如果人工循环状态为`InProgress`使用停止人工循环`StopHumanLoop`然后使用`DeleteHumanLoop`删除它。

Amazon SDK for Python (Boto3)

以下示例使用 SDK for Python (Boto3) 描述名为*##-####*. 有关更多信息，请参阅 。stop_human_LOOP中的Amazon适用于 Python 的开发工具包 (Boto) API 参考.

```
import boto3

a2i_runtime_client = boto3.client('sagemaker-a2i-runtime')
response = a2i_runtime_client.stop_human_loop(HumanLoopName='example-human-loop')
```

Amazon CLI

以下示例使用AmazonCLI 来描述名为的人类循环*##-####*. 有关更多信息，请参阅 。停止人类循环中的Amazon CLI命令参考.

```
$ aws sagemaker-a2i-runtime stop-human-loop --human-loop-name 'example-human-loop'
```

创建和管理工作人员任务模板

您可以通过创建一个Worker 任务模板. 工作人员任务模板是一个 HTML 文件，用于显示输入数据和帮助员工完成任务的说明。

对于 Amazon Rekognition 或 Amazon Textract 任务类型，您可以使用图形用户界面 (GUI) 自定义预先制作的工作人员任务模板，避免与 HTML 代码交互。对于此选项，请使用中的说明创建人工审核工作流程（控制台）(p. 2623)以创建人工审核工作流程并在 Amazon SageMaker 控制台中自定义工作人员任务模板。使用这些说明创建模板后，该模板将显示在Augmented AI 控制台.

如果要为自定义任务类型创建人工审核工作流，则必须创建工作人员任务模板使用 HTML 代码。有关更多信息，请参阅 创建自定义 Worker 任务模板 (p. 2648)。

如果您使用 HTML 创建模板，则必须使用此模板生成亚马逊 A2IAmazon Resource Name (ARN) 人工任务 UI在 Amazon A2I 控制台中。此 ARN 具有以下格式：`arn:aws:sagemaker:<aws-region>:<aws-account-number>:human-task-ui/<template-name>`. 此 ARN 与您可以在一个或多个人工审核工作流程（流程定义）中使用的工作人员任务模板资源相关联。

按照中找到的说明操作，使用工作人员任务模板来创建人工任务 UI ARN创建工作人员任务模板 (p. 2647)或者通过使用`CreateHumanTaskUi`。API 操作。

主题

- 创建和删除工作人员任务模板 (p. 2647)
- 创建自定义 Worker 任务模板 (p. 2648)
- 创建良好的工作人员说明 (p. 2652)

创建和删除工作人员任务模板

可以使用工作人员模板来自定义在工作人员在处理任务时显示的界面和说明。按照此页面上的说明执行操作，在 Amazon SageMaker 控制台的 “Augmented AI” 区域中创建工作人员任务模板。对于 Amazon Textract 和 Amazon Rekognition 任务，提供了初学者模板。要了解如何使用 HTML crowd 元素自定义您的模板，请参阅创建自定义 Worker 任务模板 (p. 2648)。

在 SageMaker 控制台的 “Augmented AI” 区域的 “工作人员任务模板” 页面中创建工作人员模板时，将生成工作人员任务模板 ARN。使用此 ARN 作为输入`HumanTaskUiArn`在使用 API 操作创建流定义时`CreateFlowDefinition`. 在控制台的 “人工审查工作流程” 页面上创建人工审查工作流时，可以选择此模板。

如果您要为 Amazon Textract 或 Amazon Rekognition 任务类型创建工作人员任务模板资源，则可以在工作人员任务模板控制台页面上预览从模板生成的工作人员 UI。您必须附加中描述的策略启用工作人员任务模板预览 (p. 2666)转换为您用于预览模板的 IAM 角色。

创建工作人员任务模板

您可以使用 SageMaker 控制台和使用 SageMaker API 操作创建工作人员任务模板。`CreateHumanTaskUi`.

创建工作人员任务模板（控制台）

1. 在以下位置打开 Amazon A2I 控制台https://console.aws.amazon.com/a2i/.
2. 在Amazon Augmented AI在左侧导航窗格中，选择员工任务模板.
3. 选择 Create template (创建模板)。
4. 在 Template name (模板名称) 中，输入一个唯一名称。
5. （可选）输入IAM 角色这授予 Amazon A2I 代表您调用服务所需的权限。

6. In模板类型中，从下拉列表中选择模板类型。如果要为 Textract-form extraction (Textract 表单提取) 或 Rekognition -image moderation (Rekognition 图像审核) 任务创建模板，请选择适当的选项。
7. 输入您的自定义模板元素，如下所示：
   - 如果您选择了 Amazon Textract 或 Amazon Rekognition 任务模板，则模板编辑器自动填充可自定义的默认模板。
   - 如果您使用的是自定义模板，请在编辑器中输入预定义的模板。
8. （可选）要完成此步骤，您必须向 IAM 角色 ARN 提供读取在中的用户界面上呈现的 Amazon S3 对象的权限步骤 5.

   只能在为 Amazon Textract 或 Amazon Rekognition 创建模板时预览模板。

   选择查看预览预览工作人员看到的界面和说明。这是一个交互式预览。在完成示例任务并选择 Submit (提交) 后，您将看到刚刚执行的任务所生成的输出。

   如果要为自定义任务类型创建工作人员任务模板，您可以使用 `RenderUiTemplate` 预览工作人员任务 UI。有关更多信息，请参阅 预览工作人员任务模板 (p. 2652)。
9. 如果您对模板感到满意，请选择 Create (创建)。

创建模板后，您可以在控制台中创建人工审核工作流程时选择该模板。您的模板也会显示在Amazon Augmented AISageMaker 控制台的部分员工任务模板. 选择您的模板可查看其 ARN。ARN 使用`CreateFlowDefinition`API 操作。

使用工作人员任务模板 (API) 创建工作人员任务模板

要使用 SageMaker API 操作生成工作人员任务模板`CreateHumanTaskUi`中，为您的 UI 指定一个名称`HumanTaskUiName`然后在中输入你的 HTML 模板`Content`下`UiTemplate`. 查找有关支持此 API 操作的语言特定 SDK 的文档，请在另请参阅的 部分`CreateHumanTaskUi`.

删除工作人员任务模板

创建工作器任务模板后，可以使用 SageMaker 控制台或 SageMaker API 操作将其删除`DeleteHumanTaskUi`.

删除工作人员任务模板时，无法使用使用该模板创建的人工审阅工作流程（流程定义）来启动人工循环。已使用您删除的工作人员任务模板创建的任何人工循环将继续处理，直到完成，不会受到影响。

删除工作人员任务模板（控制台）

1. 在以下位置打开 Amazon A2I 控制台https://console.aws.amazon.com/a2i/.
2. 在左侧导航窗格中的 Amazon Advanced Augmented AI 下，选择员工任务模板.
3. 选择要删除的模板。
4. SeleteDelete.
5. 似乎有模态确认您的选择。选择删除。

删除工作人员任务模板 (API)

要使用 SageMaker API 操作删除工作人员任务模板`DeleteHumanTaskUi`，在中指定 UI 的名称`HumanTaskUiName`.

创建自定义 Worker 任务模板

Crowd HTML 元素 是提供多种任务小部件和设计元素的 Web 组件，您可以根据您要提出的问题定制这些组件。您可以使用这些人群元素创建自定义工作人员模板，并将其与 Amazon Augmented AI (Amazon A2I) 人工审核工作流集成，以自定义工作人员控制台和说明。

有关可供 Amazon A2I 用户使用的所有 HTML 人群元素的列表，请参阅Crowd HTML 元素参考 (p. 485). 有关模板示例，请参阅AmazonGitHub 存储库，其中包含 60 多个示例自定义任务模板。

在本地开发模板

在控制台中测试模板将如何处理传入数据时，您可以通过在 HTML 文件的顶部添加以下代码，在浏览器中测试模板的 HTML 和自定义元素的外观。

这会加载必要的代码来呈现自定义 HTML 元素。如果您希望在常用编辑器而非在控制台中开发您的模板的外观，则使用此代码。

此代码不会解析您的变量。您可能希望在本地开发时将它们替换示例内容。

使用外部资产

借助 Amazon Augmented AI 自定义模板，您可以嵌入外部脚本和样式表。例如，以下标题将位于 `https://www.example.com/my-enhancement-styles.css` 的 `text/css` 样式表名称 `stylesheet` 嵌入自定义模板中。

Example

```
<script src="https://www.example.com/my-enhancment-script.js"></script>
<link rel="stylesheet" type="text/css" href="https://www.example.com/my-enhancement-
styles.css">
```

如果您遇到错误，请确保您的原始服务器随资产一起发送正确的 MIME 类型和编码标头。

例如，对于远程脚本，MIME 和编码类型为：`application/javascript;CHARSET=UTF-8`.

对于远程样式表，MIME 和编码类型为：`text/css;CHARSET=UTF-8`。

跟踪您的变量

构建自定义模板时，您必须向其中添加变量，用于表示可能在不同任务之间或者在不同工作人员之间更改的数据片段。如果您以其中一个示例模板开始，您需要确保您知道它已经使用的变量。

例如，对于将 Augmented AI 人工审核循环与 Amazon Textract 文本审核任务集成的自定义模板，`{{ task.input.selectedAiServiceResponse.blocks }}`用于初始值输入数据。对于 Amazon Augmented AI (Amazon Augmented AI (Amazon Augmented AI (Amazon A2I) 与 Amazon Rekognition 集成，`{{ task.input.selectedAiServiceResponse.moderationLabels }}`已使用。对于自定义任务类型，您需要确定任务类型的输入参数。使用指定了 *`customInputValuesForStartHumanLoop`* 的 `{{ task.input.`*`customInputValuesForStartHumanLoop`*`}}`。

Amazon Textract 的自定义模板示例

所有自定义模板都以 `<crowd-form> </crowd-form>` 元素开始和结束。与标准 HTML `<form>` 元素类似，所有格式代码都应位于这些元素之间。

对于 Amazon Textract 文档分析任务，请使用`<crowd-textract-analyze-document>`元素。它使用以下属性：

- `src`— 指定要注释的图像文件的 URL。
- `initialValue`— 为工作人员 UI 中的属性设置初始值。
- `blockTypes`（必需）— 确定工作人员可以执行的分析类型。当前仅支持 `KEY_VALUE_SET`。
- `keys`（必需）— 指定工作人员可以添加的新密钥和关联的文本值。
- `no-key-edit`（必需）— 防止工作人员编辑通过传递的注释的键`initialValue`.
- `no-geometry-edit`— 防止工作人员编辑通过传递的注释的多边形`initialValue`.

对于孩子<crowd-textract-analyze-document>元素中，您必须有两个区域。您可以在这些区域中使用任意 HTML 和 CSS 元素。

- <full-instructions>— 提供的说明。查看完整说明该工具中的链接。您可以将此项留空，但我们建议您提供完整的说明以获得更好结果。
- <short-instructions>— 工具侧栏中显示的任务的简要描述。您可以将此项留空，但我们建议您提供完整的说明以获得更好结果。

Amazon Textract 模板类似于如下所示。

Example

Amazon Rekognition 的自定义模板示例

所有自定义模板都以 <crowd-form> </crowd-form> 元素开始和结束。与标准 HTML <form> 元素类似，所有格式代码都应位于这些元素之间。对于 Amazon Rekognition 自定义任务模板，请使用<crowd-rekognition-detect-moderation-labels>元素。此元素支持以下属性：

- categories— 字符串数组要么对象数组，其中每个对象都具有name字段中返回的子位置类型。
  - 如果类别作为对象提供，则以下情况适用：
    - 显示的类别是 name 字段的值。
    - 返回的答复包含任何选定类别的完整 对象。
  - 如果类别作为字符串提供，则以下情况适用：
    - 返回的答复是选择的所有字符串的数组。
- exclusion-category— 通过设置此属性，您可以在 UI 中的类别下创建一个按钮。当用户选择该按钮时，将取消选择并禁用所有类别。如果工作人员再次选择该按钮，您将重新允许用户选择类别。如果工作人员通过选择选择提交任务提交在您选择该按钮后，该任务将返回一个空数组。

对于孩子<crowd-rekognition-detect-moderation-labels>元素中，您必须有两个区域。

- <full-instructions>— 提供的说明。查看完整说明该工具中的链接。您可以将此项留空，但我们建议您提供完整的说明以获得更好结果。
- <short-instructions>— 工具侧栏中显示的任务的简要描述。您可以将此项留空，但我们建议您提供完整的说明以获得更好结果。

使用这些元素的模板类似于如下所示。

使用 Liquid 添加自动化功能

自定义模板系统使用 Liquid 实现自动化。Liquid 是开源内联标记语言。有关更多信息和文档，请参阅 Liquid 主页。

在 Liquid 中，单大括号之间的文本以及百分比符号是指令或标签执行诸如控制流或迭代之类的操作。双大括号之间的文本是一个变量或用于输出变量值的对象。以下列表包括两种类型的液体标签，您可能会发现这些标签对自动化模板输入数据处理有用。如果选择以下标签类型之一，则会将您重定向到 Liquid 文档。

- 控制流：包括编程逻辑运算符if/else、unless, 和case/when.
- 迭代: 使您能够使用 for 循环之类的语句重复运行代码块。

  例如，以下代码示例说明了如何使用 Liquidfor创建标签for循环。此示例循环浏览moderationLabels从 Amazon Rekognition 返回并显示moderationLabels属性name和parentName供员工审查：

```
{% for label in task.input.selectedAiServiceResponse.moderationLabels %}
```

```
    {
      name: &amp;quot;{{ label.name }}&amp;quot;,
      parentName: &amp;quot;{{ label.parentName }}&amp;quot;,
    },
{% endfor %}
```

使用变量筛选条件

除了标准液体过滤器和操作，Amazon Augmented AI (Amazon Augmented AI (Amazon A2I) 提供了额外的筛选条件 您可以在变量名称之后放置竖线 (|) 字符，然后指定筛选条件名称，从而应用筛选条件。要链接筛选条件，请使用以下格式。

Example

```
{{ <content> | <filter> | <filter> }}
```

自动转义和显式转义

默认情况下，输入将进行 HTML 转义，以防止在变量文本和 HTML 之间产生混淆。您可以明确添加 `escape` 筛选条件，使其对于读取正在进行转义的模板源的用户更显而易见。

escape_once

`escape_once` 可确保您的代码已经转义，而不会再次重新转义。例如，它可以确保`&`不会变为`&`.

skip_autoescape

当您的内容要用作 HTML 时，`skip_autoescape` 很有用。例如，您可能在边界框的完整说明中有一些文本段落和一些图像。

> **Note**
>
> 请谨慎使用 `skip_autoescape`。对于模板，最佳实践是避免使用 `skip_autoescape` 传递功能代码或标记，除非您绝对确信您对正传递的内容具有严格的控制。如果您传递用户输入，您可能会让您的工作线程遭受跨站点脚本攻击。

to_json

`to_json` 对您提供给 JavaScript 对象表示法 (JSON) 的数据进行编码。如果您提供了一个对象，它会对该对象进行序列化。

grant_read_access

`grant_read_access`获取 Amazon Simple Storage Service (Amazon S3) URI 并将其编码为 HTTPS URL（具有针对该资源的短期访问令牌）。这样，就可以向工作人员显示存储在 S3 存储桶中但原本无法公开访问的照片、音频或视频对象。

Example to_json 和 grant_read_access 筛选条件示例

输入

```
auto-escape: {{ "Have you read 'James & the Giant Peach'?" }}
explicit escape: {{ "Have you read 'James & the Giant Peach'?" | escape }}
explicit escape_once: {{ "Have you read 'James &amp; the Giant Peach'?" | escape_once }}
skip_autoescape: {{ "Have you read 'James & the Giant Peach'?" | skip_autoescape }}
to_json: {{ jsObject | to_json }}
grant_read_access: {{ "s3://examplebucket/myphoto.png" | grant_read_access }}
```

Example

输出

```
auto-escape: Have you read &amp;#39;James &amp;amp; the Giant Peach&amp;#39;?
explicit escape: Have you read &amp;#39;James &amp;amp; the Giant Peach&amp;#39;?
explicit escape_once: Have you read &amp;#39;James &amp;amp; the Giant Peach&amp;#39;?
skip_autoescape: Have you read 'James & the Giant Peach'?
to_json: { "point_number": 8, "coords": [ 59, 76 ] }
grant_read_access: https://s3.amazonaws.com/examplebucket/myphoto.png?<access token and
 other params>
```

Example 自动化分类模板的示例。

要自动执行此简单文本分类示例，请包括 Liquid 标签 `{{ task.input.source }}`。此示例使用 crowd-classifier (p. 495) 元素。

预览工作人员任务模板

要预览自定义工作人员任务模板，请使用 SageMaker`RenderUiTemplate`operation. 您可以使用`RenderUiTemplate`使用Amazon CLI或者您的首选AmazonSDK。有关此 API 操作支持的特定语言 SDK 的文档，请参阅`See Also`的 部分`RenderUiTemplate`.

先决条件

要预览工作人员任务模板，您使用的 Amazon Identity and Access Management (IAM) 角色 Amazon 资源名称 (ARN) 或 `RoleArn` 必须有权访问模板使用的 S3 对象。要了解如何配置角色或用户，请参阅启用工作人员任务模板预览 (p. 2666)。

要使用 **`RenderUiTemplate`** 操作预览工作人员任务模板，请执行以下操作：

1. 提供附加了所需策略的角色的 **`RoleArn`** 以预览自定义模板。
2. 在**`Input`**的参数**`Task`**中，提供一个 JSON 对象，其中包含模板中定义的变量的值。这些变量用于替换 `task.input.source` 变量。例如，如果您在模板中定义了 task.input.text 变量，则可以在 JSON 对象中以下格式提供变量。`text：sample text.`
3. 在**`Content`**的参数**`UiTemplate`**中，插入模板。

配置完毕后`RenderUiTemplate`，使用您首选的 SDK 或Amazon CLI以提交呈现模板的请求。如果您的请求成功，响应包括：`RenderedContent`，为工作人员 UI 呈现 HTML 的 Liquid 模板。

> Important
>
> 要预览模板，您需要有权读取在用户界面上呈现的 Amazon S3 对象的 IAM 角色。有关可附加到 IAM 角色以授予这些权限的示例策略，请参阅启用工作人员任务模板预览 (p. 2666).

创建良好的工作人员说明

为人工审核作业创建良好的说明可提高工作人员完成任务的准确性。您可以在创建人工审核工作流程时修改控制台中提供的默认说明，也可以使用控制台创建自定义工作人员模板并在此模板中包含您的说明。这些说明将显示在工作人员在其中完成标记任务的 UI 页面上。

创建良好的工作人员说明

Amazon Astructions (Augmented AI) 控制台中有三种说明：

- 任务描述— 描述应提供任务的简要说明。
- 说明— 这些说明显示在工作人员完成任务的同一网页上。这些说明应提供简单的参考内容，向工作人员显示完成任务的正确方法。

- 附加说明— 这些说明显示在工作人员选择时出现的对话框中查看完整说明. 我们建议您提供完成任务的详细说明，并附上多个示例来显示边缘情况和其他难以标记对象的情况。

将示例图像添加到您的说明

图像为您的工作人员提供了有用的示例。要将可公开访问的图像添加到您的说明，请执行以下操作：

1. 将光标放在说明编辑器中的所需图像位置。
2. 选择编辑器工具栏中的图像图标。
3. 输入图像的 URL。

如果您的说明图像位于不可公开访问的 S3 存储桶中，请执行以下操作：

- 对于图像 URL，请输入：`{{ 'https://s3.amazonaws.com/your-bucket-name/image-file-name' | grant_read_access }}`。

这会提供一个附加了短期一次性访问代码的图像 URL，以便工作人员的浏览器可以显示该图像。将在说明编辑器中显示损坏的图像图标，但在预览该工具时，将在提供的预览中显示该图像。有关 `grand_read_access` 元素的更多信息，请参阅 grant_read_access (p. 2651)。

监控和管理您的人工循环

启动人工审核循环后，您可以检查发送到循环中的任务结果并使用Amazon Augmented AI 运行时 API. 此外，Amazon A2I 与 Amazon EventBridge（也称为 Amazon CloudWatch Events）集成，以便在人工审核循环状态更改为`Completed`、`Failed`，或者`Stopped`. 此事件交付至少有一次保证，这意味着在人类循环结束后创建的所有事件都会成功传送到 EventBridge。

使用以下过程可了解如何使用 Amazon A2I 运行时 API 监控和管理您的人工循环。请参阅使用Amazon CloudWatch Events在 Amazon Augmented AI (p. 2667)了解亚马逊 A2I 如何与 Amazon EventBridge 集成。

检查输出数据：

1. 通过调用 `DescribeHumanLoop` 操作来检查人工循环的结果。此 API 操作的结果包含有关循环激活的原因和结果的信息。
2. 在 Amazon Simple Storage Service (Amazon S3) 中检查人工循环中的输出数据。在数据的路径中，*YYYY*/*MM*/*DD*/*hh*/*mm*/*ss*用年代表人类循环创建日期（`YYYY`）、月 (`MM`) 和天 (`DD`)，以及创建时间（以小时为单位）`hh`）、分钟 (`mm`)，第二个（`ss`)。

```
s3://customer-output-bucket-specified-in-flow-definition/flow-definition-name/YYYY/MM/DD/hh/mm/ss/human-loop-name/output.json
```

您可以将此结构与Amazon Glue或 Amazon Athena 对输出数据进行分区和分析。有关更多信息，请参阅 。管理 中用于 ETL 输出的分区AmazonGlue.

要了解有关 Amazon A2I 输出数据格式的更多信息，请参阅Amazon A2I 输出数据 (p. 2654).

停止和删除您的人工循环：

1. 一旦人类循环启动，你可以通过调用`StopHumanLoop`操作使用`HumanLoopName`. 如果人工循环已成功停止，则服务器将发送回 HTTP 200 响应。
2. 要删除状态等于 `Failed`、`Completed` 或 `Stopped` 的人工循环，请使用 `DeleteHumanLoop` 操作。

列出人工循环：

1. 可通过调用 ListHumanLoops 操作来列出所有活动的人工循环。可以使用 CreationTimeAfter 和 CreateTimeBefore 参数按循环的创建日期筛选人工循环。
2. 如果成功，ListHumanLoops回报HumanLoopSummaries和NextToken响应元素中的对象。HumanLoopSummaries包含有关单个人类循环的信息。例如，列表s循环的状态和、如果适用，它的故障原因。

   将 NextToken 中返回的字符串用作对 ListHumanLoops 的后续调用中的输入以查看下一页人工循环。

Amazon A2I 输出数据

当您的机器学习工作流程向 Amazon A2I 发送数据对象时，人类循环已创建并且人类审阅者会收到任务来查看该数据对象。每个人工审查任务的输出数据都存储在您在人工审核工作流程中指定的 Amazon Simple Storage Service (Amazon S3) 输出存储桶中。在数据的路径中，*YYYY*/*MM*/*DD*/*hh*/*mm*/*ss*用年代表人类循环创建日期（YYYY)、月 (MM），和天（DD），以及创建时间（以小时为单位）hh)、分钟 (mm），第二个（ss)。

```
s3://customer-output-bucket-specified-in-flow-definition/flow-definition-name/YYYY/MM/DD/hh/mm/ss/human-loop-name/output.json
```

输出数据的内容取决于任务类型（内置或自定义）和类型劳动力您使用。您的输出数据始终包括人工的响应。此外，输出数据可能包括有关人类循环、人类审阅者（工作人员）和数据对象的元数据。

有关 Amazon A2I 输出数据格式的更多信息，请参阅以下部分，以了解不同任务类型和人力的更多信息。

从内置任务类型输出数据

亚马逊 A2I 内置任务类型包括 Amazon Textract 和 Amazon Rekognition。除了人为反应之外，其中一项任务的输出数据还包括有关创建人类循环原因的详细信息以及用于创建人类循环的综合服务的详细信息。有关所有内置任务类型的输出数据架构的更多信息，请参阅下表。这些区域有：值对于这些参数中的每个参数，都取决于您在 Amazon A2I 上使用的服务。有关这些服务特定值的详细信息，请参阅本节中的第二个表格。

参数	值类型	示例值	描述
awsManagedHumanLoopRequestSource	字符串	AWS/Rekognition/DetectModerationLabels/Image/V3 或者 AWS/Textract/AnalyzeDocument/Forms/V1	API 操作和关联Amazon要求亚马逊 A2I 创建人类循环的服务。这是您用来配置 Amazon A2I 人类循环的 API 操作。
flowDefinitionArn	字符串	arn:aws:sagemaker:us-west-2:*111122223333*:工作流定义/*flow-definition-name*	用于创建人工循环的人工审核工作流程（流程定义）的 Amazon 资源编号 (ARN)。
humanAnswers	JSON 对象列表	{ "answerContent": { "AWS/Rekognition/DetectModerationLabels/Image/V3": {	包含工作人员响应的 JSON 对象列表answerContent. 此对象还包含提交详细信息以及（如果使用了私人劳动力）的员工

参数	值类型	示例值	描述
		<pre>"moderationLabels": [...] } },</pre> 或 <pre>{ "answerContent": { "AWS/Textract/AnalyzeDocument/Forms/V1": { "blocks": [...] } },</pre>	元数据。要了解更多信息，请参阅 跟踪员工活动 (p. 2661)。 对于来自 Amazon Rekognition 生成的人类循环输出数据`DetectModerationLabel`审阅任务，此参数仅包含积极回应。例如，如果工作人员选择没有内容，此回复不包括在内。
`humanLoopName`	字符串	`'human-loop-name'`	人工循环的名称。
`inputContent`	JSON 对象	<pre>{ "aiServiceRequest": {...}, "aiServiceResponse": {...}, "humanTaskActivationConditionResults": {...}, "selectedAiServiceResponse": {...} }</pre>	输入内容Amazon服务在亚马逊 A2I 请求创建人类循环时发送给该服务。
`aiServiceRequest`	JSON 对象	<pre>{ "document": {...}, "featureTypes": [...], "humanLoopConfig": {...} }</pre> 或 <pre>{ "image": {...}, "humanLoopConfig": {...} }</pre>	发送到的原始请求Amazon与 Amazon A2I 集成的服务。例如，如果您将 Amazon Rekognition 与亚马逊 A2I 结合使用，则包括通过 API 操作发出的请求`DetectModerationLabels`. 对于 Amazon Textract 集成，这包括通过`AnalyzeDocument`.

参数	值类型	示例值	描述

aiServiceResponse	JSON 对象	{
		"moderationLabels": [...],
		"moderationModelVersion": "3.0"
		}
		或
		{
		"blocks": [...],
		"documentMetadata": {}
		}
			来自Amazon服务。这是用来确定是否需要人工审查的数据。此对象可能包含未与人工审阅者共享的有关数据对象的元数据。

selectedAiServiceResponse	JSON对象	{
		"moderationLabels": [...],
		"moderationModelVersion": "3.0"
		}
		或
		{
		"blocks": [...],
		"documentMetadata": {}
		}
			The subset of the aiServiceResponse匹配中的激活条件ActivationConditions.
			中列出的所有数据对象aiServiceResponse列于selectedAiServiceResponse当推论被随机抽样时，或者所有推论都启动激活条件。

humanTaskActivationConditionsResults	JSON对象	{
		"Conditions": [...]
		}
			中的 JSON 对象inputContent这包含了创造人类循环的原因。这包括激活条件的列表（Conditions) 包含在人工审核工作流程（流程定义）中，以及每种情况的评估结果-此结果为true要么false.要了解激活条件的更多信息，请参阅亚马逊中用于人 Augmented AI 循环激活条件的 JSON 架构 (p. 2628).

在下表中选择一个选项卡以了解特定于任务类型的参数，并查看每种内置任务类型的示例输出数据代码块。

Amazon Textract Task Type Output Data

当您使用 Amazon Textract 内置集成时，您会看到`'AWS/Textract/AnalyzeDocument/Forms/V1'`作为的价值`awsManagedHumanLoopRequestSource`输出数据中。

这些区域有：`answerContent`参数包含`Block`对象，其中包括发送给 Amazon A2I 的所有区块的人类响应。

这些区域有：`aiServiceResponse`参数还包括`Block`对象，其中包含亚马逊 Textract 对原始请求的响应`AnalyzeDocument`.

要了解有关在块对象中看到的参数的更多信息，请参阅Block中的Amazon Textract 开发人员指南.

以下是亚马逊 A2I 人工评论对 Amazon Textract 文档分析推断的输出数据的示例。

```
{
    "awsManagedHumanLoopRequestSource": "AWS/Textract/AnalyzeDocument/Forms/V1",
    "flowDefinitionArn": "arn:aws:sagemaker:us-west-2:111122223333:flow-
definition/flow-definition-name",
    "humanAnswers": [
        {
            "answerContent": {
                "AWS/Textract/AnalyzeDocument/Forms/V1": {
                    "blocks": [...]
                }
            },
            "submissionTime": "2020-09-28T19:17:59.880Z",
            "workerId": "111122223333",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Cognito",
                    "issuer": "https://cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111",
                    "sub": "c6aa8eb7-9944-42e9-a6b9-111122223333"
                }
            }
        }
    ],
    "humanLoopName": "humnan-loop-name",
    "inputContent": {
        "aiServiceRequest": {
            "document": {
                "s3Object": {
                    "bucket": "DOC-EXAMPLE-BUCKET1",
                    "name": "document-demo.jpg"
                }
            },
            "featureTypes": [
                "TABLES",
                "FORMS"
            ],
            "humanLoopConfig": {
                "dataAttributes": {
                    "contentClassifiers": [
                        "FreeOfPersonallyIdentifiableInformation"
                    ]
                },
                "flowDefinitionArn": "arn:aws:sagemaker:us-west-2:111122223333:flow-
definition/flow-definition-name",
                "humanLoopName": "humnan-loop-name"
            }
        },
        "aiServiceResponse": {
            "blocks": [...],
            "documentMetadata": {
```

```
                "pages": 1
            }
        },
        "humanTaskActivationConditionResults": {
            "Conditions": [
                {
                    "EvaluationResult": true,
                    "Or": [
                        {
                            "ConditionParameters": {
                                "ImportantFormKey": "Mail address",
                                "ImportantFormKeyAliases": [
                                    "Mail Address:",
                                    "Mail address:",
                                    "Mailing Add:",
                                    "Mailing Addresses"
                                ],
                                "KeyValueBlockConfidenceLessThan": 100,
                                "WordBlockConfidenceLessThan": 100
                            },
                            "ConditionType": "ImportantFormKeyConfidenceCheck",
                            "EvaluationResult": true
                        },
                        {
                            "ConditionParameters": {
                                "ImportantFormKey": "Mail address",
                                "ImportantFormKeyAliases": [
                                    "Mail Address:",
                                    "Mail address:",
                                    "Mailing Add:",
                                    "Mailing Addresses"
                                ]
                            },
                            "ConditionType": "MissingImportantFormKey",
                            "EvaluationResult": false
                        }
                    ]
                }
            ]
        },
        "selectedAiServiceResponse": {
            "blocks": [...]
        }
    }
}
```

Amazon Rekognition Task Type Output Data

当您使用 Amazon Textract 内置集成时，您会看到字符串`'AWS/Rekognition/DetectModerationLabels/Image/V3'`作为的价值`awsManagedHumanLoopRequestSource`输出数据中。

这些区域有：`answerContent`参数包含`moderationLabels`对象，其中包含针对发送至 Amazon A2I 的所有审核标签的人类响应。

这些区域有：`aiServiceResponse`参数还包括`moderationLabels`反映亚马逊 Rekognition 对发送至的原始请求的响应`DetectModerationLabels`.

要了解有关在块对象中看到的参数的更多信息，请参阅ModerationLabel在 Amazon Rekognition 开发人员指南中。

以下是亚马逊 A2I 人类评论对 Amazon Rekognition 图像审核推断的输出数据的示例。

```
{
```

```
    "awsManagedHumanLoopRequestSource": "AWS/Rekognition/DetectModerationLabels/Image/
V3",
    "flowDefinitionArn": "arn:aws:sagemaker:us-west-2:111122223333:flow-
definition/flow-definition-name",
    "humanAnswers": [
        {
            "answerContent": {
                "AWS/Rekognition/DetectModerationLabels/Image/V3": {
                    "moderationLabels": [...]
                }
            },
            "submissionTime": "2020-09-28T19:22:35.508Z",
            "workerId": "ef7294f850a3d9d1",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Cognito",
                    "issuer": "https://cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111",
                    "sub": "c6aa8eb7-9944-42e9-a6b9-111122223333"
                }
            }
        }
    ],
    "humanLoopName": "humnan-loop-name",
    "inputContent": {
        "aiServiceRequest": {
            "humanLoopConfig": {
                "flowDefinitionArn": "arn:aws:sagemaker:us-west-2:111122223333:flow-
definition/flow-definition-name",
                "humanLoopName": "humnan-loop-name"
            },
            "image": {
                "s3Object": {
                    "bucket": "DOC-EXAMPLE-BUCKET1",
                    "name": "example-image.jpg"
                }
            }
        },
        "aiServiceResponse": {
            "moderationLabels": [...],
            "moderationModelVersion": "3.0"
        },
        "humanTaskActivationConditionResults": {
            "Conditions": [
                {
                    "EvaluationResult": true,
                    "Or": [
                        {
                            "ConditionParameters": {
                                "ConfidenceLessThan": 98,
                                "ModerationLabelName": "Suggestive"
                            },
                            "ConditionType": "ModerationLabelConfidenceCheck",
                            "EvaluationResult": true
                        },
                        {
                            "ConditionParameters": {
                                "ConfidenceGreaterThan": 98,
                                "ModerationLabelName": "Female Swimwear Or Underwear"
                            },
                            "ConditionType": "ModerationLabelConfidenceCheck",
                            "EvaluationResult": false
                        }
                    ]
                }
            ]
```

```
        },
        "selectedAiServiceResponse": {
            "moderationLabels": [
                {
                    "confidence": 96.7122802734375,
                    "name": "Suggestive",
                    "parentName": ""
                }
            ],
            "moderationModelVersion": "3.0"
        }
    }
}
```

输出自定义任务类型的数据

当您将 Amazon A2I 添加到自定义人工审核工作流程时，您会在人工审核任务返回的输出数据中看到以下参数。

参数	值类型	描述
`flowDefinitionArn`	字符串	用于创建人工循环的人工审核工作流程（流程定义）的 Amazon 资源编号 (ARN)。
`humanAnswers`	JSON 对象列表	包含工作人员响应的 JSON 对象列表`answerContent`. 此参数中的值取决于从您的Worker 任务模板. 如果您使用的是私人劳动力，则包括员工元数据。要了解更多信息，请参阅 跟踪员工活动 (p. 2661)。
`humanLoopName`	字符串	人工循环的名称。
`inputContent`	JSON 对象	请求中发送给亚马逊 A2I 的输入内容`StartHumanLoop`.

以下是与 Amazon A2I 和 Amazon Transcribe 的自定义集成的输出数据的示例。在此示例中，`inputContent`包含以下内容：

- Amazon S3 中 .mp4 文件的路径和视频标题
- 从亚马逊 Transcribe 返回的转录（从 Amazon Transcribe 输出数据中解析）
- 工作人员任务模板用于裁剪 .mp4 文件并向工作人员显示视频的相关部分的开始和结束时间

```
{
    "flowDefinitionArn": "arn:aws:sagemaker:us-west-2:111122223333:flow-definition/flow-definition-name",
    "humanAnswers": [
        {
            "answerContent": {
                "transcription": "use lambda to turn your notebook"
            },
            "submissionTime": "2020-06-18T17:08:26.246Z",
```

```
            "workerId": "ef7294f850a3d9d1",
            "workerMetadata": {
                "identityData": {
                    "identityProviderType": "Cognito",
                    "issuer": "https://cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111",
                    "sub": "c6aa8eb7-9944-42e9-a6b9-111122223333"
                }
            }

        }
    ],
    "humanLoopName": "human-loop-name",
    "inputContent": {
        "audioPath": "s3://DOC-EXAMPLE-BUCKET1/a2i_transcribe_demo/Fully-Managed Notebook
 Instances with Amazon SageMaker - a Deep Dive.mp4",
        "end_time": 950.27,
        "original_words": "but definitely use Lambda to turn your ",
        "start_time": 948.51,
        "video_title": "Fully-Managed Notebook Instances with Amazon SageMaker - a Deep
 Dive.mp4"
    }
}
```

跟踪员工活动

Amazon A2I 提供的信息可用于跟踪任务输出数据中的个人员。要识别执行人工审查任务的工作人员，请使用 Amazon S3 中输出数据中的以下内容：

- 这些区域有：`acceptanceTime`是工作人员接受任务的时间。此日期和时间戳的格式为`YYYY-MM-DDTHH:MM:SS.mmmZ`为年份 (`YYYY`)、月 (`MM`)、天 (`DD`)、小时 (`HH`)、分钟 (`MM`), 第二 (`SS`) ，和毫秒 (`mmm`)。日期和时间由T.
- 这些区域有：`submissionTime`是工作人员使用提交按钮。此日期和时间戳的格式为`YYYY-MM-DDTHH:MM:SS.mmmZ`为年份 (`YYYY`)、月 (`MM`)、天 (`DD`)、小时 (`HH`)、分钟 (`MM`), 第二 (`SS`) ，和毫秒 (`mmm`)。日期和时间由T.
- `timeSpentInSeconds`报告工作人员积极处理该任务的总时间（以秒为单位）。此指标不包括工作人员暂停或休息的时间。
- `workerId` 对于每个工作人员都是唯一的。
- 如果您使用私有人力，在`workerMetadata`，您将看到以下内容。
  - 这些区域有：`identityProviderType`是用于管理私有人力的服务。
  - 这些区域有：`issuer`与分配给此人工审核任务的工作团队关联的 Amazon Cognito 用户池或 OpenID Connect (OIDC) 身份提供商 (IdP) 发行人工。
  - 唯一`sub`标识符指工作人员。如果您使用 Amazon Cognito 创建人力，您可以使用 Amazon Cognito 检索与此 ID 关联的此工作人员的详细信息（如姓名或用户名）。要了解如何操作，请参阅管理和搜索用户账户在Amazon Cognito 开发人员指南.

以下是您在使用 Amazon Cognito 创建私有人力时可能看到的输出示例。这在`identityProviderType`.

```
"submissionTime": "2020-12-28T18:59:58.321Z",
"acceptanceTime": "2020-12-28T18:59:15.191Z",
"timeSpentInSeconds": 40.543,
"workerId": "a12b3cdefg4h5i67",
"workerMetadata": {
    "identityData": {
        "identityProviderType": "Cognito",
        "issuer": "https://cognito-idp.aws-region.amazonaws.com/aws-region_123456789",
        "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
    }
```

```
}
```

以下是如果您使用自己的 OIDC IdP 来创建私人劳动力，您可能会看到的输出示例：

```
"workerMetadata": {
        "identityData": {
            "identityProviderType": "Oidc",
            "issuer": "https://example-oidc-ipd.com/adfs",
            "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
        }
}
```

要了解有关使用私有人力的更多信息，请参阅使用私有人力 (p. 467).

Amazon Augmented AI 中的权限和安全性

当您使用亚马逊 Augmented AI (Amazon A2I) 为 ML/AI 应用程序创建人工审核工作流时，您创建和配置。资源在 Amazon SageMaker 中，例如人力劳动力和员工任务模板。要配置和启动人工循环，您可以将 Amazon A2I 与其他人工循环集成。Amazon诸如 Amazon Textract 或 Amazon Rekognition 之类的服务，或者使用亚马逊 Augmented AI 运行时 API。要创建人工审核工作流并启动人工循环，您必须将特定策略附加到您的 Amazon Identity and Access Management(IAM) 角色或用户。具体来说：

- 在 2020 年 1 月 12 日或之后，您使用图像输入数据启动人工循环时，您必须将一个 CORS 标头策略添加到包含输入数据的 Amazon S3 存储桶中。请参阅CORS 权限要求 (p. 2663)了解更多信息。
- 创建流定义时，您需要提供一个角色，授予 Amazon A2I 权限以访问 Amazon S3，无论是读取在人工任务 UI 中呈现的对象，还是写入人工审核的结果。

  该角色还必须附加信任策略，以向 SageMaker 授予代入角色的权限。这使 Amazon A2I 可以根据您附加到角色的权限执行操作。

  有关您可修改并附加到用于创建流定义的角色的示例策略，请参阅向用于创建流定义的 IAM 角色添加权限 (p. 2663)。这些是附加到在人工审核工作流SageMaker 控制台的亚马逊 A2I 区域的部分。
- 要创建和启动人工循环，您可以从内置任务类型中使用 API 操作（如`DetectModerationLabel`要么`AnalyzeDocument`) 或者 Amazon A2I 运行时 API 操作`StartHumanLoop`在自定义 ML 应用程序中。您需要附加`AmazonAugmentedAIFullAccess`针对调用这些 API 操作以向这些服务授予权限以使用 Amazon A2I 操作的 IAM 用户的托管策略。要了解如何操作，请参阅创建可以调用 Amazon A2I API 操作的 IAM 用户 (p. 2664)。

  该策略不会 授予调用与内置任务类型关联的 Amazon 服务的 API 操作的权限。例如，`AmazonAugmentedAIFullAccess`未授予调用 Amazon Rekognition 的权限`DetectModerationLabel`API 操作或 Amazon Textract`AnalyzeDocument`API 操作。您可以使用更一般的策略，`AmazonAugmentedAIIntegratedAPIAccess`，以授予这些权限。有关更多信息，请参阅 创建具有调用 Amazon A2I、Amazon Textract 和 Amazon Rekognition API 操作权限的 IAM 用户 (p. 2665)。如果您希望向 IAM 用户授予广泛的权限以使用 Amazon A2I 和集成的权限，这是一个不错的选择。Amazon服务的 API 操作。

  如果要配置更精细的权限，请参阅 Amazon Rekognition 基于身份的策略示例和 Amazon Textract 基于身份的策略示例，以了解可用于授权使用这些单独服务的基于身份的策略。
- 要预览自定义工作人员任务 UI 模板，您需要有权读取在用户界面上呈现的 Amazon S3 对象的 IAM 角色。请参阅启用工作人员任务模板预览 (p. 2666)中的策略示例。

主题

- CORS 权限要求 (p. 2663)
- 向用于创建流定义的 IAM 角色添加权限 (p. 2663)

- 创建可以调用 Amazon A2I API 操作的 IAM 用户 (p. 2664)
- 创建具有调用 Amazon A2I、Amazon Textract 和 Amazon Rekognition API 操作权限的 IAM 用户 (p. 2665)
- 启用工作人员任务模板预览 (p. 2666)
- 将 Amazon A2I 与Amazon KMS加密存储桶 (p. 2666)
- 其他权限和安全资源 (p. 2667)

CORS 权限要求

2020 年早些时候，Chrome 和 Firefox 等广泛使用的浏览器改变了基于图像元数据旋转图像的默认行为，称为EXIF 数据. 以前，图像将始终在浏览器中显示它们的准确方式存储在磁盘上，磁盘上的存储方式通常不会旋转。更改之后，图像现在会根据名为的图像元数据旋转方向值. 这对整个机器学习 (ML) 社区具有重要影响。例如，如果不考虑 EXIF 方向，则用于注释图像的应用程序可能会以意想不到的方向显示图像，并导致标签不正确。

从 Chrome 89 开始，Amazon无法再自动阻止图像的旋转，因为 Web 标准组 W3C 决定控制图像轮换的能力违反了网络的同来源政策。因此，为了确保人工在提交创建人类循环的请求时以可预测的方向对输入图像进行注释，您必须向包含输入图像的 S3 存储桶添加 CORS 标头策略。

> Important
>
> 如果您没有向包含输入数据的 S3 存储桶添加 CORS 配置，那么这些输入数据对象的人工审查任务将失败。

您可以将 CORS 策略添加到 Amazon S3 控制台中包含输入数据的 S3 存储桶。要在 S3 控制台中包含输入图像的 S3 存储桶上设置所需的 CORS 标头，请按照中详细的说明进行操作如何通过 CORS 添加跨域资源共享？. 对托管您的映像的存储桶使用以下 CORS 配置代码。如果您使用 Amazon S3 控制台向存储桶添加策略，则必须使用 JSON 格式。

JSON

```
[{
    "AllowedHeaders": [],
    "AllowedMethods": ["GET"],
    "AllowedOrigins": ["*"],
    "ExposeHeaders": []
}]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>GET</AllowedMethod>
 </CORSRule>
</CORSConfiguration>
```

向用于创建流定义的 IAM 角色添加权限

要创建流定义，请将本部分中的策略附加到您在 SageMaker 控制台中或在使用`CreateFlowDefinition`API 操作。

- 如果您使用控制台创建人工审核工作流程，请在IAM 角色字段何时在控制台中创建人工审核工作流程.
- 使用 API 创建流定义时，请将这些策略附加到传递给 `CreateFlowDefinition` 操作的 `RoleArn` 参数的角色。

创建人工审核工作流（流定义）时，Amazon A2I 调用 Amazon S3 来完成您的任务。要向 Amazon A2I 授予权限以检索和存储在 Amazon S3 存储桶中的权限，请创建以下策略并将其附加到您的角色。例如，如果您发送用于进行人工审核的图像、文档和其他文件存储在名为的 S3 存储桶中。`my_input_bucket`，如果你想将人类评论存储在名为的存储桶中`my_output_bucket`中，请创建以下策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::my_input_bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::my_output_bucket/*"
            ]
        }
    ]
}
```

此外，IAM 角色必须具有以下信任策略，以向 SageMaker 授予代入角色的权限。要了解有关 IAM 信任策略的更多信息，请参基于资源的策略部分策略和权限中的AmazonIdentity and Access Management文档中）。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSageMakerToAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

有关创建和管理 IAM 角色和策略的更多信息，请参阅中的以下主题：Amazon Identity and Access Management用户指南：

- 要创建 IAM 角色，请参阅创建向 IAM 用户委派权限的角色.
- 要了解如何创建 IAM 策略，请参阅创建 IAM 策略.
- 要了解如何将 IAM 策略附加到角色，请参阅添加和删除 IAM 身份权限.

创建可以调用 Amazon A2I API 操作的 IAM 用户

要使用 Amazon A2I 为 Amazon Rekognition、Amazon Amazon Textract 或 Amazon A2I 运行时 API 创建和启动人工循环，您必须使用有权调用 Amazon A2I 操作的 IAM 用户。要执行此操作，请使用 IAM 控制台附加`AmazonAugmentedAIFullAccess`针对新的或现有 IAM 用户的托管策略。

该策略向 IAM 用户授予以下权限：从 SageMaker API 中调用 API 操作以创建和管理流定义，以及从 Amazon Advanced Augmented AI 运行时 API 中调用 API 操作以创建和管理人工循环。要了解这些 API 操作的更多信息，请参阅在 Amazon Augmented AI 中使用 API。

`AmazonAugmentedAIFullAccess`不会授予使用 Amazon Rekognition 或 Amazon Textract API 操作的权限。

Note

您也可以附加`AmazonAugmentedAIFullAccess`用于创建和启动人工循环的 IAM 角色的策略。

创建所需的 IAM 用户

1. 通过以下网址登录到 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择用户并选择现有用户，或者选择添加用户来创建新用户。要了解如何创建新用户，请参阅在您的中创建 IAM 用户Amazon账户中的Amazon Identity and Access Management用户指南.
   - 如果您选择将策略附加到现有用户，请选择添加权限。
   - 在创建新用户时，请执行设置权限页面上的下一步。
3. 选择 Attach existing policies directly（直接附上现有策略）。
4. 在搜索栏，请输入`AmazonAugmentedAIFullAccess`然后，选中该策略旁边的框。

   要允许此 IAM 用户能够与公有工作团队一起创建流定义，还需要附加`AmazonSageMakerMechanicalTurkAccess`托管策略。
5. 附加一个或多个策略后：
   a. 如果您使用现有用户，请选择后续：审核，然后选择添加权限.
   b. 如果您要创建新用户，请选择后续：标签然后，完成创建用户的过程。

有关更多信息，请参阅 。添加和删除 IAM 身份权限中的Amazon Identity and Access Management用户指南.

创建具有调用 Amazon A2I、Amazon Textract 和 Amazon Rekognition API 操作权限的 IAM 用户

要创建有权调用内置任务类型使用的 API 操作的 IAM 用户（即，`DetectModerationLables`适用于 Amazon Rekognition 和`AnalyzeDocument`对于 Amazon Textract）以及使用所有 Amazon A2I API 操作的权限，请附加 IAM 托管策略，`AmazonAugmentedAIIntegratedAPIAccess`. 如果您希望向使用具有多种任务类型的 Amazon A2I 用户授予广泛的权限，您可能需要使用此策略。要了解这些 API 操作的更多信息，请参阅在 Amazon Augmented AI 中使用 API。

Note

您也可以附加`AmazonAugmentedAIIntegratedAPIAccess`用于创建和启动人工循环的 IAM 角色的策略。

创建所需的 IAM 用户

1. 通过以下网址登录到 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择用户并选择现有用户，或者选择添加用户来创建新用户。要了解如何创建新用户，请参阅在您的中创建 IAM 用户Amazon账户中的Amazon Identity and Access Management用户指南.
   - 如果您选择将策略附加到现有用户，请选择添加权限。
   - 如果您要创建新用户，请按照设置权限页.
3. 选择 Attach existing policies directly（直接附上现有策略）。

4. 在搜索栏，请输入`AmazonAugmentedAIIntegratedAPIAccess`然后选中该策略旁边的框。

   要允许该 IAM 用户使用 Amazon Mechanical Turk 创建流定义，还需要附加`AmazonSageMakerMechanicalTurkAccess`托管策略。
5. 附加一个或多个策略后：

   a. 如果您使用现有用户，请选择后续：审核，然后选择添加权限.

   b. 如果您要创建新用户，请选择后续：标签然后，完成创建用户的过程。

有关更多信息，请参阅 。添加和删除 IAM 身份权限中的Amazon Identity and Access Management用户指南.

启用工作人员任务模板预览

要自定义工作人员在处理任务时看到的界面和说明，您可以创建工作人员任务模板。您可以使用`CreateHumanTaskUi`操作或 SageMaker 控制台。

要预览模板，您需要具有以下权限的 IAM 角色，以读取在用户界面上呈现的 Amazon S3 对象。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::my_input_bucket/*"
            ]
        }
    ]
}
```

对于 Amazon Rekognition 和 Amazon Textract 任务类型，您可以使用 SageMaker 控制台的亚马逊 Augmented AI 部分预览模板。对于自定义任务类型，您可以通过调用`RenderUiTemplate`operation. 要预览模板，请按照任务类型的说明操作：

- Amazon Rekognition 和 Amazon Textract 任务类型 — 在 SageMaker 控制台中，在中所记录的过程中使用角色的亚马逊资源名称 (ARN)。创建工作人员任务模板 (p. 2647).
- 自定义任务类型 — 在`RenderUiTemplate`操作，请 ARN`RoleArn`参数。

将 Amazon A2I 与Amazon KMS加密存储桶

如果你指定Amazon Key Management Service(Amazon KMS) 用于加密输出数据的客户管理密钥`OutputConfig`的`CreateFlowDefinition`，您必须向该密钥添加类似于以下内容的 IAM 策略。此策略授予您用于创建人类循环的 IAM 执行角色的权限，以便使用此密钥执行中列出的所有操作`"Action"`. 要了解这些操作的更多信息，请参阅Amazon KMS许可中的Amazon Key Management Service开发人员指南 的第一个版本。

要使用此策略，请在中替换 IAM 服务角色 ARN`"Principal"`使用您用于创建人工审核工作流（流定义）的执行角色的 ARN。使用创建标记作业时`CreateFlowDefinition`，这是你指定的 ARN`RoleArn`. 请注意，你不能提供`KmsKeyId`在控制台中创建流定义时。

```
{
    "Sid": "AllowUseOfKmsKey",
```

```
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/service-role/example-role"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

其他权限和安全资源

- the section called “控制对 的访问 SageMaker 使用标签资源” (p. 2715).
- the section called “SageMaker 基于身份的策略” (p. 2694)
- the section called “的控制创建 SageMaker 使用条件键的资源” (p. 2706)
- the section called “亚马逊 SageMaker API 权限参考” (p. 2739)
- 安全性 (p. 2684)

使用Amazon CloudWatch Events在 Amazon Augmented AI

当人工审核循环状态更改为时，Amazon Augmented AI 使用 Amazon CloudWatch Events 提醒您。`Completed`、`Failed`，或者`Stopped`. 此事件交付至少有一次保证，这意味着在人类循环结束后创建的所有事件都将成功传送到 CloudWatch Events (Amazon EventBridge)。当审核循环更改为其中一种状态时，Augmented AI 会向 CloudWatch Events 发送类似于以下内容的事件。

```
{
    "version":"0",
    "id":"12345678-1111-2222-3333-12345EXAMPLE",
    "detail-type":"SageMaker A2I HumanLoop Status Change",
    "source":"aws.sagemaker",
    "account":"1111111111111",
    "time":"2019-11-14T17:49:25Z",
    "region":"us-east-1",
    "resources":["arn:aws:sagemaker:us-east-1:111111111111:human-loop/humanloop-nov-14-1"],
    "detail":{
        "creationTime":"2019-11-14T17:37:36.740Z",
        "failureCode":null,
        "failureReason":null,
        "flowDefinitionArn":"arn:aws:sagemaker:us-east-1:111111111111:flow-definition/
flowdef-nov-12",
        "humanLoopArn":"arn:aws:sagemaker:us-east-1:111111111111:human-loop/humanloop-
nov-14-1",
        "humanLoopName":"humanloop-nov-14-1",
        "humanLoopOutput":{
            "outputS3Uri":"s3://customer-output-bucket-specified-in-flow-definition/
flowdef-nov-12/2019/11/14/17/37/36/humanloop-nov-14-1/output.json"
        },
        "humanLoopStatus":"Completed"
    }
}
```

JSON 输出中的详细信息包括以下内容：

`creationTime`

Augmented AI 创建人工循环时的时间戳。

`failureCode`

表示特定失败类型的失败代码。

`failureReason`

人工循环失败的原因。仅当人工审核循环状态为 `failed` 时，才会返回失败原因。

`flowDefinitionArn`

流程定义的 Amazon 资源名称 (ARN) 或人工审核工作流程。

`humanLoopArn`

人工循环的 Amazon 资源名称 (ARN)。

`humanLoopName`

人工循环的名称。

`humanLoopOutput`

包含人工循环输出相关信息的对象。

`outputS3Uri`

Augmented AI 存储人工循环输出的 Amazon S3 对象的位置。

`humanLoopStatus`

人工循环的状态。

将事件从人工循环发送到 CloudWatch Events

要配置 CloudWatch Events 规则以获取状态更新，或者事件，对于您的 Amazon A2I 人类循环，请使用 Amazon Command Line Interface(Amazon CLI) `put-rule`命令。在使用 `put-rule` 命令时，请指定以下内容来接收人工循环状态：

- `\"source\":[\"aws.sagemaker\"]`
- `\"detail-type\":[\"SageMaker A2I HumanLoop Status Change\"]`

要配置 CloudWatch Events 规则以监控所有状态更改，请使用以下命令并替换占位符文本。例如，替换*"A2IHumanLoopStatusChanges"*使用唯一的 CloudWatch Event 规则名称和*"arn:aws:iam::111122223333:role/MyRoleForThisRule"*附加了 events.amazonaws.com 策略的 IAM 角色的 Amazon 资源编号 (ARN)。Replace*##*使用Amazon要在其中创建规则的区域。

```
aws events put-rule --name "A2IHumanLoopStatusChanges"
    --event-pattern "{\"source\":[\"aws.sagemaker\"],\"detail-type\":[\"SageMaker A2I
 HumanLoop Status Change\"]}"
    --role-arn "arn:aws:iam::111122223333:role/MyRoleForThisRule"
    --region "region"
```

要了解有关的更多信息`put-rule`请求，请参阅CloudWatch Events 中的事件模式中的Amazon CloudWatch Events 用户指南.

设置目标以处理事件

要处理事件，您需要设置目标。例如，如果您希望在人工循环状态发生更改时接收电子邮件，请使用中的步骤设置 Amazon SNS 通知中的Amazon CloudWatch 用户指南设置 Amazon SNS 主题并为电子邮件订阅此主题。创建主题后，可以使用该主题创建目标。

向 CloudWatch Events 规则添加目标

1. 打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/home
2. 在导航窗格中，选择 Rules (规则)。
3. 选择要向其添加目标的规则。
4. 选择 Actions，然后选择 Edit。
5. 在 Targets (目标) 下，选择 Add Target (添加目标)，然后选择要在检测到人工循环状态更改事件时使用的 Amazon 服务。
6. 配置您的目标。有关说明，请参阅中有关配置目标的主题。Amazon该服务的文档.
7. 选择 Configure details (配置详细信息)。
8. 对于 Name (名称)，输入一个名称并（可选）在 Description (描述) 中提供有关规则用途的详细信息。
9. 请确保选中 State (状态) 旁边的复选框，以便您的规则以 Enabled (已启用) 状态列出。
10. 选择 Update rule (更新规则)。

使用人工审核输出

收到人工审核结果后，您可以分析结果并将其与机器学习预测进行比较。存储在 Amazon S3 存储桶中的 JSON 包含机器学习预测和人工审核结果。

更多信息

使用 Amazon EventBridge 自动执行 Amazon SageMaker (p. 2823)

在 Amazon Augmented AI 中使用 API

您可以通过编程方式创建人工审核工作流程或工作人员任务模板。您使用的 API 取决于您创建的是 Amazon Rekognition、Amazon Textract 还是自定义任务类型。本主题提供指向每个任务类型和编程任务的 API 参考文档的链接。

以下 API 可与 Augmented AI 一起使用：

Amazon Augmented AI

使用 Augmented AI API 启动、停止和删除人工审核循环。您还可以列出所有人工审核循环，并返回账户中有关人工审核循环的信息。

请参阅以下方面了解有关人工审核循环 APIAmazon Augmented AI 运行时 API 参考.

Amazon Rekognition

使用HumanLoopConfig的参数 `DetectModerationLabels`API 到启动使用 Amazon Rekognition 的人工审核工作流程。

Amazon SageMaker

使用 Amazon SageMaker API 创建`FlowDefinition`，也称作人工审核工作流. 您还可以创建`HumanTaskUi`要么Worker 任务模板.

有关更多信息，请参阅 `CreateFlowDefinition` 或 `CreateHumanTaskUi` API 文档。

Amazon Textract

使用HumanLoopConfig的参数AnalyzeDocument使用 Amazon Textract 启动人工审查工作流程的 API。

编程教程

以下教程提供以编程方式创建人工审核工作流程和工作人员任务模板的示例代码和分步说明。

- 教程：开始使用 Amazon A2I API (p. 2608)
- 创建人工审核工作流程 (API) (p. 2624)
- 创建和启动人工循环 (p. 2640)
- 将 Amazon Augmented AI 与 Amazon Rekognition 一起使用中的Amazon Rekognition 开发人员指南
- 将 Amazon Augmented AI 与 Amazon Textract 一起使用中的Amazon Textract 开发人员指南

在中购买和出售 Amazon SageMaker 算法和模型Amazon Web Services Marketplace

此功能在中国区域中不可用。

Amazon SageMaker 与Amazon Web Services Marketplace，使开发人员能够向使用算法和模型包的其他 SageMaker 用户收取费用。Amazon Web Services Marketplace是一个精挑细选的数字化产品目录，客户通过它可以轻松地查找、购买、部署和管理构建解决方案及运营其业务所需的第三方软件和服务。Amazon Web Services Marketplace包含各种热门类别的数千种软件产品，这些类别包括安全、联网、存储、机器学习、商业智能、数据库和 DevOps 等。它提供灵活的定价选项和多种部署方法，简化了软件许可和采购过程。

将自己的算法和模型与AmazonMarketplace

以下各节介绍了如何创建算法和模型包资源，这些资源可以在本地使用并发布到AmazonMarketplace。

主题

- 创建算法和模型包资源 (p. 2671)
- 使用算法和模型包资源 (p. 2676)

创建算法和模型包资源

在将训练和/或推理代码包到 Docker 容器中后，创建可在 Amazon SageMaker 账户中使用并（可选）在上发布的算法和模型包资源。Amazon Web Services Marketplace.

主题

- 创建算法资源 (p. 2671)
- 创建模型包资源 (p. 2674)

创建算法资源

为了创建可用于在 Amazon SageMaker 中运行训练作业并在上发布的算法资源Amazon Web Services Marketplace指定以下信息：

- 包含训练和（可选）推理代码的 Docker 容器。
- 您的算法预期用于训练的输入数据的配置。
- 您的算法支持的超参数。
- 您的算法在训练作业期间发送到 Amazon CloudWatch 的指标。
- 您的算法支持的用于训练和推理的实例类型，以及它是否支持跨多个实例的分布式训练。
- 验证配置文件，即 SageMaker 用来测试算法的训练代码的训练作业和用来测试算法的推理代码的批量转换作业。

为了确保买家和卖家都能确信产品适用于 SageMaker，我们要求您在将算法列入中之前对其进行验证。Amazon Web Services Marketplace. 您只能在验证成功后将产品列入 Amazon Web Services Marketplace。为了验证您的算法，SageMaker 使用您的验证配置文件和示例数据来运行以下验证任务：

1. 在您的账户中创建一个训练作业以验证您的训练镜像是否适用于 SageMaker。
2. 如果已在算法中包含推理代码，则使用算法的推理镜像以及训练作业所生成的模型构件在账户中创建模型。
3. 如果已在算法中包含推理代码，则使用该模型在您的账户中创建一个转换作业以验证您的推理镜像是否适用于 SageMaker。

当您在 Amazon Web Services Marketplace 上列出您的产品时，此验证过程的输入和输出将作为您的产品的一部分保留，并提供给您的买家。这有助于买家在购买产品前先了解和评估产品。例如，买家可以检查您使用的输入数据、生成的输出以及代码发出的日志和指标。验证规范越全面，客户就越容易评估您的产品。

> **Note**
>
> 在验证配置文件中，仅提供您希望公开的数据。

验证可能需要几个小时。要查看账户中作业的状态，请在 SageMaker 控制台中，查看训练任务和转换作业页面。如果验证失败，您可以从 SageMaker 控制台访问扫描和验证报告。如果发现任何问题，您必须重新创建算法。

> **Note**
>
> 要在 Amazon Web Services Marketplace 上发布您的算法，至少需要一个验证配置文件。

您可以使用 SageMaker 控制台或 SageMaker API 创建算法。

主题

- 创建算法资源（控制台）(p. 2672)
- 创建算法资源 (API) (p. 2674)

创建算法资源（控制台）

创建算法资源（控制台）

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 从左侧菜单中，选择训练.
3. 从下拉菜单中，选择算法，然后选择创建算法.
4. 在 Training specifications (训练规范) 页面上，提供以下信息：
   a. 对于 Algorithm name (算法名称)，键入算法的名称。算法名称在您的账户和 Amazon 区域中必须是唯一的。名称必须具有 1 到 64 个字符。有效字符为 a-z、A-Z、0-9 和 -（连字符）。
   b. 键入算法的描述。此描述将显示在 SageMaker 控制台和Amazon Web Services Marketplace.
   c. 适用于训练图片，在 Amazon ECR 中键入存储训练容器的路径。
   d. 对于 Support distributed training (支持分布式训练)，如果您的算法支持在多个实例上进行训练，则选择 Yes (是)。否则，请选择 No。
   e. 对于 Support instance types for training (支持训练的实例类型)，选择您的算法支持的实例类型。
   f. 对于 Channel specification (通道规范)，为算法指定最多 8 个输入数据通道。例如，您可以指定 3 个输入通道，它们分别名为 `train`、`validation` 和 `test`。对于每个通道，可以指定以下信息：
      i. 对于 Channel name (通道名称)，键入通道的名称。名称必须具有 1 到 64 个字符。有效字符为 a-z、A-Z、0-9 和 -（连字符）。

ii. 要将您的算法设为需要通道，请选择 Channel required (需要通道)。

iii. 键入通道的说明。

iv. 对于 Supported input modes (支持的输入模式)，如果算法支持流式传输输入数据，则选择 Pipe mode (管道模式)；如果算法支持以文件形式下载输入数据，则选择 File mode (文件模式)。您可以选择二者。

v. 对于 Supported content types (支持的内容类型)，键入算法预期用于输入数据的 MIME 类型。

vi. 对于 Supported compression type (支持的压缩类型)，如果算法支持 Gzip 压缩，则选择 Gzip。否则，选择 None (无)。

vii. 选择 Add channel (添加通道) 以添加另一个数据输入通道；如果您已添加完通道，则选择 Next (下一步)。

5. 在 Tuning specifications (优化规范) 页面上，提供以下信息：

a. 对于 Hyperparameter specification (超参数规范)，通过编辑 JSON 对象来指定算法支持的超参数。对于算法支持的每个超参数，构造一个类似于以下内容的 JSON 块：

```
{
"DefaultValue": "5",
"Description": "The first hyperparameter",
"IsRequired": true,
"IsTunable": false,
"Name": "intRange",
"Range": {
"IntegerParameterRangeSpecification": {
"MaxValue": "10",
"MinValue": "1"
},
"Type": "Integer"
}
```

在 JSON 中，提供以下内容：

i. 对于 `DefaultValue`，指定超参数的默认值（如果有）。

ii. 对于 `Description`，指定超参数的描述。

iii. 对于 `IsRequired`，指定是否需要超参数。

iv. 对于 `IsTunable`，如果在用户运行使用此算法的超参数优化作业时可以优化此超参数，则指定 `true`。有关信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

v. 对于 `Name`，指定超参数的名称。

vi. 对于 `Range`，指定下列项之一：

- `IntegerParameterRangeSpecification` - 超参数的值为整数。指定超参数的最小值和最大值。
-
- `ContinuousParameterRangeSpecification` - 超参数的值为浮点值。指定超参数的最小值和最大值。
- `CategoricalParameterRangeSpecification` - 超参数的值为分类值。指定所有可能的值的列表。

vii. 对于 `Type`，指定 `Integer`、`Continuous` 或 `Categorical`。该值必须对应于您指定的 `Range` 类型。

b. 适用于指标定义中，指定您希望算法发出的任何训练指标。SageMaker 使用您指定的正则表达式在训练期间解析训练容器在训练期间生成的日志，查找指标。用户可以在使用您的算法运行训练作业时查看这些指标，并且可以在 Amazon CloudWatch 中监视和绘制这些指标。有关信息，请参阅使用 Amazon CloudWatch 指标监控和分析训练作业 (p. 2093)。对于每个指标，提供以下信息：

i. 对于 Metric name (指标名称)，键入指标的名称。

ii. 适用于Regex对于，键入 SageMaker 用来解析训练日志以查找指标值的正则表达式。

iii. 对于 Objective metric support (目标指标支持)，如果此指标可用作超参数优化作业的目标指标，则选择 Yes (是)。有关信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939)。

iv. 选择 Add metric (添加指标) 以添加另一个指标；如果您已添加完指标，则选择 Next (下一步)。

6. 在 Inference specifications (推理规范) 页面上，如果算法支持推理，则提供以下信息：

   a. 适用于推理图像的位置对于，键入 Amazon ECR 中存储推理容器的路径。

   b. 对于 Container DNS host name (容器 DNS 主机名)，键入镜像的 DNS 主机名。

   c. 适用于支持的实时推理实例类型对于，选择算法支持在 SageMaker 中部署为托管终端节点的模型的实例类型。有关信息，请参阅部署推理模型 (p. 2101)。

   d. 对于 Supported instance types for batch transform jobs (支持的批量转换作业实例类型)，选择算法支持的用于批量转换作业的实例类型。有关信息，请参阅使用批量转换 (p. 2229)。

   e. 对于 Supported content types (支持的内容类型)，键入算法预期用于推理请求的输入数据的类型。

   f. 对于 Supported response MIME types (支持的响应 MIME 类型)，键入算法支持的推理响应的 MIME 类型。

   g. 选择 Next (下一步) 。

7. 在 Validation specifications (验证规范) 页面上，提供以下信息：

   a. 适用于在上发布此算法Amazon Web Services Marketplace，选择是要在中发布算法Amazon Web Services Marketplace.

   b. 适用于验证此资源，选择是如果您希望 SageMaker 运行您指定的用来测试算法的训练和/或推理代码的批量转换作业。

      Note

      要在 Amazon Web Services Marketplace 上发布您的算法，必须验证您的算法。

   c. 适用于IAM 角色对于，选择具有在 SageMaker 中运行训练作业和批量转换作业所需的权限的 IAM 角色，或选择创建新角色以允许 SageMaker 创建具有AmazonSageMakerFullAccess已附加托管策略。有关信息，请参阅SageMaker 角色 (p. 2722)。

   d. 对于 Validation profile (验证配置文件)，请指定以下内容：

      - 验证配置文件的名称。
      - Training job definition (训练作业定义)。这是一个描述训练作业的 JSON 数据块。其格式与 CreateAlgorithm API 的 TrainingJobDefinition 输入参数的格式相同。
      - Transform job definition (转换作业定义)。这是一个描述批量转换作业的 JSON 数据块。其格式与 CreateAlgorithm API 的 TransformJobDefinition 输入参数的格式相同。

   e. 选择 Create algorithm (创建算法)。

创建算法资源 (API)

要使用 SageMaker API 创建算法资源，请调用CreateAlgorithmAPI。

创建模型包资源

要创建可用于在亚马逊中创建可部署模型的模型包资源，请执行以下操作： SageMaker 然后发布于Amazon Web Services Marketplace指定以下信息：

- 包含推理代码或已用于训练模型的算法资源的 Docker 容器。
- 模型构件的位置。模型构件既可以打包在与推理代码相同的 Docker 容器中，也可以存储在 Amazon S3 中。
- 模型包支持的用于实时推理和批量转换作业的实例类型。
- 验证配置文件，它们是批量转换作业， SageMaker 运行以测试模型包的推理代码。

在 Amazon Web Services Marketplace 上列出模型包之前，您必须先对其进行验证。这确保买家和卖家可以确信产品适用于 Amazon SageMaker。您只有在验证成功后，才能在 Amazon Web Services Marketplace 上列出产品。

验证过程使用您的验证配置文件和示例数据来运行以下验证任务：

1. 使用模型包的推理镜像和存储在 Amazon S3 中的可选模型构件，在您的账户中创建模型。

   Note

   模型包特定于创建它时所在的区域。存储模型构件的 S3 存储桶必须位于您创建模型包的区域内。
2. 使用该模型在您的账户中创建一个转换作业以验证您的推理镜像是否适用于 SageMaker。
3. 创建验证配置文件。

   Note

   在验证配置文件中，仅提供您希望公开的数据。

验证可能需要几个小时。要查看账户中作业的状态，请在 SageMaker 控制台，请参阅转换作业页面。如果验证失败，您可以从 SageMaker 控制台。修复问题后，请重新创建算法。算法的状态是什么时候`COMPLETED`，在 SageMaker 控制台然后开始上市流程

Note

要在 Amazon Web Services Marketplace 上发布您的模型包，至少需要一个验证配置文件。

您可以使用 SageMaker 控制台或使用 SageMaker API。

主题

- 创建模型包资源（控制台） (p. 2675)
- 创建模型包资源 (API) (p. 2676)

创建模型包资源（控制台）

要在 SageMaker 控制台：

1. 打开 SageMaker 控制台在https://console.aws.amazon.com/sagemaker/.
2. 从左侧菜单中，选择推理.
3. 选择Marketplace 模型包选择，然后选择创建市场模型包.
4. 在 Inference specifications (推理规范) 页面上，提供以下信息：

   a. 对于 Model package name (模型包名称)，键入您的模型包的名称。模型包名称在您的账户和 Amazon 区域中必须是唯一的。名称必须具有 1 到 64 个字符。有效字符为 a-z、A-Z、0-9 和 - (连字符)。

   b. 键入模型包的描述。此描述将显示在 SageMaker 控制台和Amazon Web Services Marketplace.

   c. 对于 Inference specification options (推理规范选项)，选择 Provide the location of the inference image and model artifacts (提供推理镜像和模型构件的位置) 以通过使用推理容器和模型构件来创建模型包。选择 Provide the algorithm used for training and its model artifacts (提供用于训练的算法及其模型构件) 以从您创建的或通过 Amazon Web Services Marketplace 订阅的算法资源创建模型包。

   d. 如果你选择了提供推理图像和模型伪影的位置为了推理规范选项，请提供以下信息：容器定义和支持的资源：

      i. 对于 Location of inference image (推理镜像的位置)，键入包含推理代码的镜像的路径。镜像必须作为 Docker 容器存储在 Amazon ECR 中。

ii. 对于 Location of model data artifacts (模型数据构件的位置)，键入 S3 中存储模型构件的位置。

iii. 对于 Container DNS host name (容器 DNS 主机名)，键入要用于容器的 DNS 主机的名称。

iv. 对于 Supported instance types for real-time inference (支持的实时推理实例类型)，选择模型包支持的用于从 SageMaker 托管终端节点进行实时推理的实例类型。

v. 对于 Supported instance types for batch transform jobs (支持的批量转换作业实例类型)，选择模型包支持的用于批量转换作业的实例类型。

vi. 对于 Supported content types (支持的内容类型)，键入模型包预期用于推理请求的内容类型。

vii. 对于 Supported response MIME types (支持的响应 MIME 类型)，键入模型包用于提供推理的 MIME 类型。

e. 如果你选择了提供用于训练的算法及其模型工件为了推理规范选项，请提供以下信息：

i. 对于 Algorithm ARN (算法 ARN)，键入要用于创建模型包的算法资源的 Amazon 资源名称 (ARN)。

ii. 对于 Location of model data artifacts (模型数据构件的位置)，键入 S3 中存储模型构件的位置。

f. 选择 Next (下一步) 。

5. 在 Validation and scanning (验证和扫描) 页面上，提供以下信息：

a. 适用于在上发布此模型包Amazon Web Services Marketplace，选择是在上发布模型包Amazon Web Services Marketplace.

b. 适用于验证此资源，选择是如果你想 SageMaker 以运行您指定的批量转换作业以测试模型包的推理代码。

Note

要在 Amazon Web Services Marketplace 上发布您的模型包，必须对其进行验证。

c. 适用于IAM 角色选择具有在 SageMaker 中运行批量转换作业所需权限的 IAM 角色，或选择创建新角色允许 SageMaker 创建具有`AmazonSageMakerFullAccess`已附加托管策略。有关信息，请参阅SageMaker 角色 (p. 2722)。

d. 对于 Validation profile (验证配置文件)，请指定以下内容：

- 验证配置文件的名称。
- Transform job definition (转换作业定义)。这是一个描述批量转换作业的 JSON 数据块。其格式与 `CreateAlgorithm` API 的 `TransformJobDefinition` 输入参数的格式相同。

6. 选择创建市场模型包.

创建模型包资源 (API)

要创建模型包，请使用 SageMaker API 调用`CreateModelPackage`API。

使用算法和模型包资源

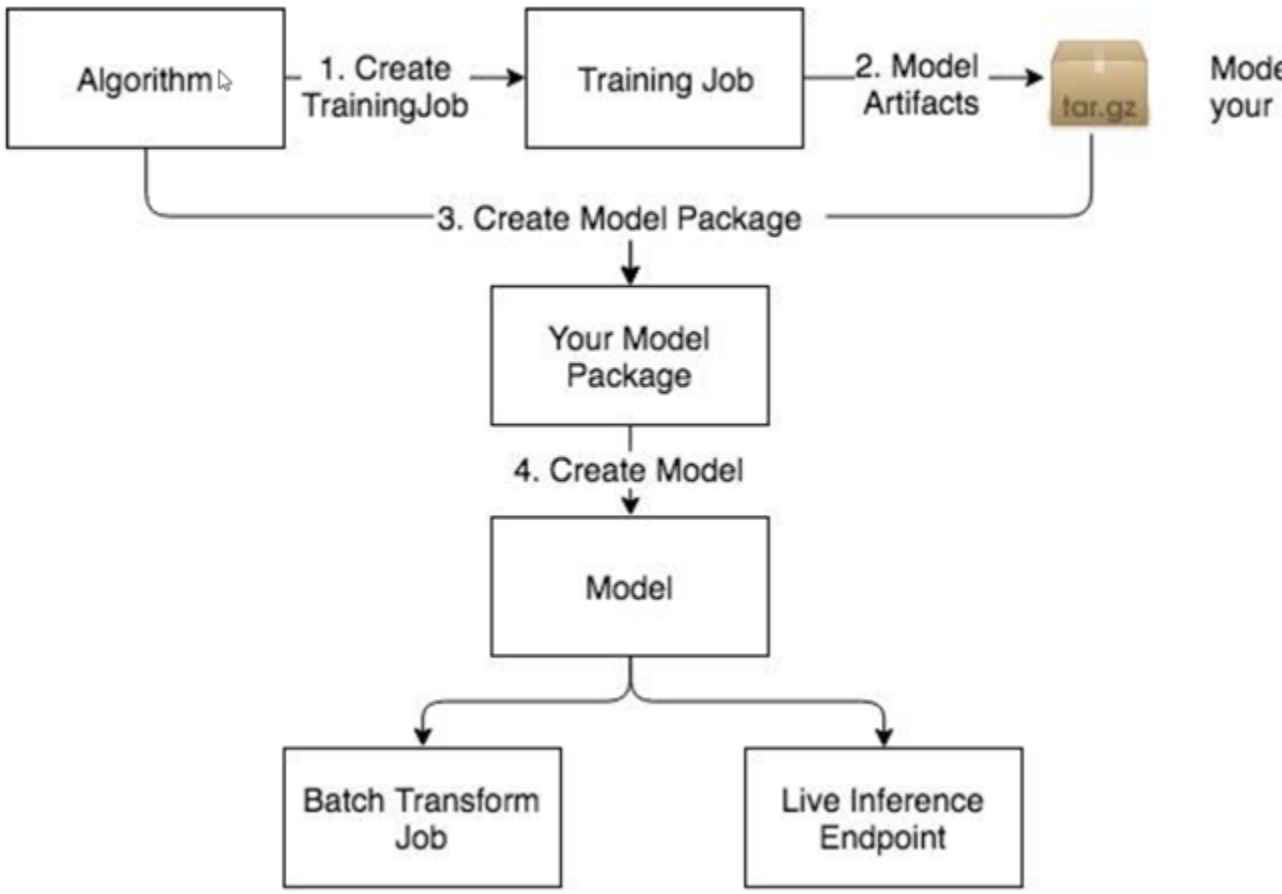
您可以在 Amazon SageMaker 账户中创建算法和模型包作为资源，也可以在上查找和订阅算法和模型包。Amazon Web Services Marketplace.

使用算法可以：

- 运行训练作业。有关信息，请参阅使用算法运行训练作业 (p. 2677)。
- 运行超参数优化作业。有关信息，请参阅使用算法运行超参数优化作业 (p. 2679)。
- 创建模型包。在使用算法资源运行训练作业或超参数优化作业后，可以使用算法和这些作业输出的模型构件来创建模型包。有关信息，请参阅创建模型包资源 (p. 2674)。

Note

如果您在 Amazon Web Services Marketplace 上订阅算法，则必须先创建模型包，然后才能通过创建托管终端节点或运行批量转换作业来使用它获取推理结果。



使用模型包可以：

- 创建可用于获取实时推理或运行批量转换作业的模型。有关信息，请参阅使用模型包创建模型 (p. 2682)。
- 创建托管终端节点以获取实时推理。有关信息，请参阅将模型部署到 SageMaker 托管服务 (p. 69)。
- 创建批量转换作业。有关信息，请参阅（可选）使用 Batch 变换进行预测 (p. 69)。

主题

- 使用算法运行训练作业 (p. 2677)
- 使用算法运行超参数优化作业 (p. 2679)
- 使用模型包创建模型 (p. 2682)

使用算法运行训练作业

您可以使用算法资源通过亚马逊创建训练作业。SageMaker控制台，低级别亚马逊SageMakerAPI，或者亚马逊SageMakerPython 开发工具包.

主题

- 使用算法运行训练作业（控制台）(p. 2678)
- 使用算法运行训练作业 (API) (p. 2679)
- 使用算法运行训练 Job (亚马逊SageMakerPython 开发工具包）(p. 2679)

使用算法运行训练作业（控制台）

使用算法运行训练作业（控制台）

1. 在 SageMakerhttps://console.aws.amazon.com/sagemaker/ 打开 控制台。
2. 选择 Algorithms (算法)。
3. 从列表中选择您创建的算法。我的算法选项卡或者选择您在Amazon Web Services Marketplace订阅选项卡。
4. 选择 Create training job (创建训练作业)。

   将自动选择您选定的算法。
5. 在 Create training job (创建训练作业) 页面上，提供以下信息：

   a. 对于 Job name (作业名称)，请为训练作业键入名称。

   b. 适用于IAM 角色中选择具有所需权限的 IAM 角色以运行训练作业SageMaker，或者选择创建新角色允许SageMaker创建具有`AmazonSageMakerFullAccess`已附加托管策略。有关信息，请参阅SageMaker 角色 (p. 2722)。

   c. 对于 Resource configuration (资源配置)，提供以下信息：

      i. 对于 Instance type (实例类型)，选择要用于训练的实例类型。

      ii. 对于 Instance count (实例计数)，键入要用于训练作业的 ML 实例的数目。

      iii. 对于 Additional volume per instance (GB) (每个实例的附加卷 (GB))，键入要预配置的 ML 存储卷的大小。ML 存储卷存储模型构件和增量状态。

      iv. 适用于加密密钥，如果你想要亚马逊SageMaker要使用Amazon密钥 Management Service 密钥要加密附加到训练实例的 ML 存储卷中的数据，请指定密钥。

      v. 对于 Stopping condition (停止条件)，请指定希望训练作业运行的最长时间（以秒、分钟、小时或天为单位）。

   d. 适用于VPC中，选择您希望允许训练容器访问的 Amazon VPC。有关更多信息，请参阅 授予权 SageMaker 训练作业对 Amazon VPC 中的资源的访问权限 (p. 2796)。

   e. 对于 Hyperparameters (超参数)，请指定要用于训练作业的超参数的值。

   f. 对于 Input data configuration (输入数据配置)，请指定要用于训练作业的每个输入数据通道的以下值。你可以在下面看到你用于训练支持的算法以及每个频道的内容类型、支持的压缩类型和支持的输入模式频道规格的 部分算法摘要该算法的页面。

      i. 对于 Channel name (通道名称)，键入输入通道的名称。

      ii. 对于 Content type (内容类型)，键入算法预期用于通道的数据的内容类型。

      iii. 对于 Compression type (压缩类型)，选择要使用的数据压缩类型（如果有）。

      iv. 对于 Record wrapper (记录包装程序)，如果算法需要 `RecordIO` 格式的数据，则选择 `RecordIO`。

      v. 对于 S3 data type (S3 数据类型)、S3 data distribution type (S3 数据分布类型) 和 S3 location (S3 位置)，请指定适当的值。有关这些值的含义的信息，请参阅`S3DataSource`。

      vi. 对于 Input mode (输入模式)，选择 File (文件) 以将数据下载到预配置的 ML 存储卷，并将目录挂载到 Docker 卷。选择 Pipe (管道) 以直接从 Amazon S3 将数据流式传输到容器。

      vii. 要添加另一个输入通道，请选择 Add channel (添加通道)。如果已添加完输入通道，请选择 Done (完成)。

   g. 对于 Output (输出) 位置，请指定以下值：

i. 对于 S3 output path (S3 输出路径)，请选择训练作业将输出（例如模型构件）存储到的 S3 位置。

Note

您可以使用存储在此位置的模型构件来从该训练作业创建模型或模型包。

ii. 对于 Encryption key (加密密钥)，如果您希望 SageMaker 使用 Amazon KMS 密钥对 S3 位置的静态输出数据进行加密，请指定密钥。

h. 对于 Tags (标签)，请指定一个或多个标签来管理训练作业。每一个标签都包含一个键和一个可选值。每个资源的标签键必须是唯一的。

i. 选择 Create training job (创建训练作业) 以运行训练作业。

使用算法运行训练作业 (API)

要使用算法通过 SageMaker API 运行训练作业，请指定名称或 Amazon 资源名称 (ARN) 作为传递给 `CreateTrainingJob` 的 `AlgorithmSpecification` 对象的 `AlgorithmName` 字段。有关 SageMaker 中的训练模型的信息，请参阅使用 Amazon SageMaker 训练模型 (p. 7)。

使用算法运行训练 Job (亚马逊SageMakerPython 开发工具包)

使用您创建的或在 Amazon Web Services Marketplace 上订阅的算法创建训练作业、创建 `AlgorithmEstimator` 对象并指定 Amazon 资源名称 (ARN) 或算法名称作为 `algorithm_arn` 参数的值。然后，调用评估程序的 `fit` 方法。例如：

```
from sagemaker import AlgorithmEstimator
data_path = os.path.join(DATA_DIR, 'marketplace', 'training')

algo = AlgorithmEstimator(
algorithm_arn='arn:aws:sagemaker:us-east-2:012345678901:algorithm/my-algorithm',
        role='SageMakerRole',
        instance_count=1,
        instance_type='ml.c4.xlarge',
        sagemaker_session=sagemaker_session,
        base_job_name='test-marketplace')

train_input = algo.sagemaker_session.upload_data(
path=data_path, key_prefix='integ-test-data/marketplace/train')

algo.fit({'training': train_input})
```

使用算法运行超参数优化作业

超参数优化作业通过使用您指定的算法和超参数范围在数据集上运行很多训练作业来查找模型的最佳版本。然后，它会选择超参数值来生成性能最佳的模型（按所选指标衡量）。有关更多信息，请参阅 使用 SageMaker 执行自动模型优化 (p. 1939)。

您可以使用算法资源通过 Amazon SageMaker 控制台、低级别 Amazon SageMaker API 或Amazon SageMaker Python 开发工具包.

主题

- 使用算法运行超参数优化作业（控制台）(p. 2680)
- 使用算法运行超参数优化作业 (API) (p. 2681)
- 使用算法运行超参数优化 Job (Amazon SageMaker Python 开发工具包）(p. 2681)

使用算法运行超参数优化作业（控制台）

使用算法运行超参数优化作业（控制台）

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 选择 Algorithms (算法)。
3. 从列表中选择您创建的算法。我的算法选项卡或者选择您在Amazon Web Services Marketplace订阅选项卡。
4. 选择 Create hyperparameter tuning job (创建超参数优化作业)。

   将自动选择您选定的算法。
5. 在 Create hyperparameter tuning job (创建超参数优化作业) 页面上，提供以下信息：

   a. 对于 Warm start (热启动)，选择 Enable warm start (启用热启动) 以使用来自之前的超参数优化作业的信息作为此超参数优化作业的起点。有关更多信息，请参阅 运行热启动超参数优化作业 (p. 1959)。

      i. 选择 Identical data and algorithm (相同的数据和算法)（如果您的输入数据与此超参数优化作业的父作业的输入数据相同），或选择 Transfer learning (迁移学习) 以将附加或其他输入数据用于此超参数优化作业。

      ii. 对于 Parent hyperparameter tuning job(s) (父超参数优化作业)，请选择最多 5 个超参数优化作业以用作此超参数优化作业的父作业。

   b. 对于 Hyperparameter tuning job name (超参数优化作业名称)，请键入优化作业的名称。

   c. 适用于IAM 角色对于 IAM 角色，请选择具有在 SageMaker 中运行超参数优化作业所需的权限的 IAM 角色，或选择创建新角色以允许 SageMaker 创建具有`AmazonSageMakerFullAccess`已附加托管策略。有关信息，请参阅SageMaker 角色 (p. 2722)。

   d. 适用于VPC对于 Amazon VPC，请选择要允许优化作业启动的训练作业访问的。有关更多信息，请参阅 授予权 SageMaker 训练作业对 Amazon VPC 中的资源的访问权限 (p. 2796)。

   e. 选择 Next（下一步）。

   f. 对于 Objective metric (目标指标)，选择超参数优化作业用于确定超参数的最佳组合的指标，并选择是最小化该指标还是最大化该指标。有关更多信息，请参阅 查看最佳训练作业 (p. 1956)。

   g. 对于 Hyperparameter configuration (超参数配置)，选择希望优化作业搜索的可优化超参数的范围，并为要在超参数优化作业启动的所有训练作业中保持不变的超参数设置静态值。有关更多信息，请参阅 定义超参数范围 (p. 1942)。

   h. 选择 Next（下一步）。

   i. 对于 Input data configuration (输入数据配置)，请指定要用于超参数优化作业的每个输入数据通道的以下值。你可以在下面看到你用于超参数调整的算法支持哪些频道，以及每个频道的内容类型、支持的压缩类型和支持的输入模式频道规格的 部分算法摘要该算法的页面。

      i. 对于 Channel name (通道名称)，键入输入通道的名称。

      ii. 对于 Content type (内容类型)，键入算法预期用于通道的数据的内容类型。

      iii. 对于 Compression type (压缩类型)，选择要使用的数据压缩类型（如果有）。

      iv. 对于 Record wrapper (记录包装程序)，如果算法需要 `RecordIO` 格式的数据，则选择 `RecordIO`。

      v. 对于 S3 data type (S3 数据类型)、S3 data distribution type (S3 数据分布类型) 和 S3 location (S3 位置)，请指定适当的值。有关这些值的含义的信息，请参阅`S3DataSource`。

      vi. 对于 Input mode (输入模式)，选择 File (文件) 以将数据下载到预配置的 ML 存储卷，并将目录挂载到 Docker 卷。选择 Pipe (管道) 以直接从 Amazon S3 将数据流式传输到容器。

      vii. 要添加另一个输入通道，请选择 Add channel (添加通道)。如果已添加完输入通道，请选择 Done (完成)。

   j. 对于 Output (输出) 位置，请指定以下值：

i. 对于 S3 output path (S3 输出路径)，请选择此超参数优化作业启动的训练作业将输出（例如模型构件）存储到的 S3 位置。

Note

您可以使用存储在此位置的模型构件来从该超参数优化作业创建模型或模型包。

ii. 适用于加密密钥，如果你希望 SageMaker 使用Amazon KMS密钥对 S3 位置的静态输出数据进行加密。

k. 对于 Resource configuration (资源配置)，提供以下信息：

i. 对于 Instance type (实例类型)，选择要用于超参数优化作业启动的每个训练作业的实例类型。

ii. 对于 Instance count (实例计数)，键入要用于超参数优化作业启动的每个训练作业的 ML 实例的数目。

iii. 对于 Additional volume per instance (GB) (每个实例的附加卷 (GB))，键入预配置超参数优化作业启动的每个训练作业所需的 ML 存储卷的大小。ML 存储卷存储模型构件和增量状态。

iv. 适用于加密密钥，如果您希望 Amazon SageMaker 使用Amazon要对附加到训练实例的 ML 存储卷中的数据进行加密，请指定密钥。

l. 对于 Resource limits (资源限制)，提供以下信息：

i. 对于 Maximum training jobs (最大训练作业数)，请指定您希望超参数优化作业启动的训练作业的最大数目。超参数优化作业最多可启动 500 个训练作业。

ii. 对于 Maximum parallel training jobs (最大并行训练作业数)，请指定超参数优化作业可启动的并发训练作业的最大数目。超参数优化作业最多可启动 10 个并发训练作业。

iii. 对于 Stopping condition (停止条件)，请指定您希望超参数优化作业启动的每个训练作业运行的最长时间（以秒、分钟、小时或天为单位）。

m. 对于 Tags (标签)，请指定一个或多个标签来管理超参数优化作业。每一个标签都包含一个键和一个可选值。每个资源的标签键必须是唯一的。

n. 选择 Create jobs (创建作业) 以运行超参数优化作业。

使用算法运行超参数优化作业 (API)

要使用算法通过 SageMaker API 运行超参数优化作业，请指定算法的名称或 Amazon 资源名称 (ARN) 作为`AlgorithmName`字段中的`AlgorithmSpecification`你传递给的对象`CreateHyperParameterTuningJob`. 有关 SageMaker 中的超参数优化的信息，请参阅使用 SageMaker 执行自动模型优化 (p. 1939).

使用算法运行超参数优化 Job (Amazon SageMaker Python 开发工具包)

使用您创建的或在 Amazon Web Services Marketplace 上订阅的算法创建超参数优化作业、创建 `AlgorithmEstimator` 对象并指定算法的名称或 Amazon 资源名称 (ARN) 作为 `algorithm_arn` 参数的值。然后，使用您创建的 `AlgorithmEstimator` 作为 `estimator` 参数的值来初始化 `HyperparameterTuner` 对象。最后，调用 `AlgorithmEstimator` 的 `fit` 方法。例如：

```
from sagemaker import AlgorithmEstimator
from sagemaker.tuner import HyperparameterTuner

data_path = os.path.join(DATA_DIR, 'marketplace', 'training')

algo = AlgorithmEstimator(
            algorithm_arn='arn:aws:sagemaker:us-east-2:764419575721:algorithm/scikit-
decision-trees-1542410022',
            role='SageMakerRole',
            instance_count=1,
            instance_type='ml.c4.xlarge',
            sagemaker_session=sagemaker_session,
            base_job_name='test-marketplace')
```

```
train_input = algo.sagemaker_session.upload_data(
    path=data_path, key_prefix='integ-test-data/marketplace/train')

algo.set_hyperparameters(max_leaf_nodes=10)
tuner = HyperparameterTuner(estimator=algo, base_tuning_job_name='some-name',
                                objective_metric_name='validation:accuracy',
                                hyperparameter_ranges=hyperparameter_ranges,
                                max_jobs=2, max_parallel_jobs=2)

tuner.fit({'training': train_input}, include_cls_metadata=False)
tuner.wait()
```

使用模型包创建模型

使用模型包创建可部署模型，您可以使用该模型通过创建托管终端节点获取实时推理或运行批量转换作业。您可以使用 Amazon SageMaker 控制台、低级别 SageMaker API 或从模型包创建可部署模型。Amazon SageMaker Python 开发工具包.

主题

- 使用模型包创建模型（控制台）(p. 2682)
- 使用模型包创建模型 (API) (p. 2682)
- 使用模型包创建模型 (Amazon SageMaker Python 开发工具包）(p. 2683)

使用模型包创建模型（控制台）

从模型包中创建可部署的模型（控制台）

1. 从打开 SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 选择 Model packages (模型包)。
3. 从中选择您从中的列表中选择您创建的模型包。我的模型包选项卡或者选择您在Amazon Web Services Marketplace订阅选项卡。
4. 选择 Create model (创建模型)。
5. 对于 Model name (模型名称)，键入模型的名称。
6. 适用于IAM 角色中选择具有代表您调用其他服务的必需权限的 IAM 角色，创建新角色允许 SageMaker 创建具有`AmazonSageMakerFullAccess`已附加托管策略。有关信息，请参阅SageMaker 角色 (p. 2722)。
7. 适用于VPC中选择您希望允许模型访问的 Amazon VPC。有关更多信息，请参阅 授予权 SageMaker 托管终端节点对 Amazon VPC 中的资源的访问权限 (p. 2799)。
8. 保留 Container input options (容器输入选项) 和 Choose model package (选择模型包) 的默认值。
9. 对于环境变量，提供要传递给模型容器的环境变量的名称和值。
10. 对于 Tags (标签)，请指定一个或多个标签来管理模型。每一个标签都包含一个键和一个可选值。每个资源的标签键必须是唯一的。
11. 选择 Create model (创建模型)。

创建可部署模型后，您可以使用它来设置实时推理的终端节点，或创建批量转换作业来获取对整个数据集的推理。有关 SageMaker 中托管终端节点的信息，请参阅部署模型以进行推理.

使用模型包创建模型 (API)

要使用模型包通过 SageMaker API 创建可部署模型，请将模型包的名称或 Amazon 资源名称 (ARN) 指定为`ModelPackageName`字段中的`ContainerDefinition`你传递给的对象`CreateModel`API。

创建可部署模型后，您可以使用它来设置实时推理的终端节点，或创建批量转换作业来获取对整个数据集的推理。有关 SageMaker 中托管终端节点的信息，请参阅部署模型以进行推理.

使用模型包创建模型 (Amazon SageMaker Python 开发工具包)

要使用模型包通过 SageMaker Python 开发工具包创建可部署模型，请初始化`ModelPackage`对象，并将模型包的 Amazon 资源名称 (ARN) 作为`model_package_arn`参数。例如：

```
from sagemaker import ModelPackage
model = ModelPackage(role='SageMakerRole',
         model_package_arn='training-job-scikit-decision-trees-1542660466-6f92',
         sagemaker_session=sagemaker_session)
```

创建可部署模型后，您可以使用它来设置实时推理的终端节点，或创建批量转换作业来获取对整个数据集的推理。有关 SageMaker 中托管终端节点的信息，请参阅部署模型以进行推理.

Amazon SageMaker 中的安全

Amazon 十分重视云安全性。作为 Amazon 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是 Amazon 和您的共同责任。责任共担模型将其描述为云的安全性和云中的安全性：

- 云的安全性 – Amazon负责保护在Amazon云中运行Amazon服务的基础设施。Amazon还向您提供可安全使用的服务。作为 Amazon 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用于 Amazon SageMaker 的合规性计划，请参阅Amazon合规性计划范围内的服务.
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其它因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 SageMaker 时应用责任共担模型。以下主题将向您展示如何配置：SageMaker 以实现您的安全性和合规性目标。您还将了解如何使用其他Amazon帮助您监控和保护您的服务 SageMaker 资源的费用。

主题

- 访问控制 (p. 2684)
- Amazon SageMaker 中的数据保护 (p. 2686)
- 适用于 Amazon SageMaker 的 Identity and Access Management (p. 2690)
- 日志记录和监控 (p. 2779)
- Amazon SageMaker 的合规性验证 (p. 2780)
- Amazon SageMaker 中的恢复能力 (p. 2780)
- Amazon SageMaker 的基础设施安全性 (p. 2780)

访问控制

亚马逊 SageMaker 和 Studio 笔记本 SageMaker 笔记本实例的运行时环境不同。以下主题介绍如何控制对这些笔记本的根访问。

主题

- 访问控制和 SageMaker Studio 笔记本 (p. 2684)
- 控制对 SageMaker 笔记本实例 (p. 2685)

访问控制和 SageMaker Studio 笔记本

亚马逊 SageMaker Studio 使用文件系统和容器权限对 Studio 用户和笔记本进行访问控制和隔离。这是 Studio 笔记本电脑和 SageMaker 笔记本实例。本主题介绍如何设置权限以避免安全威胁， SageMaker 默认情况下，以及客户如何自定义权限。有关 Studio 笔记本及其运行时环境的更多信息，请参阅。使用 Amazon SageMaker Studio Notebooks (p. 82).

SageMaker 应用权限

一个以用户身份运行是用于运行 JupyterServer 应用程序 KernelGateway 容器内的应用程序。

的运行身份用户 JupyterServer 默认情况下，应用程序是 sagemaker 用户 (1000)。此用户拥有 sudo 权限来启用依赖项的安装，例如 yum 包。

的运行身份用户 KernelGateway 默认情况下，应用程序为 root (0)。该用户可以使用 pip/apt-get/conda 安装依赖项。

由于用户重新映射，两个用户都无法访问资源或对主机实例进行更改。

用户重新映射

SageMaker 执行用户重新映射以将容器内的用户映射到容器外主机实例上的用户。容器中的用户 ID 范围 (0-65535) 映射到实例上 65535 以上的非特权用户 ID。例如，容器内的 sagemaker-user (1000) 可能映射到实例上的用户 (200001)，其中括号中的数字是用户 ID。如果客户在容器内创建新的用户/组，则无论用户/组 ID 如何，它都不会在主机实例上获得特权。容器的根用户还会映射到实例上的非特权用户。有关更多信息，请参阅 。使用用户命名空间隔离容器.

自定义映像权限

客户可自带定制 SageMaker 映像。这些图像可以指定不同的用户/组以运行身份来启动 KernelGateway 应用。客户可以在映像内部实施精细的权限控制，例如，禁用 root 访问权限或执行其他操作。这里也适用同样的用户重新映射。有关更多信息，请参阅 自带 SageMaker 图像 (p. 101)。

容器隔离

Docker 保留容器可以使用的默认功能的列表。 SageMaker 不会添加其他功能。 SageMaker 添加特定的路由规则以阻止向 Amazon EFS 发出的请求和实例元数据服务（IMDS）来自容器。客户无法从容器中更改这些路线规则。有关更多信息，请参阅 。运行时权限和 Linux 功能.

访问应用元数据

运行应用程序使用的元数据将以只读权限挂载到容器中。客户无法修改容器中的此元数据。有关可用的元数据，请参阅获取笔记本和应用程序元数据 (p. 91).

EFS 上的用户隔离

当您注册到 Studio 时， SageMaker 为域中的所有 Studio 用户共享的域创建 Amazon Elastic File System (EFS) 卷。每个用户都可以在 EFS 卷上获得自己的私有主目录。此主目录用于存储用户的笔记本、Git 存储库和其他数据。为防止域中的其他用户访问用户的数据，SageMaker 为该用户的个人资料创建了一个全局唯一的用户 ID，并将其应用为用户主目录的 POSIX 用户/组 ID。

EBS 访问权限

Amazon Elastic Block Store (Amazon EBS) 卷附加到主机实例，并在所有映像之间共享。它用于笔记本电脑的根卷，并存储在容器内生成的临时数据。删除运行笔记本电脑的实例时，存储不会持久存储。容器内的 root 用户无法访问 EBS 卷。

IMDS 访问

由于安全考虑，在中无法访问 Amazon Elastic Compute Cloud (Amazon EC2) 实例元数据服务 (IMDS)。SageMaker Studio。有关 IMDS 的更多信息，请参阅。实例元数据和用户数据.

控制对 SageMaker 笔记本实例

默认情况下，当您创建一个笔记本实例时，登录该笔记本实例的用户将拥有根访问权限。数据科学是一个迭代过程，可能需要数据科学家测试和使用不同的软件工具和包，因此许多笔记本实例用户需要具有根访问权限才能安装这些工具和包。由于具有根访问权限的用户具有管理员权限，所以，用户可以在启用根访问权限的情况下，访问和编辑笔记本实例上的所有文件。

如果不希望用户对笔记本实例具有根访问权限，则在调用 `CreateNotebookInstance` 或 `UpdateNotebookInstance` 操作时，将 `RootAccess` 字段设置为 `Disabled`。您也可以在 Amazon SageMaker 控制台中创建或更新笔记本实例时，禁用用户的根访问权限。有关信息，请参阅第 1 步：创建 Amazon SageMaker 笔记本实例 (p. 59)。

Note

生命周期配置需要根访问权限才能设置笔记本实例。因此，即使禁用了用户的根访问权限，与笔记本实例关联的生命周期配置也始终以根访问权限运行。

Note

出于安全考虑，Rootless Docker 安装在禁用根功能的笔记本实例上，而不是普通 Docker 上。有关更多信息，请参阅 。以非 root 用户身份运行 Docker 守护程序（无根模式）

Amazon SageMaker 中的数据保护

这些区域有：Amazon 责任共担模式适用于 Amazon SageMaker 中的数据保护。如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的 Amazon Web Services 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 Amazon Web Services 账户 凭证并使用 Amazon Identity and Access Management (IAM) 设置单独的用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 Amazon 资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 Amazon CloudTrail 设置 API 和用户活动日志记录。
- 使用 Amazon 加密解决方案以及 Amazon 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Simple Storage Service（Amazon S3）中的个人数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括您与 Amazon 合作时 SageMaker 或其他Amazon使用控制台、API、Amazon CLI，或者Amazon开发工具包。您在用于名称的标签或自由格式字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

主题

- 使用加密保护静态数据 (p. 2686)
- 使用加密保护传输中的数据 (p. 2688)
- 密钥管理 (p. 2690)
- 互联网络流量保密性 (p. 2690)

使用加密保护静态数据

为了保护你的亚马逊 SageMaker 和 Studio 笔记本 SageMaker 笔记本实例以及模型构建数据和模型构件， SageMaker 加密笔记本电脑，以及训练和 Batch 转换作业的输出。 SageMaker 默认情况下使用

AmazonAmazon S3 的托管密钥。该Amazon无法共享 Amazon S3 的托管密钥以进行跨账户访问。要进行跨账户访问，请在创建时指定您的客户托管密钥 SageMaker 资源以便共享该资源以进行跨账户访问。有关 Amazon KMS，请参阅是什么AmazonKey Management Service？.

主题

- Studio 笔记本 (p. 2687)
- 笔记本实例 SageMaker 工作 (p. 2687)

Studio 笔记本

在 Amazon 中 SageMaker 工作室，你 SageMaker Studio 笔记本电脑和数据可以存储在以下位置：

- S3 存储桶 — 当您加载到 Studio 并启用可共享的笔记本资源时， SageMaker 在 Amazon Simple Storage Service (Amazon S3) 存储桶中共享笔记本快照和元数据。
- EFS 卷 — 当你加入工作室时， SageMaker 将 Amazon Elastic File System (Amazon EFS) 卷附加到您的域以存储您的 Studio 笔记本和数据文件。删除域后，EFS 卷仍然存在。
- EBS 卷 — 当您在 Studio 中打开笔记本时，该笔记本所在的实例附加 Amazon EBS) 的 Amazon Elastic Block Store (Amazon EBS)。EBS 卷在实例持续期内一直存在。

SageMaker 使用Amazon Key Management Service(Amazon KMS) 来加密 S3 存储桶和两个卷。默认情况下，它使用Amazon 托管式密钥. 要获取更多控制，您可以在登入 Studio 时或通过 SageMaker API。有关更多信息，请参阅 。注册到 Amazon SageMaker 域 (p. 32)和CreateDomain.

在`CreateDomain`API，您将`S3KmsKeyId`参数来指定可共享笔记本的客户管理密钥。您可以使用`KmsKeyId`参数来指定 EFS 和 EBS 卷的客户管理密钥。两个卷都使用相同的客户管理密钥。可共享笔记本电脑的客户管理密钥可以是用于卷的同一个客户管理的密钥，也可以是不同的客户管理密钥。

笔记本实例 SageMaker 工作

要加密附加到笔记本、处理作业、训练作业、超参数优化作业、批量转换作业和终端节点的机器学习 (ML) 存储卷，您可以通过Amazon KMSSageMaker 的钥匙。如果未指定 KMS 密钥， SageMaker 使用临时密钥加密存储卷，并在加密存储卷后立即放弃该密钥。对于笔记本实例，如果您没有指定 KMS 密钥，SageMaker 会使用系统管理的 KMS 密钥对操作系统卷和机器学习数据卷进行加密。

您可以使用Amazon管理Amazon KMS键以加密所有实例操作系统卷。您可以对所有 ML 数据卷进行加密 SageMaker 具有Amazon KMS键是您指定的键。按以下方式挂载 ML 存储卷：

- 笔记本 - `/home/ec2-user/SageMaker`
- 处理 - `/opt/ml/processing` 和 `/tmp/`
- 训练 - `/opt/ml/` 和 `/tmp/`
- 批处理 - `/opt/ml/` 和 `/tmp/`
- 终端节点 - `/opt/ml/` 和 `/tmp/`

处理、批量转换和训练作业容器及其存储本质上是短暂的。作业完成后，输出将使用上传到 Amazon S3Amazon KMS使用可选的加密Amazon KMS您指定的键并且实例被拆除。如果Amazon KMS作业请求中没有提供密钥，默认情况下输出会使用AmazonAmazon S3 的托管密钥。

> **Note**
>
> 的关键政策Amazon无法编辑 Amazon S3 的托管密钥，因此无法为这些密钥策略授予跨账户权限。如果请求的输出 Amazon S3 存储桶来自另一个账户，请指定您自己的Amazon KMS作业请求中的客户密钥，并确保作业的执行角色具有使用它加密数据的权限。

Important

出于合规性原因需要使用 KMS 密钥加密的敏感数据应存储在 ML 存储卷或 Amazon S3 中，这两种方法都可以使用您指定的 KMS 密钥进行加密。

当你打开笔记本实例时， SageMaker 将其以及与其关联的任何文件保存在 SageMaker 默认情况下，ML 存储卷中的文件夹。当您停止笔记本实例时，SageMaker 将创建 ML 存储卷的快照。对已停止实例的操作系统的任何自定义设置（例如已安装的自定义库或操作系统级别设置）都将丢失。考虑使用生命周期配置来自动执行默认笔记本实例的自定义。当您终止实例时，快照和 ML 存储卷将被删除。您需要在笔记本实例的生命周期之后保留的任何数据都应传输到 Amazon S3 存储桶。

Note

某些基于硝基的 SageMaker 具体实例包括本地存储，具体取决于实例类型。使用实例上的硬件模块对本地存储卷进行加密。无法在具有本地存储的实例类型上使用 KMS 密钥。有关支持本地实例存储的实例类型的列表，请参阅实例存储卷。有关基于 Nitory 的实例上的存储卷的更多信息，请参阅Linux 实例上的 Amazon EBS 和 NVMe.
有关本地实例存储加密的更多信息，请参阅 SSD 实例存储卷。

使用加密保护传输中的数据

所有传输中的网际数据都支持 TLS 1.2 加密。

亚马逊 SageMaker 确保在传输和静止状态时对机器学习 (ML) 模型项目和其他系统项目进行加密。向 SageMaker API 和控制台通过安全 (SSL) 连接创建。传递Amazon Identity and Access Management角色到 SageMaker 提供代表您访问资源以进行训练和部署的权限。您可以将加密的 Amazon S3 存储桶用于模型工件和数据，也可以通过Amazon KMS关键 SageMaker 实例来加密附加的 ML 存储卷。

某些网内数据在传输（在服务平台内）未加密。这包括：

- 服务控制层面和训练作业实例（不是客户数据）之间的命令和控制通信。
- 分布式处理作业（网络内）中节点之间的通信。
- 分布式训练作业（网络内）中节点之间的通信。

没有用于批处理的节点间通信。

您可以选择加密训练群集中节点之间的通信。有关如何执行此操作的信息，请参阅 保护分布式训练作业中机器学习计算实例之间的通信 (p. 2688)。启用容器间流量加密可能会增加训练时间，在您使用分布式深度学习算法时尤其如此。对于受影响的算法，添加此另一层安全性还会增加成本。大多数人的训练时间 SageMaker 内置算法（如 XGBoost、DeepAR 和线性学习器）通常不受影响。

FIPS 验证的终端节点可用于 SageMaker 托管模型的 API 和请求路由器（运行时）。有关 FIPS 兼容终端节点的信息，请参阅美国联邦信息处理标准 (FIPS) 140-2.

保护分布式训练作业中机器学习计算实例之间的通信

默认情况下，亚马逊 SageMaker 运行训练工作在 Amazon Virtual Private Cloud (Amazon VPC) 中，以帮助保护您的数据安全。您可以通过配置一个私有 VPC 添加另一层安全性来保护您的训练容器和数据。分布式 ML 框架和算法通常传输与模型直接相关的信息（如权重），而不是传输训练数据集。当执行分布式训练时，您可以进一步保护在实例之间传输的数据。这可以帮助您遵守法规要求。为此，请使用容器间流量加密。

启用容器间流量加密可能会增加训练时间，在您使用分布式深度学习算法时尤其如此。启用容器间流量加密不会影响具有单个计算实例的训练作业。但是，对于具有多个计算实例的训练作业，对训练时间的影响取决于计算实例之间的通信量。对于受影响的算法，添加此另一层安全性还会增加成本。大多数人的训练时间 SageMaker 内置算法（如 XGBoost、DeepAR 和线性学习器）通常不受影响。

您可以为训练作业或超级参数优化作业启用容器间流量加密。您可以使用 SageMaker 启用容器间流量加密的 API 或控制台。

有关在私有 VPC 中运行训练作业的信息，请参阅。授予权 SageMaker 训练作业对 Amazon VPC 中的资源的访问权限 (p. 2796).

启用容器间流量加密 (API)

使用 API 在训练或超参数优化作业上启用容器间流量加密之前，您需要将入站和出站规则添加到私有 VPC 的安全组中。

启用容器间流量加密 (API)

1. 在安全组中为您的私有 VPC 添加以下入站和出站规则：

协议	端口范围	源
UDP	500	##### *ID*
ESP 50	N/A	##### *ID*

2. 当您将请求发送到`CreateTrainingJob`要么`CreateHyperParameterTuningJob`API，请指定`True`(对于)`EnableInterContainerTrafficEncryption`参数r。

   Note

   对于`ESP 50`协议，Amazon安全组控制台可能会将端口范围显示为 “全部”。但是，Amazon EC2 会忽略指定的端口范围，因为它不适用于 ESP 50 IP 协议。

启用容器间流量加密（控制台）

在训练作业中启用容器间流量加密

在训练作业中启用容器间流量加密

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中，选择 Training (训练)，然后选择 Training jobs (训练作业)。
3. 选择 Create training job (创建训练作业)。
4. UNDER网络，选择VPC. 可以使用默认 VPC 或您创建的 VPC。
5. 选择启用容器间流量加密.

启用容器间流量加密后，完成训练作业的创建。有关更多信息，请参阅 第 4 步：训练模型 (p. 64)。

在超级参数优化作业中启用容器间流量加密

在超级参数优化作业中启用容器间流量加密

1. 打开 Amazon SageMaker 控制台https://console.aws.amazon.com/sagemaker/.
2. 在导航窗格中，选择训练，然后选择超参数调整作业.
3. 选择 Create hyperparameter tuning job (创建超参数优化作业)。
4. UNDER网络，选择VPC. 可以使用默认 VPC 或您创建的 VPC。
5. 选择 Enable inter-container traffic encryption (启用容器间流量加密)。

启用容器间流量加密后，完成超级参数优化作业的创建。有关更多信息，请参阅 配置并启动超参数优化作业 (p. 1950)。

密钥管理

客户可以指定Amazon KMS密钥（包括自带密钥 (BYOK)），用于 Amazon S3 输入/输出存储桶和机器学习 (ML) Amazon EBS 卷的信封加密。对于笔记本实例以及处理、训练和托管模型 Docker 容器的 ML 卷，可以选择使用 Amazon KMS 客户拥有的密钥进行加密。所有实例操作系统卷都使用 Amazon 托管的 Amazon KMS 密钥进行加密。

> Note
>
> 某些基于 Nitro 的实例包括本地存储，具体取决于实例类型。使用实例上的硬件模块对本地存储卷进行加密。在将实例类型用于本地存储时，无法请求 `VolumeKmsKeyId`。
> 有关支持本地实例存储的实例类型的列表，请参阅实例存储卷。
> 有关本地实例存储加密的更多信息，请参阅 SSD 实例存储卷。
> 有关基于 Nito 的实例上的存储卷的更多信息，请参阅Linux 实例上的 Amazon EBS 和 NVMe.

有关的信息Amazon KMS钥匙请参阅是什么AmazonKey Management Service？中的Amazon Key Management Service开发人员指南.

互联网络流量保密性

本主题介绍 Amazon 如何 SageMaker 确保从服务到其他位置的连接安全。

互联网络通信支持所有组件和客户端之间的 TLS 1.2 加密。

实例可以连接到客户 VPC，从而提供对 S3 VPC 终端节点或客户存储库的访问。如果为笔记本禁用服务平台 Internet 出口，则客户可以通过此接口管理 Internet 出口。对于训练和托管，当连接到客户的 VPC 时，通过服务平台的出口不可用。

默认情况下，对已发布的终端节点进行的 API 调用会遍历公共网络到请求路由器。 SageMaker 支持 Amazon Virtual Private Cloud 接口端节点Amazon PrivateLink 用于客户的 VPC 与请求路由器之间的私有连接，以访问托管模型终端节点。有关 Amazon VPC 的信息，请参阅连接到 SageMaker 通过 VPC 接口终端节点节点 (p. 2785)

适用于 Amazon SageMaker 的 Identity and Access Management

Amazon Identity and Access Management (IAM) 是一项 Amazon Web Service，可以帮助管理员安全地控制对 Amazon 资源的访问。IAM 管理员控制哪些人可以身份验证（已登录）和授权（有权限）可以使用 SageMaker 资源的费用。IAM 是一项无需额外费用即可使用的 Amazon Web Service。

主题

- Audience (p. 2691)
- 使用身份进行身份验证 (p. 2691)
- 使用策略管理访问 (p. 2692)
- Amazon 的方式 SageMaker 与 IAM 协同工作 (p. 2694)
- Amazon SageMaker 基于身份的策略示例 (p. 2696)
- 跨服务困惑副预防 (p. 2717)
- SageMaker 角色 (p. 2722)
- 亚马逊 SageMaker API 权限：操作、权限和资源参考 (p. 2739)
- AmazonAmazon SageMaker 的托管式策略 (p. 2754)

- 排查 Amazon SageMaker 身份和访问权限 (p. 2778)

Audience

如何使用Amazon Identity and Access Management(IAM) 因您可以在 SageMaker 中执行的操作而异。

服务用户— 如果您使用 SageMaker 服务以完成您的工作，则您的管理员会为您提供所需的凭证和权限。随着你使用更多 SageMaker 功能来完成工作时，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 SageMaker 中的功能，请参阅排查 Amazon SageMaker 身份和访问权限 (p. 2778).

服务管理员— 如果你负责 SageMaker 您公司的资源，则您可能具有访问 SageMaker 的完全访问权限。确定哪个是你的工作 SageMaker 您的员工应访问的功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 搭配使用的更多信息 SageMaker 请参阅。Amazon 的方式 SageMaker 与 IAM 协同工作 (p. 2694).

IAM 管理员— 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 SageMaker 的访问的详细信息。查看示例 SageMaker 您可在 IAM 中使用的基于身份的策略，请参阅Amazon SageMaker 基于身份的策略示例 (p. 2696).

使用身份进行身份验证

身份验证是您使用身份凭证登录 Amazon 的方法。有关使用Amazon Web Services Management Console 登录的更多信息，请参阅 IAM 用户指南中的以 IAM 用户或根用户身份登录Amazon Web Services Management Console。

您必须作为Amazon Web Services 账户根用户、IAM 用户或代入 IAM 角色以进行身份验证（登录到 Amazon）。您还可以使用公司的单一登录身份验证方法，甚至使用 Google 或 Facebook 登录。在这些情况下，您的管理员以前使用 IAM 角色设置了联合身份验证。在您使用来自其它公司的凭证访问 Amazon 时，您间接地代入了角色。

要直接登录到Amazon Web Services Management Console，请将密码与根用户电子邮件地址或 IAM 用户名一起使用。您可以使用根用户或 IAM 用户访问密钥以编程方式访问 Amazon。Amazon 提供了 SDK 和命令行工具，可使用您的凭证对您的请求进行加密签名。如果您不使用 Amazon 工具，则必须自行对请求签名。使用 Signature Version 4（用于对入站 API 请求进行验证的协议）完成此操作。有关验证请求的更多信息，请参阅《Amazon 一般参考》中的 Signature Version 4 签名流程。

无论使用何种身份验证方法，您可能还需要提供其它安全信息。例如，Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的在 Amazon 中使用多重身份验证 (MFA)。

Amazon Web Services 账户根用户

当您首次创建 Amazon Web Services 账户时，最初使用的是一个对账户中所有 Amazon Web Services和资源有完全访问权限的单点登录身份。此身份称为Amazon Web Services 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不使用根用户执行日常任务，即使是管理任务。相反，请遵循仅使用根用户创建您的第一个 IAM 用户的最佳实践。然后请妥善保存根用户凭证，仅用它们执行少数账户和服务管理任务。

IAM 用户和组

IAM 用户是Amazon Web Services 账户内对某个人员或应用程序具有特定权限的一个身份。IAM 用户可能具有长期凭证，例如用户名和密码或一组访问密钥。要了解如何生成访问密钥，请参阅 IAM 用户指南 中的管理 IAM 用户的访问密钥。为 IAM 用户生成访问密钥时，请确保查看并安全保存密钥对。您以后无法找回秘密访问密钥，而是必须生成新的访问密钥对。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是Amazon Web Services 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 Amazon Web Services Management Console 中暂时代入 IAM 角色。您可以调用 Amazon CLI 或 Amazon API 操作或使用自定义 URL 以代入角色。有关使用角色的方法的更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 临时 IAM 用户权限 – IAM 用户可以代入 IAM 角色，以暂时获得不同的权限以执行特定的任务。
- 联合身份用户访问 – 您可以不创建 IAM 用户，而是使用来自 Amazon Directory Service、您的企业用户目录或 Web 身份提供商的现有身份。这些用户被称为联合用户。在通过身份提供商请求访问权限时，Amazon 将为联合身份用户分配角色。有关联合身份用户的更多信息，请参阅 IAM 用户指南 中的联合身份用户和角色。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 Amazon Web Services，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》https://docs.amazonaws.cn/IAM/latest/UserGuide/id_roles_compare-resource-policies.html中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 – 某些 Amazon Web Services使用其它 Amazon Web Services中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service（Amazon S3）中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 主体权限 – 当您使用 IAM 用户或角色在 Amazon 中执行操作时，您将被视为主体。策略向主体授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否需要策略中的其他依赖操作，请参阅。Amazon SageMaker 的操作、资源和条件键中的服务授权参考.
  - 服务角色 – 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 Amazon Web Service 委派权限的角色。
  - 服务相关角色 – 服务相关角色是与 Amazon Web Service关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 Amazon 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 IAM 身份或Amazon资源，以便控制Amazon中的访问。策略是Amazon中的对象；在与标识或资源相关联时，策略定义它们的权限。您可以通过 root 用户或 IAM 用户身份登录，也可以代入 IAM 角色。随后，当您提出请求时，Amazon会评估相关的基于身份或基于资源的策略。策略中的权限确

定是允许还是拒绝请求。大多数策略在 Amazon 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南 中的 JSON 策略概览。

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

每个 IAM 实体（用户或角色）最初没有任何权限。换言之，预设情况下，用户什么都不能做，甚至不能更改他们自己的密码。要为用户授予执行某些操作的权限，管理员必须将权限策略附加到用户。或者，管理员可以将用户添加到具有预期权限的组中。当管理员为某个组授予访问权限时，该组内的全部用户都会获得这些访问权限。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管式策略是可以附加到Amazon Web Services 账户中的多个用户、组和角色的独立策略。托管式策略包括 Amazon 托管式策略和客户托管式策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南 中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合身份用户或 Amazon Web Services。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 Amazon 托管式策略。

访问控制列表 (ACL)

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Simple Storage Service（Amazon S3）、Amazon WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的访问控制列表 (ACL) 概览。

其他策略类型

Amazon 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体的基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 Amazon Organizations 中的最大权限。Amazon Organizations 服务可以分组和集中管理您的企业拥有的多个 Amazon Web Services 账户 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成

员账户中实体的权限，包括每个 Amazon Web Services 账户 根用户。有关 Organizations 和 SCP 的更多信息，请参阅《Amazon Organizations 用户指南》中的 SCP 的工作原理。

- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 Amazon 如何确定在涉及多种策略类型时是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon 的方式 SageMaker 与 IAM 协同工作

在使用 IAM 管理对 SageMaker 的访问之前，您应了解哪些 IAM 功能可与 SageMaker 协同工作。获得如何从概要了解 SageMaker 和其他Amazon使用 IAM 的服务，请参阅Amazon使用 IAM 的服务中的IAM 用户指南.

主题

- SageMaker 基于身份的策略 (p. 2694)

SageMaker 基于身份的策略

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。SageMaker 支持特定的操作、资源和条件键。要了解在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南 中的 IAM JSON 策略元素参考。

操作

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 Amazon API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

中的策略操作 SageMaker 在操作前使用以下前缀：`sagemaker:`. 例如，要授予某人运行 SageMaker 使用 SageMaker 进行培训`CreateTrainingJob`API 操作，你包括`sagemaker:CreateTrainingJob`他们的政策中的行动。策略声明必须包括`Action`要么`NotAction`元素。 SageMaker 定义了一组自己的操作，以描述您可以使用该服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

```
"Action": [
      "sagemaker:action1",
      "sagemaker:action2"
]
```

您也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，包括以下操作：

```
"Action": "sagemaker:Describe*"
```

要查看的列表 SageMaker 操作，请参阅Amazon SageMaker 的操作、资源和条件键中的服务授权参考.

资源

SageMaker 不支持在策略中指定资源 ARN。

条件键

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 Amazon 使用逻辑 `AND` 运算评估它们。如果您为单个条件键指定多个值，则 Amazon 使用逻辑 `OR` 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅 IAM 用户指南 中的 IAM 策略元素：变量和标签。

Amazon支持全局条件键和特定于服务的条件键。要查看所有 Amazon 全局条件键，请参阅《IAM 用户指南》中的 Amazon 全局条件上下文键。

SageMaker 定义了自己的一组条件键，还支持使用一些全局条件键。要查看所有 Amazon 全局条件键，请参阅《IAM 用户指南》中的 Amazon 全局条件上下文键。

SageMaker 支持许多特定于服务的条件键，您可以将这些键用于以下操作的细粒度访问控制：

- `CreateProcessingJob`
- `CreateTrainingJob`
- `CreateModel`
- `CreateEndpointConfig`
- `CreateTransformJob`
- `CreateHyperParameterTuningJob`
- `CreateLabelingJob`
- `CreateNotebookInstance`
- `UpdateNotebookInstance`

要查看的列表 SageMaker 条件键，请参阅Amazon 的条件键 SageMaker 中的IAM 用户指南. 要了解您可以对哪些操作和资源使用条件键，请参阅Amazon SageMaker 定义的操作.

有关使用的示例 SageMaker 请参阅以下内容：的控制创建 SageMaker 使用条件键的资源 (p. 2706).

示例

查看示例 SageMaker 请参阅基于身份的策略。Amazon SageMaker 基于身份的策略示例 (p. 2696).

SageMaker 基于资源的策略

SageMaker 不支持基于资源的策略。

基于 SageMaker 标签的授权

您可以将标签附加到 SageMaker 向 SageMaker 的请求中的资源或传递标签。要基于标签控制访问，您需要使用 `sagemaker:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在

策略的条件元素中提供标签信息。有关标记 SageMaker 资源的更多信息，请参阅控制对 的访问 SageMaker 使用标签资源 (p. 2715)。

要查看基于身份的策略（用于根据资源上的标签来限制对该资源的访问）的示例，请参阅控制对 的访问 SageMaker 使用标签资源 (p. 2715)。

SageMaker IAM 角色

IAM 角色是 Amazon 账户中具有特定权限的实体。

将临时凭证用于 SageMaker

您可以使用临时凭证进行联合身份登录，担任 IAM 角色或担任跨账户角色。您可以通过调用 Amazon STS API 操作（如 AssumeRole 或 GetFederationToken）获得临时安全凭证。

SageMaker 支持使用临时凭证。

服务相关角色

SageMaker 部分支持服务相关角色. 服务相关角色 SageMaker Studio 和 SageMaker 训练任务。

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其它服务中的资源以代表您完成操作。服务角色显示在您的 IAM 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

SageMaker 支持服务角色。

在 SageMaker 中选择 IAM 角色

当您在 SageMaker 中创建笔记本实例、处理作业、训练作业、托管终端节点或批量转换作业资源时，您必须选择一个角色以允许 SageMaker 访问 SageMaker 代表您。如果您之前已经创建了一个服务角色或服务相关角色，则 SageMaker 会为您提供一个角色列表供您选择。选择允许访问所需 Amazon 操作和资源的角色非常重要。有关更多信息，请参阅 SageMaker 角色 (p. 2722)。

Amazon SageMaker 基于身份的策略示例

预设情况下，IAM 用户和角色没有创建或修改 SageMaker 资源的权限。它们还无法使用 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 执行任务。IAM 管理员必须创建 IAM 策略，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。要了解如何将策略附加到 IAM 用户或组，请参阅添加和删除 IAM 身份权限中的IAM 用户指南.

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅 IAM 用户指南 中的在 JSON 选项卡上创建策略。

主题

- 策略最佳实践 (p. 2697)
- 使用 SageMaker 控制台 (p. 2697)
- 允许用户查看他们自己的权限 (p. 2705)
- 的控制创建 SageMaker 使用条件键的资源 (p. 2706)
- 控制对的访问 SageMaker 使用基于身份的策略 API (p. 2713)
- 限制访问 SageMaker 按 IP 地址划分的 API 和运行时调用 (p. 2714)
- 按 IP 地址限制对笔记本实例的访问 (p. 2714)

- 控制对 的访问 SageMaker 使用标签资源 (p. 2715)
- 要求 API 调用存在或不存在标签 (p. 2717)

策略最佳实践

基于身份的策略非常强大。它们决定某个人是否可以创建、访问或删除 SageMaker 您的账户中的资源。这些操作可能会使 Amazon Web Services 账户 产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 开始使用Amazon托管策略— 开始使用 SageMaker 快速使用Amazon托管策略，以向您的员工授予他们所需的权限。这些策略已在您的账户中提供，并由 Amazon 维护和更新。有关更多信息，请参阅 IAM 用户指南中的开始使用 Amazon 托管式策略中的权限。
- 授予最低权限 – 创建自定义策略时，仅授予执行任务所需的许可。最开始只授予最低权限，然后根据需要授予其它权限。这样做比起一开始就授予过于宽松的权限而后再尝试收紧权限来说更为安全。有关更多信息，请参阅 IAM 用户指南 中的授予最低权限。
- 为敏感操作启用 MFA – 为增强安全性，要求 IAM 用户使用多重身份验证 (MFA) 来访问敏感资源或 API 操作。要了解更多信息，请参阅《IAM 用户指南》中的在 Amazon 中使用多重身份验证 (MFA)。
- 使用策略条件来增强安全性 – 在切实可行的范围内，定义基于身份的策略在哪些情况下允许访问资源。例如，您可编写条件来指定请求必须来自允许的 IP 地址范围。您也可以编写条件，以便仅允许指定日期或时间范围内的请求，或者要求使用 SSL 或 MFA。有关更多信息，请参阅 。IAM JSON 策略元素：Condition中的IAM 用户指南.

使用 SageMaker 控制台

访问亚马逊 SageMaker 控制台，您必须拥有一组最低的权限。这些权限必须允许您列出和查看有关您的 Amazon 账户中的 SageMaker 资源的详细信息。如果您创建的基于身份的策略比所需的最低权限更严格，则无法为具有该策略的实体（IAM 用户或角色）正常运行控制台。

为确保这些实体仍然可以使用 SageMaker 控制台，还要附加以下内容Amazon对实体进行托管策略。有关更多信息，请参阅 IAM 用户指南中的为用户添加权限：

对于只需要调用 Amazon CLI 或 Amazon API 的用户，无需为其提供最低控制台权限。相反，只允许访问与您尝试执行的 API 操作相匹配的操作。

主题

- 使用 Amazon SageMaker 控制台所需的权限 (p. 2697)
- 使用亚马逊所需的权限 SageMaker TGround Truth 控制台 (p. 2699)
- 使用 Amazon Augmented AI（预览版）控制台所需的权限 (p. 2700)

使用 Amazon SageMaker 控制台所需的权限

权限参考表列出了亚马逊 SageMaker API 操作并显示每个操作所需的权限。有关 Amazon SageMaker API 操作的更多信息，请参阅。亚马逊 SageMaker API 权限：操作、权限和资源参考 (p. 2739).

使用亚马逊 SageMaker 在控制台上，您需要授予对其他操作的权限。具体而言，控制台需要允许 `ec2` 操作显示子网、VPC 和安全组的权限。或者，控制台需要授予权限才能创建执行角色用于任务，例如`CreateNotebook`、`CreateTrainingJob`, 和`CreateModel`. 使用以下权限策略授予这些权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
          "Sid": "SageMakerApis",
```

```
        "Effect": "Allow",
        "Action": [
          "sagemaker:*"
        ],
        "Resource": "*"
      },
      {
        "Sid": "VpcConfigurationForCreateForms",
        "Effect": "Allow",
        "Action": [
          "ec2:DescribeVpcs",
          "ec2:DescribeSubnets",
          "ec2:DescribeSecurityGroups"
        ],
        "Resource": "*"
      },
      {
          "Sid":"KmsKeysForCreateForms",
          "Effect":"Allow",
          "Action":[
            "kms:DescribeKey",
            "kms:ListAliases"
          ],
          "Resource":"*"
      },
      {
        "Sid": "AccessAwsMarketplaceSubscriptions",
        "Effect": "Allow",
        "Action": [
          "aws-marketplace:ViewSubscriptions"
        ],
        "Resource": "*"
      },
      {
        "Effect": "Allow",
        "Action": [
          "codecommit:BatchGetRepositories",
          "codecommit:CreateRepository",
          "codecommit:GetRepository",
          "codecommit:ListRepositories",
          "codecommit:ListBranches",
          "secretsmanager:CreateSecret",
          "secretsmanager:DescribeSecret",
          "secretsmanager:ListSecrets"
        ],
        "Resource": "*"
      },
      {
        "Sid":"ListAndCreateExecutionRoles",
        "Effect":"Allow",
        "Action":[
          "iam:ListRoles",
          "iam:CreateRole",
          "iam:CreatePolicy",
          "iam:AttachRolePolicy"
        ],
        "Resource":"*"
      },
      {
        "Sid": "DescribeECRMetaData",
        "Effect": "Allow",
        "Action": [
            "ecr:Describe*"
        ],
        "Resource": "*"
      },
```

```
        {
          "Sid": "PassRoleForExecutionRoles",
          "Effect": "Allow",
          "Action": [
            "iam:PassRole"
          ],
          "Resource": "*",
          "Condition": {
            "StringEquals": {
                "iam:PassedToService": "sagemaker.amazonaws.com"
            }
          }
        }
    ]
}
```

使用亚马逊所需的权限 SageMaker TGround Truth 控制台

使用亚马逊 SageMaker 您需要授予对其他资源的权限。具体来说，控制台需要对Amazon要查看订阅的 Marketplace，用于管理您的专用员工的 Amazon Cognito 操作，以及 Amazon S3 操作的权限来访问您的输入和输出文件，以及Amazon Lambda用于列出和调用函数的操作。使用以下权限策略授予这些权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GroundTruthConsole",
            "Effect": "Allow",
            "Action": [
                "aws-marketplace:DescribeListings",
                "aws-marketplace:ViewSubscriptions",

                "cognito-idp:AdminAddUserToGroup",
                "cognito-idp:AdminCreateUser",
                "cognito-idp:AdminDeleteUser",
                "cognito-idp:AdminDisableUser",
                "cognito-idp:AdminEnableUser",
                "cognito-idp:AdminRemoveUserFromGroup",
                "cognito-idp:CreateGroup",
                "cognito-idp:CreateUserPool",
                "cognito-idp:CreateUserPoolClient",
                "cognito-idp:CreateUserPoolDomain",
                "cognito-idp:DescribeUserPool",
                "cognito-idp:DescribeUserPoolClient",
                "cognito-idp:ListGroups",
                "cognito-idp:ListIdentityProviders",
                "cognito-idp:ListUsers",
                "cognito-idp:ListUsersInGroup",
                "cognito-idp:ListUserPoolClients",
                "cognito-idp:ListUserPools",
                "cognito-idp:UpdateUserPool",
                "cognito-idp:UpdateUserPoolClient",

                "groundtruthlabeling:DescribeConsoleJob",
                "groundtruthlabeling:ListDatasetObjects",
                "groundtruthlabeling:RunFilterOrSampleManifestJob",
                "groundtruthlabeling:RunGenerateManifestByCrawlingJob",

                "lambda:InvokeFunction",
                "lambda:ListFunctions",

                "s3:GetObject",
                "s3:PutObject",
```

```
                "s3:SelectObjectContent"
            ],
            "Resource": "*"
        }
    ]
}
```

使用 Amazon Augmented AI（预览版）控制台所需的权限

要使用 Augmented AI 控制台，您需要授予对其他资源的权限。使用以下权限策略授予这些权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:*Algorithm",
                "sagemaker:*Algorithms",
                "sagemaker:*App",
                "sagemaker:*Apps",
                "sagemaker:*AutoMLJob",
                "sagemaker:*AutoMLJobs",
                "sagemaker:*CodeRepositories",
                "sagemaker:*CodeRepository",
                "sagemaker:*CompilationJob",
                "sagemaker:*CompilationJobs",
                "sagemaker:*Endpoint",
                "sagemaker:*EndpointConfig",
                "sagemaker:*EndpointConfigs",
                "sagemaker:*EndpointWeightsAndCapacities",
                "sagemaker:*Endpoints",
                "sagemaker:*Environment",
                "sagemaker:*EnvironmentVersion",
                "sagemaker:*EnvironmentVersions",
                "sagemaker:*Environments",
                "sagemaker:*Experiment",
                "sagemaker:*Experiments",
                "sagemaker:*FlowDefinitions",
                "sagemaker:*HumanLoop",
                "sagemaker:*HumanLoops",
                "sagemaker:*HumanTaskUi",
                "sagemaker:*HumanTaskUis",
                "sagemaker:*HyperParameterTuningJob",
                "sagemaker:*HyperParameterTuningJobs",
                "sagemaker:*LabelingJob",
                "sagemaker:*LabelingJobs",
                "sagemaker:*Metrics",
                "sagemaker:*Model",
                "sagemaker:*ModelPackage",
                "sagemaker:*ModelPackages",
                "sagemaker:*Models",
                "sagemaker:*MonitoringExecutions",
                "sagemaker:*MonitoringSchedule",
                "sagemaker:*MonitoringSchedules",
                "sagemaker:*NotebookInstance",
                "sagemaker:*NotebookInstanceLifecycleConfig",
                "sagemaker:*NotebookInstanceLifecycleConfigs",
                "sagemaker:*NotebookInstanceUrl",
                "sagemaker:*NotebookInstances",
                "sagemaker:*ProcessingJob",
                "sagemaker:*ProcessingJobs",
                "sagemaker:*RenderUiTemplate",
                "sagemaker:*Search",
```

```
            "sagemaker:*SearchSuggestions",
            "sagemaker:*Tags",
            "sagemaker:*TrainingJob",
            "sagemaker:*TrainingJobs",
            "sagemaker:*TransformJob",
            "sagemaker:*TransformJobs",
            "sagemaker:*Trial",
            "sagemaker:*TrialComponent",
            "sagemaker:*TrialComponents",
            "sagemaker:*Trials",
            "sagemaker:*Workteam",
            "sagemaker:*Workteams"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "sagemaker:*FlowDefinition"
        ],
        "Resource": "*",
        "Condition": {
            "StringEqualsIfExists": {
                "sagemaker:WorkteamType": [
                    "private-crowd",
                    "vendor-crowd"
                ]
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "application-autoscaling:DeleteScalingPolicy",
            "application-autoscaling:DeleteScheduledAction",
            "application-autoscaling:DeregisterScalableTarget",
            "application-autoscaling:DescribeScalableTargets",
            "application-autoscaling:DescribeScalingActivities",
            "application-autoscaling:DescribeScalingPolicies",
            "application-autoscaling:DescribeScheduledActions",
            "application-autoscaling:PutScalingPolicy",
            "application-autoscaling:PutScheduledAction",
            "application-autoscaling:RegisterScalableTarget",
            "aws-marketplace:ViewSubscriptions",
            "cloudwatch:DeleteAlarms",
            "cloudwatch:DescribeAlarms",
            "cloudwatch:GetMetricData",
            "cloudwatch:GetMetricStatistics",
            "cloudwatch:ListMetrics",
            "cloudwatch:PutMetricAlarm",
            "cloudwatch:PutMetricData",
            "codecommit:BatchGetRepositories",
            "codecommit:CreateRepository",
            "codecommit:GetRepository",
            "codecommit:ListBranches",
            "codecommit:ListRepositories",
            "cognito-idp:AdminAddUserToGroup",
            "cognito-idp:AdminCreateUser",
            "cognito-idp:AdminDeleteUser",
            "cognito-idp:AdminDisableUser",
            "cognito-idp:AdminEnableUser",
            "cognito-idp:AdminRemoveUserFromGroup",
            "cognito-idp:CreateGroup",
            "cognito-idp:CreateUserPool",
            "cognito-idp:CreateUserPoolClient",
            "cognito-idp:CreateUserPoolDomain",
```

```
                "cognito-idp:DescribeUserPool",
                "cognito-idp:DescribeUserPoolClient",
                "cognito-idp:ListGroups",
                "cognito-idp:ListIdentityProviders",
                "cognito-idp:ListUserPoolClients",
                "cognito-idp:ListUserPools",
                "cognito-idp:ListUsers",
                "cognito-idp:ListUsersInGroup",
                "cognito-idp:UpdateUserPool",
                "cognito-idp:UpdateUserPoolClient",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:CreateVpcEndpoint",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:CreateRepository",
                "ecr:Describe*",
                "ecr:GetAuthorizationToken",
                "ecr:GetDownloadUrlForLayer",
                "elastic-inference:Connect",
                "elasticfilesystem:DescribeFileSystems",
                "elasticfilesystem:DescribeMountTargets",
                "fsx:DescribeFileSystems",
                "glue:CreateJob",
                "glue:DeleteJob",
                "glue:GetJob",
                "glue:GetJobRun",
                "glue:GetJobRuns",
                "glue:GetJobs",
                "glue:ResetJobBookmark",
                "glue:StartJobRun",
                "glue:UpdateJob",
                "groundtruthlabeling:*",
                "iam:ListRoles",
                "kms:DescribeKey",
                "kms:ListAliases",
                "lambda:ListFunctions",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "logs:PutLogEvents",
                "sns:ListTopics"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:DescribeResourcePolicies",
                "logs:GetLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:UpdateLogDelivery"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:SetRepositoryPolicy",
                "ecr:CompleteLayerUpload",
                "ecr:BatchDeleteImage",
                "ecr:UploadLayerPart",
                "ecr:DeleteRepositoryPolicy",
                "ecr:InitiateLayerUpload",
                "ecr:DeleteRepository",
                "ecr:PutImage"
            ],
            "Resource": "arn:aws:ecr:*:*:repository/*sagemaker*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:GitPull",
                "codecommit:GitPush"
            ],
            "Resource": [
                "arn:aws:codecommit:*:*:*sagemaker*",
                "arn:aws:codecommit:*:*:*SageMaker*",
                "arn:aws:codecommit:*:*:*Sagemaker*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:CreateSecret"
            ],
            "Resource": [
                "arn:aws:secretsmanager:*:*:secret:AmazonSageMaker-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:ResourceTag/SageMaker": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "robomaker:CreateSimulationApplication",
                "robomaker:DescribeSimulationApplication",
                "robomaker:DeleteSimulationApplication"
```

```
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "robomaker:CreateSimulationJob",
                "robomaker:DescribeSimulationJob",
                "robomaker:CancelSimulationJob"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:AbortMultipartUpload",
                "s3:GetBucketCors",
                "s3:PutBucketCors"
            ],
            "Resource": [
                "arn:aws:s3:::*SageMaker*",
                "arn:aws:s3:::*Sagemaker*",
                "arn:aws:s3:::*sagemaker*",
                "arn:aws:s3:::*aws-glue*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "s3:ExistingObjectTag/SageMaker": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:function:*SageMaker*",
                "arn:aws:lambda:*:*:function:*sagemaker*",
                "arn:aws:lambda:*:*:function:*Sagemaker*",
                "arn:aws:lambda:*:*:function:*LabelingFunction*"
            ]
```

```
        },
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/aws-service-role/sagemaker.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "sagemaker.application-autoscaling.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "robomaker.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:Subscribe",
                "sns:CreateTopic"
            ],
            "Resource": [
                "arn:aws:sns:*:*:*SageMaker*",
                "arn:aws:sns:*:*:*Sagemaker*",
                "arn:aws:sns:*:*:*sagemaker*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "sagemaker.amazonaws.com",
                        "glue.amazonaws.com",
                        "robomaker.amazonaws.com",
                        "states.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上完成此操作或者以编程方式使用 Amazon CLI 或 Amazon API 所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

的控制创建 SageMaker 使用条件键的资源

控制细粒度访问，以允许创建 SageMaker 通过使用 SageMaker 特定于的条件键来进行资源。有关在 IAM 策略中使用条件键的信息，请参阅。IAM JSON 策略元素：Condition中的IAM 用户指南.

条件密钥以及相关 API 操作以及相关文档的链接列于SageMaker 的条件键中的IAM 用户指南.

以下示例显示了如何使用 SageMaker 控制访问的条件键。

主题

- 控制对 的访问 SageMaker 使用文件系统条件键 (p. 2706)
- 限制对特定 VPC 的训练 (p. 2708)
- 限制对 Ground Truth 标记作业和 Amazon A2I 人工审查工作流的人力类型的访问 (p. 2709)
- 强制加密输入数据 (p. 2710)
- 强制加密笔记本实例存储卷 (p. 2710)
- 为训练作业强制进行网络隔离 (p. 2711)
- 强制为训练作业实施特定实例类型 (p. 2711)
- 强制为训练作业实施特定的 EI 加速器 (p. 2711)
- 在创建笔记本实例时强制禁用 Internet 访问和根访问 (p. 2712)

控制对 的访问 SageMaker 使用文件系统条件键

SageMaker 训练提供了安全的基础设施供训练算法在其中运行，但在某些情况下，您可能希望增加防御深度。例如，您可以最大程度地降低在算法中运行不受信任的代码的风险，或者您在组织中具有特定的安全性要求。对于这些场景，您可以在 IAM 策略的 Condition 元素中使用特定于服务的条件键，以将用户范围缩小到特定文件系统、目录、访问模式（读写，只读）和安全组。

主题

- 将 IAM 用户限制为特定目录和访问模式 (p. 2707)

- 将 IAM 用户限制为特定文件系统 (p. 2707)

将 IAM 用户限制为特定目录和访问模式

以下策略限制 IAM 用户使用`/sagemaker/xgboost-dm/train`和`/sagemaker/xgboost-dm/validation`用于 EFS 文件系统的目录`ro`（只读）AccessMode:

> **Note**
>
> 当允许目录时，训练算法也可以访问其所有子目录。POSIX 权限会被忽略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessToElasticFileSystem",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateTrainingJob",
                "sagemaker:CreateHyperParameterTuningJob"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sagemaker:FileSystemId": "fs-12345678",
                    "sagemaker:FileSystemAccessMode": "ro",
                    "sagemaker:FileSystemType": "EFS",
                    "sagemaker:FileSystemDirectoryPath": "/sagemaker/xgboost-dm/train"
                }
            }
        },
        {
            "Sid": "AccessToElasticFileSystemValidation",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateTrainingJob",
                "sagemaker:CreateHyperParameterTuningJob"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sagemaker:FileSystemId": "fs-12345678",
                    "sagemaker:FileSystemAccessMode": "ro",
                    "sagemaker:FileSystemType": "EFS",
                    "sagemaker:FileSystemDirectoryPath": "/sagemaker/xgboost-dm/validation"
                }
            }
        }
    ]
}
```

将 IAM 用户限制为特定文件系统

要防止使用用户空间客户端的恶意算法直接访问您账户中的任何文件系统，您可以通过允许来自特定安全组的入口来限制网络流量。在以下示例中，IAM 用户只能使用指定的安全组访问文件系统：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessToLustreFileSystem",
            "Effect": "Allow",
```

```
            "Action": [
                "sagemaker:CreateTrainingJob",
                "sagemaker:CreateHyperParameterTuningJob"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sagemaker:FileSystemId": "fs-12345678",
                    "sagemaker:FileSystemAccessMode": "ro",
                    "sagemaker:FileSystemType": "FSxLustre",
                    "sagemaker:FileSystemDirectoryPath": "/fsx/sagemaker/xgboost/train"
                },
                "ForAllValues:StringEquals": {
                    "sagemaker:VpcSecurityGroupIds": [
                        "sg-12345678"
                    ]
                }
            }
        }
    ]
}
```

虽然上述示例可以将算法限制为特定文件系统，但它不会阻止算法使用用户空间客户端访问该文件系统中的任何目录。要缓解这种情况，您可以：

- 确保文件系统仅包含您信任的 IAM 用户访问的数据。
- 创建 IAM 角色，限制您的 IAM 用户使用已批准的 ECR 存储库中的算法启动训练作业

有关如何将角色与 SageMaker 配合使用的更多信息，请参阅SageMaker 角色.

限制对特定 VPC 的训练

限制Amazon用户从 Amazon VPC 中创建训练作业。在 VPC 中创建训练作业时，您可以使用 VPC 流日志来监控进出训练集群的所有流量。有关使用 VPC 流日志的信息，请参阅VPC 流日志中的Amazon Virtual Private Cloud 用户指南.

以下策略强制由调用的 IAM 用户创建训练作业：`CreateTrainingJob`来自 VPC 内部：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowFromVpc",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateTrainingJob",
                "sagemaker:CreateHyperParameterTuningJob"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "sagemaker:VpcSubnets": ["subnet-a1234"],
                    "sagemaker:VpcSecurityGroupIds": ["sg12345", "sg-67890"]
                },
                "Null": {
                    "sagemaker:VpcSubnets": "false",
                    "sagemaker:VpcSecurityGroupIds": "false"
                }
            }
        }

    ]
```

```
}
```

限制对 Ground Truth 标记作业和 Amazon A2I 人工审查工作流的人力类型的访问

亚马逊 SageMaker Ground Truth 和亚马逊 Augmented AI 工作团队分为三个之一劳动力类型：公有（使用 Amazon Mechanical Turk）、专用和供应商。要使用这些类型之一或工作组 ARN 限制 IAM 用户对特定工作组的访问，请使用`sagemaker:WorkteamType`和/或`sagemaker:WorkteamArn`条件键。对于 `sagemaker:WorkteamType` 条件键，请使用字符串条件运算符。对于 `sagemaker:WorkteamArn` 条件键，请使用 Amazon 资源名称 (ARN) 条件运算符。如果用户尝试通过受限制的工作组来创建标记作业，SageMaker 返回访问被拒绝错误。

以下策略演示了将 `sagemaker:WorkteamType` 和 `sagemaker:WorkteamArn` 条件键与适当的条件运算符和有效条件值结合使用的不同方法。

以下示例使用 `sagemaker:WorkteamType` 条件键和 `StringEquals` 条件运算符一起限制对公有工作组的访问。它接受以下格式的条件值：*`workforcetype`*`-crowd`，其中 *`workforcetype`* 可以等于 `public`、`private` 或 `vendor`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RestrictWorkteamType",
            "Effect": "Deny",
            "Action": "sagemaker:CreateLabelingJob",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sagemaker:WorkteamType": "public-crowd"
                }
            }
        }
    ]
}
```

以下策略演示如何使用 `sagemaker:WorkteamArn` 条件键限制对公有工作组的访问。第一个策略演示如何将该条件键与工作组 ARN 的有效 IAM 正则表达式变量和`ArnLike`条件运算符。第二个策略演示如何将该条件键与 `ArnEquals` 条件运算符和工作组 ARN 一起使用。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RestrictWorkteamType",
            "Effect": "Deny",
            "Action": "sagemaker:CreateLabelingJob",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "sagemaker:WorkteamArn": "arn:aws:sagemaker:*:*:workteam/public-crowd/
*"
                }
            }
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "RestrictWorkteamType",
            "Effect": "Deny",
            "Action": "sagemaker:CreateLabelingJob",
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "sagemaker:WorkteamArn": "arn:aws:sagemaker:us-
west-2:394669845002:workteam/public-crowd/default"
                }
            }
        }
    ]
}
```

强制加密输入数据

以下策略限制 IAM 用户指定Amazon KMS在创建训练、超参数优化和标记作业时，使用`sagemaker:VolumeKmsKey`条件键：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceEncryption",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateTrainingJob",
                "sagemaker:CreateHyperParameterTuningJob",
                "sagemaker:CreateLabelingJob",
                "sagemaker:CreateFlowDefiniton"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "sagemaker:VolumeKmsKey": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
                }
            }
        }

     ]
}
```

强制加密笔记本实例存储卷

以下策略限制 IAM 用户指定Amazon KMS密钥，用于在创建或更新笔记本实例时使用`sagemaker:VolumeKmsKey`条件键：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceEncryption",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateNotebookInstance"
            ],
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "sagemaker:VolumeKmsKey": "*key/volume-kms-key-12345"
                }
```

```
            }
        }

    ]
}
```

为训练作业强制进行网络隔离

以下策略限制 IAM 用户在使用创建训练作业时启用网络隔离：`sagemaker:NetworkIsolation`条件键：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceIsolation",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateTrainingJob",
                "sagemaker:CreateHyperParameterTuningJob"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "sagemaker:NetworkIsolation": "true"
                }
            }
        }
    ]
}
```

强制为训练作业实施特定实例类型

以下策略限制 IAM 用户在使用创建训练作业时使用特定实例类型：`sagemaker:InstanceTypes`条件键：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceInstanceType",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateTrainingJob",
                "sagemaker:CreateHyperParameterTuningJob"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringLike": {
                    "sagemaker:InstanceTypes": ["ml.c5.*"]
                }
            }
        }

    ]
}
```

强制为训练作业实施特定的 EI 加速器

当创建或更新笔记本实例以及通过使用创建终端节点配置时，以下策略限制 IAM 用户使用特定的弹性推理 (EI) 加速器（如果提供了加速器）。`sagemaker:AcceleratorTypes`条件键：

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceAcceleratorType",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateNotebookInstance",
                "sagemaker:UpdateNotebookInstance",
                "sagemaker:CreateEndpointConfig"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "sagemaker:AcceleratorTypes": ["ml.eia1.medium"]
                }
            }
        }

     ]
}
```

在创建笔记本实例时强制禁用 Internet 访问和根访问

您可以禁用对笔记本实例的 Internet 访问和根访问，以帮助提高它们的安全性。有关控制对笔记本实例的根访问权限的信息，请参阅控制对 SageMaker 笔记本实例 (p. 2685)。有关禁用笔记本电脑实例的 Internet 访问的信息，请参阅将 VPC 中的笔记本实例 Connect 到外部资源 (p. 2783)。

以下策略要求 IAM 用户在创建实例时禁用网络访问，并在创建或更新笔记本实例时禁用根访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LockDownCreateNotebookInstance",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreateNotebookInstance"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sagemaker:DirectInternetAccess": "Disabled",
                    "sagemaker:RootAccess": "Disabled"
                },
                "Null": {
                  "sagemaker:VpcSubnets": "false",
                  "sagemaker:VpcSecurityGroupIds": "false"
                }
            }
        },
        {
            "Sid": "LockDownUpdateNotebookInstance",
            "Effect": "Allow",
            "Action": [
                "sagemaker:UpdateNotebookInstance"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sagemaker:RootAccess": "Disabled"
                }
            }
        }
     ]
```

```
}
```

控制对的访问 SageMaker 使用基于身份的策略 API

控制对的访问 SageMaker 对的 API 调用和调用 SageMaker 托管终端节点，请使用基于身份的 IAM 策略。

主题

- 限制访问 SageMaker API 和运行时间到您的 VPC 内的调用 (p. 2713)

限制访问 SageMaker API 和运行时间到您的 VPC 内的调用

如果您在 VPC 中设置了接口终端节点，则 VPC 之外的个人仍可以连接到 SageMaker 除非您附加 IAM 策略以将对来自 VPC 的调用的访问限制为仅有权访问您的您的所有用户和组，否则应用于 Internet 的 API 和运行时。 SageMaker 资源的费用。有关创建 VPC 接口终端节点的信息 SageMaker API 和运行时，请参阅连接到 SageMaker 通过 VPC 接口终端节点节点 (p. 2785).

> **Important**
>
> 如果您应用类似于以下内容之一的 IAM 策略，则用户无法访问指定的 SageMaker 通过控制台执行 API。

要将访问限制为仅限从您 VPC 中进行的连接，请创建 Amazon Identity and Access Management 策略，将访问限制为仅允许来自您 VPC 中调用。然后将该策略添加到每个Amazon Identity and Access Management 用于访问 SageMaker API 或运行时环境的。

> **Note**
>
> 此策略只允许连接到创建接口终端节点的子网中的调用方。

```
{
    "Id": "api-example-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableAPIAccess",
            "Effect": "Allow",
            "Action": [
                "sagemaker:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpc": "vpc-111bbaaa"
                }
            }
        }
    ]
}
```

如果要将 API 访问限制为仅使用接口终端节点进行的调用，请使用 `aws:SourceVpce` 条件键而不是 `aws:SourceVpc`：

```
{
    "Id": "api-example-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableAPIAccess",
            "Effect": "Allow",
            "Action": [
```

```
                "sagemaker:CreatePresignedNotebookInstanceUrl"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpce": [
                        "vpce-111bbccc",
                        "vpce-111bbddd"
                    ]
                }
            }
        }
    ]
}
```

限制访问 SageMaker 按 IP 地址划分的 API 和运行时调用

允许访问 SageMaker 仅从您指定列表中的 IP 地址进行 API 调用和运行时调用，请附加 IAM 策略，除非调用来自列表中的 IP 地址，否则该策略拒绝对该 API 的访问，除非调用来自列表中的 IP 地址。Amazon Identity and Access Management用于访问 API 或运行时的用户、组或角色。有关创建 IAM 策略的信息，请参阅创建 IAM 策略中的Amazon Identity and Access Management用户指南. 要指定您希望可以访问 API 调用的 IP 地址的列表，请使用 `IpAddress` 条件运算符和 `aws:SourceIP` 条件上下文键。有关 IAM 条件运算符的信息，请参阅IAM JSON 策略元素：条件运算符中的Amazon Identity and Access Management用户指南. 有关 IAM 条件上下文键的更多信息，请参阅Amazon全局条件上下文键.

例如，以下策略仅允许从范围 `192.0.2.0-192.0.2.255` 和 `203.0.113.0-203.0.113.255` 内的 IP 地址中访问 `CreateTrainingJob`：

```
{
    "Version": "2012-10-17",
    "Statement": [

        {
            "Effect": "Allow",
            "Action": "sagemaker:CreateTrainingJob",
            "Resource": "*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
                }
            }
        }
    ]
}
```

按 IP 地址限制对笔记本实例的访问

要仅允许从您指定列表中的 IP 地址访问笔记本实例，请附加一个 IAM 策略，请附加拒绝访问的 IAM 策略。`CreatePresignedNotebookInstanceUrl`除非呼叫来自列表中的一个 IP 地址到每个Amazon Identity and Access Management用于访问笔记本实例的用户、组或角色。有关创建 IAM 策略的信息，请参阅创建 IAM 策略中的Amazon Identity and Access Management用户指南. 要指定您希望可以访问笔记本实例的 IP 地址列表，请使用 `IpAddress` 条件运算符和 `aws:SourceIP` 条件上下文键。有关 IAM 条件运算符的信息，请参阅IAM JSON 策略元素：条件运算符中的Amazon Identity and Access Management用户指南. 有关 IAM 条件上下文键的更多信息，请参阅Amazon全局条件上下文键.

例如，以下策略仅允许从范围 `192.0.2.0-192.0.2.255` 和 `203.0.113.0-203.0.113.255` 之间的 IP 地址访问笔记本实例：

```
{
    "Version": "2012-10-17",
    "Statement": [

        {
            "Effect": "Allow",
            "Action": "sagemaker:CreatePresignedNotebookInstanceUrl",
            "Resource": "*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
                }
            }
        }
    ]
}
```

该策略限制对 `CreatePresignedNotebookInstanceUrl` 的调用以及调用返回的 URL 的访问。该策略还限制在控制台中打开笔记本实例的访问权限，并针对每个 HTTP 请求强制执行。 WebSocket 尝试连接到笔记本实例的帧。

Note

在以下情况下，使用此方法按 IP 地址筛选不兼容：连接到 SageMaker 通过 VPC 接口终端节点。. 有关在通过 VPC 接口终端节点连接时限制对笔记本实例访问的信息，请参阅通过 VPC 接口终端节点连接到笔记本实例 (p. 2790)。

控制对 的访问 SageMaker 使用标签资源

控制对组的访问 SageMaker 通过将标签附加到资源并指定`ResourceTag`IAM 策略中的条件。

Note

基于标签的策略无法限制以下 API 调用：

- ListAlgorithms
- ListCodeRepositories
- ListCompilationJobs
- ListEndpointConfigs
- ListEndpoints
- ListFlowDefinitions
- ListHumanTaskUis
- ListHyperparameterTuningJobs
- ListLabelingJobs
- ListLabelingJobsForWorkteam
- ListModelPackages
- ListModels
- ListNotebookInstanceLifecycleConfigs
- ListNotebookInstances
- ListSubscribedWorkteams
- ListTags
- ListProcessingJobs
- ListTrainingJobs

- ListTrainingJobsForHyperParameterTuningJob
- ListTransformJobs
- ListWorkteams
- 搜索

例如，假设您定义了两个不同的 IAM 组，名为DevTeam1和DevTeam2中的Amazonaccount. 假设您已创建了 10 个笔记本实例，其中 5 个用于一个项目，还有 5 个用于第二个项目。您希望允许 DevTeam1 的成员在用于第一个项目的笔记本实例上进行 API 调用，DevTeam2 的成员在用于第二个项目的笔记本实例上进行 API 调用。

控制对 API 调用的访问（示例）

1. 将具有键 Project 和值 A 的标签添加到用于第一个项目的笔记本实例。有关向添加标签的信息 SageMaker 资源，请参阅AddTags.
2. 将具有键 Project 和值 B 的标签添加到用于第二个项目的笔记本实例。
3. 使用ResourceTag拒绝对用于第二个项目的笔记本实例的访问，并将该策略附加到DevTeam1. 以下策略示例拒绝具有键为 Project 且值为 B 的标签的任何笔记本实例上的所有 API 调用：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sagemaker:*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": "sagemaker:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "sagemaker:ResourceTag/Project": "B"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "sagemaker:AddTags",
        "sagemaker:DeleteTags"
      ],
      "Resource": "*"
    }
  ]
}
```

有关创建 IAM 策略并将其附加到身份的信息，请参阅使用策略控制访问中的Amazon Identity and Access Management用户指南.

4. 使用ResourceTag拒绝对用于第一个项目的笔记本实例的访问，并将该策略附加到DevTeam2. 以下策略示例拒绝具有键为 Project 且值为 A 的标签的任何笔记本实例上的所有 API 调用：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sagemaker:*",
      "Resource": "*"
```

```
    },
    {
      "Effect": "Deny",
      "Action": "sagemaker:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "sagemaker:ResourceTag/Project": "A"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "sagemaker:AddTags",
        "sagemaker:DeleteTags"
      ],
      "Resource": "*"
    }
  ]
}
```

要求 API 调用存在或不存在标签

使用要求存在或不存在特定标签或特定标签值。`RequestTag`IAM 策略中的条件密钥。例如，如果您希望要求 IAM 组的任何成员创建的每个终端节点都使用带有密钥的标签创建。`environment`和值`dev`，创建策略，如下所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sagemaker:CreateEndpoint",
            "Resource": "arn:aws:sagemaker:*:*:endpoint/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": "dev"
                }
            }
        }
    ]
}
```

跨服务困惑副预防

混淆代理问题是一个安全问题，即没有执行操作权限的实体可能会迫使更具权限的实体执行该操作。在 Amazon 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为了防止这种情况，Amazon 提供可帮助您保护所有服务的服务委托人数据的工具，这些服务委托人有权限访问账户中的资源。

使用全局条件键限制权限

我们建议使用`aws:SourceArn`和`aws:SourceAccount`资源策略中的全局条件密钥，以限制亚马逊的权限 SageMaker 为资源提供了另一项服务。如果你同时使用全局条件密钥和`aws:SourceArn`值包含账户 ID，`aws:SourceAccount`值和中的账户`aws:SourceArn`值在同一策略语句中使用相同的账户 ID。如果您只希望将一个资源与跨服务访问相关联，请使用 `aws:SourceArn`。如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用 `aws:SourceAccount`。

防范混淆代理问题最有效的方法是使用aws:SourceArn具有资源的完整 ARN 的全局条件密钥。如果不知道资源的完整 ARN，或者正在指定多个资源，请使用aws:SourceArn带通配符的全局条件键 (*) 对于 ARN 的未知部分。例如，arn:aws:sagemaker:*:*123456789012*:*。

以下示例说明如何使用aws:SourceArn和aws:SourceAccount中的全局条件键 SageMaker 以防范混淆代理人问题。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "sagemaker.amazonaws.com"
    },
    # Specify an action and resource policy for another service
    "Action": "service:ActionName",
    "Resource": [
      "arn:aws:service:::ResourceName/*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:partition:sagemaker:region:123456789012:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

跨服务混淆副预防 SageMaker 边缘管理器

以下示例说明如何使用aws:SourceArn防止跨服务混淆副问题的全局条件关键 SageMaker 通过帐号创建的边缘管理器*123456789012*在地区*us-west-2*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sagemaker:us-west-2:123456789012:*"
        }
      }
    }
  ]
}
```

您可以替换aws:SourceArn在此模板中包含一个特定打包作业的完整 ARN 以进一步限制权限。

跨服务混淆副预防 SageMaker 推理

以下示例说明如何使用aws:SourceArn防止跨服务混淆副问题的全局条件关键 SageMaker 实时、无服务器, 和异步的推理。请注意，因为账号是账号的一部分aws:SourceArn值，您无需指定aws:SourceAccount值。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "sagemaker.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:sagemaker:us-west-2:123456789012:*"
      }
    }
  }
}
```

不要更换aws:SourceArn在此模板中包含特定模型或端点的完整 ARN。ARN 必须采用以上提供的格式。ARN 模板中的星号不代表通配符，不应更改。

跨服务混淆副预防 SageMaker Marketplace

以下示例说明如何使用aws:SourceArn防止跨服务混淆副问题的全局条件关键 SageMaker 通过账号创建的 Marketplace 资源*123456789012*在地区*us-west-2*. 请注意，由于账号在 ARN 中，因此您无需指定aws:SourceAccount值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sagemaker:us-west-2:123456789012:*"
        }
      }
    }
  ]
}
```

不要更换aws:SourceArn在此模板中包含特定算法或模型包的完整 ARN。ARN 必须采用以上提供的格式。ARN 模板中的星号确实代表通配符，涵盖了验证步骤中的所有训练作业、模型和批量转换作业，以及发布到的算法和模型包 SageMaker Marketplace。

跨服务混淆副预防 SageMaker 新

以下示例说明如何使用aws:SourceArn防止跨服务混淆副问题的全局条件关键 SageMaker 由帐号创建的 Neo 编译工作*123456789012*在地区*us-west-2*. 请注意，由于账号在 ARN 中，因此您无需指定aws:SourceAccount值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sagemaker:us-west-2:123456789012:compilation-job/*"
        }
      }
    }
  ]
}
```

您可以替换aws:SourceArn在此模板中包含一个特定编译作业的完整 ARN 以进一步限制权限。

跨服务混淆副预防 SageMaker 工作室

以下示例说明如何使用aws:SourceArn防止跨服务混淆副问题的全局条件关键 SageMaker 通过帐号创建的工作室*123456789012*在地区*us-west-2*. 请注意，因为账号是账号的一部分aws:SourceArn值，您无需指定aws:SourceAccount值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sagemaker:us-west-2:123456789012:*"
        }
      }
    }
  ]
}
```

不要更换aws:SourceArn在此模板中包含特定 Studio 应用程序、用户配置文件或域的完整 ARN。ARN 必须采用以上提供的格式。ARN 模板中的星号不代表通配符，不应更改。

跨服务混淆副预防 SageMaker 管道

以下示例说明如何使用aws:SourceArn防止跨服务混淆副问题的全局条件关键SageMaker 管道使用来自一个或多个管道的管道执行记录。请注意，由于账号在 ARN 中，因此您无需指定aws:SourceAccount值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:partition:sagemaker:region:123456789012:pipeline/
mypipeline/*"
        }
      }
    }
  ]
```

```
}
```

不要更换aws:SourceArn在此模板中包含特定管道执行的完整 ARN。ARN 必须采用以上提供的格式。这些区域有：partition占位符应该指定Amazon商业分区（aws）或者Amazon在中国分区 (aws-cn)，具体取决于管道的执行位置。同样，regionARN 中的占位符可以是任意有效地区哪里 SageMaker 管道可用。

ARN 模板中的星号确实代表通配符，涵盖名为的管道的所有管道执行mypipeline. 如果您想允许AssumeRole账户中所有管道的权限123456789012而不是一个特定的管道，那么aws:SourceArn将是arn:aws:sagemaker:*:123456789012:pipeline/*.

跨服务混淆副预防 SageMaker 处理任务

以下示例说明如何使用aws:SourceArn防止跨服务混淆副问题的全局条件关键 SageMaker 处理由账号创建的作业*123456789012*在地区*us-west-2*. 请注意，由于账号在 ARN 中，因此您无需指定aws:SourceAccount值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sagemaker:us-west-2:123456789012:processing-job/*"
        }
      }
    }
  ]
}
```

您可以替换aws:SourceArn在此模板中包含一个特定处理作业的完整 ARN 以进一步限制权限。

跨服务混淆副预防 SageMaker 训练任务

以下示例说明如何使用aws:SourceArn防止跨服务混淆副问题的全局条件关键 SageMaker 通过账号创建的培训职位*123456789012*在地区*us-west-2*. 请注意，由于账号在 ARN 中，因此您无需指定aws:SourceAccount值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sagemaker:us-west-2:123456789012:training-job/*"
        }
      }
    }
  ]
}
```

您可以替换aws:SourceArn在此模板中包含一个特定培训作业的完整 ARN 以进一步限制权限。

后续步骤

有关管理执行角色的更多信息，请参阅SageMaker 角色.

SageMaker 角色

作为托管服务，亚马逊 SageMaker 代表你执行操作Amazon由 SageMaker 管理的硬件。 SageMaker 仅执行用户允许的操作。

一个 SageMaker 用户可为这些权限授予 IAM 角色 (称为执行角色)。

要创建和使用本地可用的执行角色，可以使用以下过程。

获取执行角色

您可以通过以下方式找到 IAM 执行角色：

来自笔记本

当你在内部运行笔记本 SageMaker 来自的 SageMaker 控制台或 SageMaker Studio) 你可以使用以下代码访问执行角色：

```
sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()
```

> **Note**
>
> 只有在 SageMaker 中运行笔记本时，执行角色才可用。如果你跑get_execution_role在不在 SageMaker 上的笔记本中，预计会出现 “区域” 错误。

从 SageMaker 控制台

UNDER笔记本 > 笔记本实例中，选择笔记本。ARN 在权限和加密部分。

创建执行角色

使用以下过程使用 IAM 托管策略创建执行角色，AmazonSageMakerFullAccess挂载到。如果您的使用案例需要更精细的权限，请使用本页上的其他部分创建满足业务需求的执行角色。

> **Important**
>
> IAM 托管式策略，AmazonSageMakerFullAccess，在以下过程中使用仅授予执行角色对存储桶或对象执行某些 Amazon S3 操作的权限SageMaker、Sagemaker、sagemaker，或者aws-glue在名字里。要了解如何向执行角色添加额外策略以授予其他 Amazon S3 存储桶和对象的访问权限，请参阅将其他 Amazon S3 权限添加到 SageMaker 执行角色 (p. 2723).

如欲创建新角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. Select角色然后选择创建角色.
3. SelectSageMaker.
4. Select后续：Permissions (下一步：权限)。
5. IAM 托管式策略，AmazonSageMakerFullAccess自动附加到此角色。要查看此策略中包含的权限，请选择策略名称旁边的侧向箭头。Select后续：标签。
6. （可选）添加标签并选择后续：审核。
7. 在下面的文本字段中为角色命名Role name (角色名称)然后选择创建角色.

8. 在存储库的角色在 IAM 控制台的部分中，选择您刚刚创建的角色。如果需要，请使用文本框使用您在步骤 7 中输入的角色名称搜索角色。
9. 在角色摘要页面上，记录 ARN。

使用角色的已知 ARN，您可以在本地或在 SageMaker 上运行笔记本时以编程方式检查角色。Replace`RoleName`用你已知的 ARN：

```
try:
    role = sagemaker.get_execution_role()
except ValueError:
    iam = boto3.client('iam')
    role = iam.get_role(RoleName='AmazonSageMaker-ExecutionRole-20201200T100000')['Role']
['Arn']
```

将其他 Amazon S3 权限添加到 SageMaker 执行角色

当您使用 SageMaker 功能，其中包含 Amazon S3 中的资源，例如输入数据、您在请求中指定的执行角色（例如`CreateTrainingJob`) 用于访问这些资源。

如果您附加了 IAM 托管策略，`AmazonSageMakerFullAccess`，对于执行角色，则该角色有权对存储桶或对象执行某些 Amazon S3 操作`SageMaker`、`Sagemaker`、`sagemaker`，或者`aws-glue`在名字里。它还有权对任何 Amazon S3 资源执行以下操作：

```
"s3:CreateBucket",
"s3:GetBucketLocation",
"s3:ListBucket",
"s3:ListAllMyBuckets",
"s3:GetBucketCors",
"s3:PutBucketCors"
```

要向执行角色授予访问 Amazon S3 中一个或多个特定存储桶的权限，您可以向该角色附加类似以下内容的策略。此策略授予 IAM 角色执行以下所有操作的权限：`AmazonSageMakerFullAccess`允许但限制对存储桶的此访问*`####### 1`*和*`DOC-#-BUCKET2`*. 有关具体的信息，请参阅安全文档 SageMaker 您正在使用的功能来了解有关该功能所需的 Amazon S3 权限的更多信息。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET2/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetBucketCors",
                "s3:PutBucketCors"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketAcl",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET1",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET2"
            ]
        }
    ]
}
```

传递角色

诸如在服务之间传递角色之类的操作是 SageMaker 中的常见功能。您可以在上找到更多详细信息SageMaker 的操作、资源和条件键中的IAM 用户指南.

你通过了角色 (`iam:PassRole`) 进行这些 API 调用时：`CreateAutoMLJob`、`CreateCompilationJob`、`CreateDomain`、`CreateFlowDefiniton`、`CreateHyperPa`和`UpdateImage`.

您将以下信任策略附加到授予的 IAM 角色。 SageMaker 委托人权限代入该角色，并且对于所有执行角色均相同：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "sagemaker.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

您需要授予该角色的权限有所不同，具体取决于您所调用的 API。以下几节解释了这些权限。

> **Note**
>
> 您可以使用，而不用通过创建权限策略来管理权限。Amazon-托管`AmazonSageMakerFullAccess`权限策略。此策略中的权限相当广泛，以允许您在 SageMaker 中执行您可能希望执行的任何操作。有关此策略的列表，包括有关添加许多权限的原因的信息，请参阅 Amazon托管策略：AmazonSageMakerFullAccess (p. 2755)。如果您更愿意创建自定义策略和管理权限以将权限限定于您需要使用执行角色执行的操作，请参阅以下主题。

有关 IAM 角色的更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

主题

- CreateAutoMLJob API：执行角色权限 (p. 2725)
- CreateDomain API：执行角色权限 (p. 2726)
- CreateImage 并更新 Image API：执行角色权限 (p. 2727)
- CreateNotebookInstance API：执行角色权限 (p. 2727)
- CreateHyperParameterTuningJob API：执行角色权限 (p. 2730)

- CreateProcessingJob API：执行角色权限 (p. 2733)
- CreateTrainingJob API：执行角色权限 (p. 2735)
- CreateModel API：执行角色权限 (p. 2737)

CreateAutoMLJob API：执行角色权限

对于可以传入的执行角色`CreateAutoMLJob`API 请求，您可以将以下最低权限策略附加到该角色：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "sagemaker.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:DescribeEndpointConfig",
                "sagemaker:DescribeModel",
                "sagemaker:InvokeEndpoint",
                "sagemaker:ListTags",
                "sagemaker:DescribeEndpoint",
                "sagemaker:CreateModel",
                "sagemaker:CreateEndpointConfig",
                "sagemaker:CreateEndpoint",
                "sagemaker:DeleteModel",
                "sagemaker:DeleteEndpointConfig",
                "sagemaker:DeleteEndpoint",
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*"
        }
    ]
}
```

如果您为 AutoML 任务指定一个私有 VPC，请添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface",
```

```
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DeleteNetworkInterface",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
    ]
}
```

如果您的输入采用服务器端加密加密进行加密，则Amazon添加 KMS 托管密钥 (SSE-KMS)，添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt"
    ]
}
```

如果在 AutoML 作业的输出配置中指定一个 KMS 密钥，则添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
    "kms:Encrypt"
    ]
}
```

如果在 AutoML 作业的资源配置中指定一个卷 KMS 密钥，则添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
    "kms:CreateGrant"
    ]
}
```

CreateDomain API：执行角色权限

的执行角色IAM Identity Center当您通过 IAM 域名的域和用户/执行角色时需要以下权限Amazon KMS作为客户托管密钥`KmsKeyId`中的`CreateDomain`API 请求。这些权限是在`CreateApp`API 调用。

对于可以传入的执行角色`CreateDomain`API 请求，您可以将以下权限策略附加到该角色：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:region:account-id:key/kms-key-id"
        }
    ]
}
```

或者，如果在 KMS 策略中指定权限，您可以将以下策略附加到该角色：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::account-id:role/ExecutionRole"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        }
    ]
}
```

CreateImage 并更新 Image API：执行角色权限

对于可以传入的执行角色`CreateImage`要么`UpdateImage`API 请求，您可以将以下权限策略附加到该角色：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "sagemaker.amazonaws.com"
                }
            }
        }
    ]
}
```

CreateNotebookInstance API：执行角色权限

您向执行角色授予调用 `CreateNotebookInstance` API 的权限取决于您计划对笔记本实例执行的操作。如果您计划使用它来调用 SageMaker API 并在调用时传递相同的角色`CreateTrainingJob`和`CreateModel`API，将以下权限策略附加到角色：

```
{
    "Version": "2012-10-17",
```

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "sagemaker:*",
            "ecr:GetAuthorizationToken",
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability",
            "ecr:SetRepositoryPolicy",
            "ecr:CompleteLayerUpload",
            "ecr:BatchDeleteImage",
            "ecr:UploadLayerPart",
            "ecr:DeleteRepositoryPolicy",
            "ecr:InitiateLayerUpload",
            "ecr:DeleteRepository",
            "ecr:PutImage",
            "ecr:CreateRepository",
            "cloudwatch:PutMetricData",
            "cloudwatch:GetMetricData",
            "cloudwatch:GetMetricStatistics",
            "cloudwatch:ListMetrics",
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:DescribeLogStreams",
            "logs:PutLogEvents",
            "logs:GetLogEvents",
            "s3:CreateBucket",
            "s3:ListBucket",
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:PutObject",
            "s3:DeleteObject",
            "robomaker:CreateSimulationApplication",
            "robomaker:DescribeSimulationApplication",
            "robomaker:DeleteSimulationApplication",
            "robomaker:CreateSimulationJob",
            "robomaker:DescribeSimulationJob",
            "robomaker:CancelSimulationJob",
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeRouteTables",
            "elasticfilesystem:DescribeMountTargets"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "codecommit:GitPull",
            "codecommit:GitPush"
        ],
        "Resource": [
            "arn:aws:codecommit:*:*:*sagemaker*",
            "arn:aws:codecommit:*:*:*SageMaker*",
            "arn:aws:codecommit:*:*:*Sagemaker*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "iam:PassedToService": "sagemaker.amazonaws.com"
```

```
                }
            }
        }
    ]
}
```

要收紧权限，请通过限制使其仅限于特定 Amazon S3 和 Amazon ECR 资源。`"Resource": "*"`，如下所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:*",
                "ecr:GetAuthorizationToken",
                "cloudwatch:PutMetricData",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:GetLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "sagemaker.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::inputbucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::inputbucket/object1",
                "arn:aws:s3:::outputbucket/path",
                "arn:aws:s3:::inputbucket/object2",
                "arn:aws:s3:::inputbucket/object3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": [
                "arn:aws:ecr:region::repository/my-repo1",
                "arn:aws:ecr:region::repository/my-repo2",
                "arn:aws:ecr:region::repository/my-repo3"
            ]
        }
    ]
}
```

如果您计划访问其他资源（如 Amazon DynamoDB 或 Amazon Relational Database Service），请向此策略添加相关权限。

在上一个策略中，您按如下方式确定策略范围：

- 仅向您在 `s3:ListBucket` 请求中指定作为 `InputDataConfig.DataSource.S3DataSource.S3Uri` 的特定存储桶授予 `CreateTrainingJob` 权限。
- 按如下方式确定 `s3:GetObject` 、`s3:PutObject` 和 `s3:DeleteObject` 的权限范围：
  - 将范围限定为您在 `CreateTrainingJob` 请求中指定的以下值：

    `InputDataConfig.DataSource.S3DataSource.S3Uri`

    `OutputDataConfig.S3OutputPath`
  - 将范围限定为您在 `CreateModel` 请求中指定的以下值：

    `PrimaryContainer.ModelDataUrl`

    `SuplementalContainers.ModelDataUrl`
- 按如下方式确定 `ecr` 权限的范围：
  - 将范围限定为您在 `AlgorithmSpecification.TrainingImage` 请求中指定的 `CreateTrainingJob` 值。
  - 将范围限定为您在 `PrimaryContainer.Image` 请求中指定的 `CreateModel` 值：

`cloudwatch` 和 `logs` 操作适用于“*”资源。有关更多信息，请参阅 。CloudWatch 资源和操作在 Amazon 中 CloudWatch 用户指南。

CreateHyperParameterTuningJob API：执行角色权限

对于可在 `CreateHyperParameterTuningJob` API 请求中传递的执行角色，您可以将以下权限策略附加到该角色：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:PutObject",
```

```
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*"
        }
    ]
}
```

而不是指定`"Resource": "*"`，您可以将这些权限限制为特定的 Amazon S3 和 Amazon ECR 资源：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::inputbucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::inputbucket/object",
                "arn:aws:s3:::outputbucket/path"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:region::repository/my-repo"
        }
    ]
}
```

如果与超参数优化作业关联的训练容器需要访问其他数据源 (如 DynamoDB 或 Amazon RDS 资源)，则向此策略添加相关权限。

在上一个策略中，您按如下方式确定策略范围：

- 仅向您在 `s3:ListBucket` 请求中指定作为 `InputDataConfig.DataSource.S3DataSource.S3Uri` 的特定存储桶授予 `CreateTrainingJob` 权限。
- 仅向您在 `s3:GetObject` 请求的输入和输出数据配置中指定的以下对象授予 `s3:PutObject`和 `CreateHyperParameterTuningJob` 权限：

  `InputDataConfig.DataSource.S3DataSource.S3Uri`

  `OutputDataConfig.S3OutputPath`
- 将 Amazon ECR 权限限定为注册表路径 (`AlgorithmSpecification.TrainingImage`) 你在`CreateHyperParameterTuningJob`请求.

`cloudwatch` 和 `logs` 操作适用于“*”资源。有关更多信息，请参阅 。CloudWatch 资源和操作在 Amazon 中 CloudWatch 用户指南。

如果您为超参数优化作业指定一个私有 VPC，请添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DeleteNetworkInterface",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
    ]
}
```

如果您的输入采用服务器端加密加密进行加密，则Amazon添加 KMS 托管密钥 (SSE-KMS)，添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt"
    ]
}
```

如果在超参数优化作业的输出配置中指定一个 KMS 密钥，则添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
    "kms:Encrypt"
    ]
}
```

如果在超参数优化作业的资源配置中指定一个批量 KMS 密钥，则添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
    "kms:CreateGrant"
    ]
}
```

CreateProcessingJob API：执行角色权限

对于可在 `CreateProcessingJob` API 请求中传递的执行角色，您可以将以下权限策略附加到该角色：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*"
        }
    ]
}
```

而不是指定`"Resource": "*"`，您可以将这些权限限制为特定的 Amazon S3 和 Amazon ECR 资源：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::inputbucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::inputbucket/object",
                "arn:aws:s3:::outputbucket/path"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:region::repository/my-repo"
        }
    ]
}
```

如果CreateProcessingJob.AppSpecification.ImageUri如果需要访问其他数据源 (如 DynamoDB 或 Amazon RDS 资源)，则向此策略添加相关权限。

在上一个策略中，您按如下方式确定策略范围：

- 仅向您在 s3:ListBucket 请求中指定作为 ProcessingInputs 的特定存储桶授予 CreateProcessingJob 权限。
- 将 s3:GetObject 和 s3:PutObject 权限的范围限定在 CreateProcessingJob 请求中要在 ProcessingInputs 和 ProcessingOutputConfig 中下载或上传的对象。
- 将 Amazon ECR 权限限定为注册表路径 (AppSpecification.ImageUri) 你在CreateProcessingJob请求.

cloudwatch 和 logs 操作适用于“*”资源。有关更多信息，请参阅 。CloudWatch 资源和操作在 Amazon 中 CloudWatch 用户指南。

如果您为处理作业指定一个私有 VPC，请添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DeleteNetworkInterface",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
    ]
}
```

如果您的输入采用服务器端加密加密进行加密，则Amazon添加 KMS 托管密钥 (SSE-KMS)，添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt"
    ]
}
```

如果在处理作业的输出配置中指定一个 KMS 密钥，则添加以下权限：

```
{
```

```
    "Effect": "Allow",
    "Action": [
    "kms:Encrypt"
    ]
}
```

如果在处理作业的资源配置中指定一个批量 KMS 密钥，则添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
    "kms:CreateGrant"
    ]
}
```

CreateTrainingJob API：执行角色权限

对于可在 `CreateTrainingJob` API 请求中传递的执行角色，您可以将以下权限策略附加到该角色：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*"
        }
    ]
}
```

而不是指定`"Resource": "*"`，您可以将这些权限限制为特定的 Amazon S3 和 Amazon ECR 资源：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::inputbucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::inputbucket/object",
                "arn:aws:s3:::outputbucket/path"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:region::repository/my-repo"
        }
    ]
}
```

如果CreateTrainingJob.AlgorithSpecifications.TrainingImage如果需要访问其他数据源 (如 DynamoDB 或 Amazon RDS 资源)，则向此策略添加相关权限。

在上一个策略中，您按如下方式确定策略范围：

- 仅向您在 s3:ListBucket 请求中指定作为 InputDataConfig.DataSource.S3DataSource.S3Uri 的特定存储桶授予 CreateTrainingJob 权限。
- 仅向您在 s3:GetObject 请求的输入和输出数据配置中指定的以下对象授予 s3:PutObject和 CreateTrainingJob 权限：

  InputDataConfig.DataSource.S3DataSource.S3Uri

  OutputDataConfig.S3OutputPath
- 将 Amazon ECR 权限限定为注册表路径 (AlgorithmSpecification.TrainingImage) 你在CreateTrainingJob请求.

cloudwatch 和 logs 操作适用于“*”资源。有关更多信息，请参阅 。CloudWatch 资源和操作在 Amazon 中 CloudWatch 用户指南。

如果您为训练作业指定一个私有 VPC，请添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterface",
      "ec2:CreateNetworkInterfacePermission",
      "ec2:DeleteNetworkInterface",
      "ec2:DeleteNetworkInterfacePermission",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DescribeVpcs",
      "ec2:DescribeDhcpOptions",
      "ec2:DescribeSubnets",
```

```
      "ec2:DescribeSecurityGroups"
    ]
}
```

如果您的输入采用服务器端加密加密进行加密，则Amazon添加 KMS 托管密钥 (SSE-KMS)，添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt"
    ]
}
```

如果在训练作业的输出配置中指定一个 KMS 密钥，则添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
    "kms:Encrypt"
    ]
}
```

如果在训练作业的资源配置中指定一个批量 KMS 密钥，则添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
    "kms:CreateGrant"
    ]
}
```

CreateModel API：执行角色权限

对于可在 `CreateModel` API 请求中传递的执行角色，您可以将以下权限策略附加到该角色：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*"
        }
    ]
}
```

而不是指定`"Resource": "*"`，您可以将这些权限限制为特定的 Amazon S3 和 Amazon ECR 资源：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::inputbucket/object"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": [
                "arn:aws:ecr:region::repository/my-repo",
                "arn:aws:ecr:region::repository/my-repo"
             ]
        }
    ]
}
```

如果 `CreateModel.PrimaryContainer.Image` 需要访问其他数据源 (如 Amazon DynamoDB 或 Amazon RDS 资源)，则向此策略添加相关权限。

在上一个策略中，您按如下方式确定策略范围：

- 仅向您在 `PrimaryContainer.ModelDataUrl` 请求的 `CreateModel` 中指定的对象授予 S3 权限。
- 将 Amazon ECR 权限限定为您指定作为的特定注册表路径。`PrimaryContainer.Image`和`SecondaryContainer.Image`在`CreateModel`请求.

`cloudwatch` 和 `logs` 操作适用于“*”资源。有关更多信息，请参阅 。CloudWatch 资源和操作在 Amazon 中 CloudWatch 用户指南。

如果您为模型指定一个私有 VPC，请添加以下权限：

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DeleteNetworkInterface",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
```

```
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
    ]
}
```

亚马逊 SageMaker API 权限：操作、权限和资源参考

在设置访问控制和编写您可附加到 IAM 身份的权限策略 (基于身份的策略) 时，可以使用下作为参考。这些区域有：每个亚马逊 SageMaker API 操作、您可授予执行该操作的权限的对应操作以及Amazon您可以授予权限的资源。您可以在策略的 `Action` 字段中指定这些操作，并在策略的 `Resource` 字段中指定资源值。

> **Note**
>
> 除 `ListTags` API 外，资源级别限制在 `List-` 调用中不可用。任何调用 `List-` API 的用户都将看到账户中该类型的所有资源。

在亚马逊中表达状况 SageMaker 策略，您可以使用Amazon宽范围的条件键。有关 Amazon 范围内的键的完整列表，请参阅《IAM 用户指南》https://docs.amazonaws.cn/IAM/latest/UserGuide/reference_policies_elements.html#AvailableKeys中的可用键。

亚马逊 SageMaker API 操作和必需的操作权限

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
AddTags	sagemaker:AddTags	arn:aws:sagemaker:*region*:*account-id*:*
CreateApp	sagemaker:CreateApp	arn:aws:sagemaker:*region*:*account-id*:app/*domain-id*/*user-profile-name*/*app-type*/*appName*
CreateAppImageConfig	sagemaker:CreateAppImageConfig	arn:aws:sagemaker:*region*:*account-id*:app-image-config/*appImageConfigName*
CreateAutoMLJob	sagemaker:CreateAutoMLJob iam:PassRole 仅需要关联的以下权限ResourceConfig有指定VolumeKmsKeyId 而且关联的角色没有允许此操作的策略： kms:CreateGrant	arn:aws:sagemaker:*region*:*account-id*:automl-job/*autoMLJobName*
CreateDomain	sagemaker:CreateDomain iam:CreateServiceLinkedRole iam:PassRole 如果指定了 KMS 客户管理的密钥，则为必填项KmsKeyId： elasticfilesystem:CreateFileSystem kms:CreateGrant	arn:aws:sagemaker:*region*:*account-id*:domain/*domain-id*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
	kms:Decrypt kms:DescribeKey kms:GenerateDataKeyWithoutPlainText	
CreateEndpoint	sagemaker:CreateEndpoint kms:CreateGrant (仅当关联的 EndPointConfig 指定了 KmsKeyId 时才需要)	arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName* arn:aws:sagemaker:*region*:*account-id*:endpoint-config/*endpointConfigName*
CreateEndpointConfig	sagemaker:CreateEndpointConfig	arn:aws:sagemaker:*region*:*account-id*:endpoint-config/*endpointConfigName*
CreateFlowDefinition	sagemaker:CreateFlowDefinition iam:PassRole	arn:aws:sagemaker:*region*:*account-id*:flow-definition/*flowDefinitionName*
CreateHumanTaskUi	sagemaker:CreateHumanTaskUi	arn:aws:sagemaker:*region*:*account-id*:human-task-ui/*humanTaskUiName*
CreateInferenceRecommendationsJob	sagemaker:CreateInferenceRecommendationsJob iam:PassRole 仅当您指定加密密钥时，才需要以下权限： kms:CreateGrant kms:Decrypt kms:DescribeKey kms:GenerateDataKey	arn:aws:sagemaker:*region*:*account-id*:inference-recommendations-job/*inferenceRecommendationsJobName*
CreateHyperParameterTuningJob	sagemaker:CreateHyperParameterTuningJob iam:PassRole 仅当任何关联的ResourceConfig有指定VolumeKmsKeyId 而且关联的角色没有允许此操作的策略： kms:CreateGrant	arn:aws:sagemaker:*region*:*account-id*:hyper-parameter-tuning-job/*hyperParameterTuningJobName*
CreateImage	sagemaker:CreateImage iam:PassRole	arn:aws:sagemaker:*region*:*account-id*:image/*
CreateImageVersion	sagemaker:CreateImageVersion	arn:aws:sagemaker:*region*:*account-id*:image-version/*imageName*/*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
CreateLabelingJob	sagemaker:CreateLabelingJob iam:PassRole	arn:aws:sagemaker:*region*:*account-id*:labeling-job/*labelingJobName*
CreateModel	sagemaker:CreateModel iam:PassRole	arn:aws:sagemaker:*region*:*account-id*:model/*modelName*
CreateModelPackage	sagemaker:CreateModelPackage	arn:aws:sagemaker:*region*:*account-id*:model-package/*modelPackageName*
CreateModelPackageGroup	sagemaker:CreateModelPackageGroup	arn:aws:sagemaker:*region*:*account-id*:model-package-group/*modelPackageGroupName*
CreateNotebookInstance	sagemaker:CreateNotebookInstance iam:PassRole 仅当您为您的笔记本实例指定 VPC 时，才需要以下权限： ec2:CreateNetworkInterface ec2:DescribeSecurityGroups ec2:DescribeSubnets ec2:DescribeVpcs 仅当您为您的笔记本实例指定 VPC 和弹性推理加速器时，才需要以下权限： ec2:DescribeVpcEndpoints 仅当您指定加密密钥时，才需要以下权限： kms:DescribeKey kms:CreateGrant 仅当您指定 Amazon Secrets Manager 密钥以访问私有 Git 存储库时，才需要以下权限： secretsmanager:GetSecretValue	arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*
CreatePipeline	sagemaker:CreatePipeline iam:PassRole	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name* arn:*aws-partition*:iam::*account-id*:role/*role-name*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
CreatePresignedDomainUrl	sagemaker:CreatePresignedDomainUrl	arn:aws:sagemaker:*region*:*account-id*:app/domain-id/*userProfileName*/*
CreatePresignedNotebookInstanceUrl	sagemaker:CreatePresignedNotebookInstanceUrl	arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*
CreateProcessingJob	sagemaker:CreateProcessingJob iam:PassRole kms:CreateGrant（仅当关联的 ProcessingResources 具有一个指定的 VolumeKmsKeyId 且关联角色没有允许此操作的策略时才必需）	arn:aws:sagemaker:*region*:*account-id*:processing-job/*processingJobName*
CreateStudioLifecycleConfig	sagemaker:CreateStudioLifecycleConfig	arn:aws:sagemaker:*region*:*account-id*:studio-lifecycle-config/.*
CreateTrainingJob	sagemaker:CreateTrainingJob iam:PassRole kms:CreateGrant（仅当关联的 ResourceConfig 具有一个指定的 VolumeKmsKeyId 且关联角色没有允许此操作的策略时才必需）	arn:aws:sagemaker:*region*:*account-id*:training-job/*trainingJobName*
CreateTransformJob	sagemaker:CreateTransformJob kms:CreateGrant（仅当关联的 TransformResources 具有一个指定的 VolumeKmsKeyId 且关联角色没有允许此操作的策略时才必需）	arn:aws:sagemaker:*region*:*account-id*:transform-job/*transformJobName*
CreateUserProfile	sagemaker:CreateUserProfile iam:PassRole	arn:aws:sagemaker:*region*:*account-id*:user-profile/domain-id/*userProfileName*
CreateWorkforce	sagemaker:CreateWorkforce cognito-idp:DescribeUserPoolClient cognito-idp:UpdateUserPool cognito-idp:DescribeUserPool cognito-idp:UpdateUserPoolClient	arn:aws:sagemaker:*region*:*account-id*:workforce/*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
CreateWorkteam	sagemaker:CreateWorkteam cognito-idp:DescribeUserPoolClient cognito-idp:UpdateUserPool cognito-idp:DescribeUserPool cognito-idp:UpdateUserPoolClient	arn:aws:sagemaker:*region*:*account-id*:workteam/private-crowd/*work team name*
DeleteApp	sagemaker:DeleteApp	arn:aws:sagemaker:*region*:*account-id*:app/*domain-id*/*user-profile-name*/*app-type*/*appName*
DeleteAppImageConfig	sagemaker:DeleteAppImageConfig	arn:aws:sagemaker:*region*:*account-id*:app-image-config/*appImageConfigName*
DeleteDomain	sagemaker:DeleteDomain	arn:aws:sagemaker:*region*:*account-id*:domain/*domainId*
DeleteEndpoint	sagemaker:DeleteEndpoint	arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName*
DeleteEndpointConfig	sagemaker:DeleteEndpointConfig	arn:aws:sagemaker:*region*:*account-id*:endpoint-config/*endpointConfigName*
DeleteFlowDefinition	sagemaker:DeleteFlowDefinition	arn:aws:sagemaker:*region*:*account-id*:flow-definition/*flowDefinitionName*
DeleteHumanLoop	sagemaker:DeleteHumanLoop	arn:aws:sagemaker:*region*:*account-id*:human-loop/*humanLoopName*
DeleteImage	sagemaker:DeleteImage	arn:aws:sagemaker:*region*:*account-id*:image/*imageName*
DeleteImageVersion	sagemaker:DeleteImageVersion	arn:aws:sagemaker:*region*:*account-id*:image-version/*imageName*/*versionNumber*
DeleteModel	sagemaker:DeleteModel	arn:aws:sagemaker:*region*:*account-id*:model/*modelName*
DeleteModelPackage	sagemaker:DeleteModelPackage	arn:aws:sagemaker:*region*:*account-id*:model-package/*modelPackageName*
DeleteModelPackageGroup	sagemaker:DeleteModelPackageGroup	arn:aws:sagemaker:*region*:*account-id*:model-package-group/*modelPackageGroupName*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
DeleteModelPackageGroup	sagemaker:DeleteModelPackageGroup sagemaker:DeletePolicy	arn:aws:sagemaker:*region*:*account-id*:model-package-group/*modelPackageGroupName*
DeleteNotebookInstance	sagemaker:DeleteNotebookInstance 仅当您为您的笔记本实例指定 VPC 时，才需要以下权限： ec2:DeleteNetworkInterface 仅当您在创建笔记本实例时指定了加密密钥的情况下，才需要以下权限： kms:DescribeKey	arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*
DeletePipeline	sagemaker:DeletePipeline	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*
DeleteTags	sagemaker:DeleteTags	arn:aws:sagemaker:*region*:*account-id*:*
DeleteUserProfile	sagemaker:DeleteUserProfile	arn:aws:sagemaker:*region*:*account-id*:user-profile/domain-id/*userProfileName*
DeleteWorkforce	sagemaker:DeleteWorkforce	arn:aws:sagemaker:*region*:*account-id*:workforce/*
DeleteWorkteam	sagemaker:DeleteWorkteam	arn:aws:sagemaker:*region*:*account-id*:workteam/private-crowd/*
DescribeApp	sagemaker:DescribeApp	arn:aws:sagemaker:*region*:*account-id*:app/*domain-id*/*user-profile-name*/*app-type*/*appName*
DescribeAppImageConfig	sagemaker:DescribeAppImageConfig	arn:aws:sagemaker:*region*:*account-id*:app-image-config/*appImageConfigName*
DescribeDomain	sagemaker:DescribeDomain	arn:aws:sagemaker:*region*:*account-id*:domain/*domainId*
DescribeEndpoint	sagemaker:DescribeEndpoint	arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName*
DescribeEndpointConfig	sagemaker:DescribeEndpointConfig	arn:aws:sagemaker:*region*:*account-id*:endpoint-config/*endpointConfigName*
DescribeFlowDefinition	sagemaker:DescribeFlowDefinition	arn:aws:sagemaker:*region*:*account-id*:flow-definition/*flowDefinitionName*
DescribeHumanLoop	sagemaker:DescribeHumanLoop	arn:aws:sagemaker:*region*:*account-id*:human-loop/*humanLoopName*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
DescribeHumanTaskUi	sagemaker:DescribeHumanTaskUi	arn:aws:sagemaker:*region*:*account-id*:human-task-ui/*humanTaskUiName*
DescribeHyperParameterTuningJob	sagemaker:DescribeHyperParameterTuningJob	arn:aws:sagemaker:*region*:*account-id*:hyper-parameter-tuning-job/*hyperParameterTuningJob*
DescribeImage	sagemaker:DescribeImage	arn:aws:sagemaker:*region*:*account-id*:image/*imageName*
DescribeImageVersion	sagemaker:DescribeImageVersion	arn:aws:sagemaker:*region*:*account-id*:image-version/*imageName*/*versionNumber*
DescribeLabelingJob	sagemaker:DescribeLabelingJob	arn:aws:sagemaker:*region*:*account-id*:labeling-job/*labelingJobName*
DescribeModel	sagemaker:DescribeModel	arn:aws:sagemaker:*region*:*account-id*:model/*modelName*
DescribeModelPackage	sagemaker:DescribeModelPackage	arn:aws:sagemaker:*region*:*account-id*:model-package/*modelPackageName*
DescribeModelPackageGroup	sagemaker:DescribeModelPackageGroup	arn:aws:sagemaker:*region*:*account-id*:model-package-group/*modelPackageGroupName*
DescribeNotebookInstance	sagemaker:DescribeNotebookInstance	arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*
DescribePipeline	sagemaker:DescribePipeline	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*
DescribePipelineDefinitionForExecution	sagemaker:DescribePipelineDefinitionForExecution	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*/execution/*execution-id*
DescribePipelineExecution	sagemaker:DescribePipelineExecution	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*/execution/*execution-id*
DescribeProcessingJob	sagemaker:DescribeProcessingJob	arn:aws:sagemaker:*region*:*account-id*:processing-job/*processingjobname*
DescribeSubscribedWorkteam	sagemaker:DescribeSubscribedWorkteam aws-marketplace:ViewSubscriptions	arn:aws:sagemaker:*region*:*account-id*:workteam/vendor-crowd/*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
DescribeTrainingJob	sagemaker:DescribeTrainingJob	arn:aws:sagemaker:*region*:*account-id*:training-job/*trainingjobname*
DescribeTransformJob	sagemaker:DescribeTransformJob	arn:aws:sagemaker:*region*:*account-id*:transform-job/*transformjobname*
DescribeUserProfile	sagemaker:DescribeUserProfile	arn:aws:sagemaker:*region*:*account-id*:user-profile/domain-id/*userProfileName*
DescribeWorkforce	sagemaker:DescribeWorkforce	arn:aws:sagemaker:*region*:*account-id*:workforce/*
DescribeWorkteam	sagemaker:DescribeWorkteam	arn:aws:sagemaker:*region*:*account-id*:workteam/private-crowd/*
GetModelPackageGroupPolicy	sagemaker:GetModelPackageGroupPolicy	arn:aws:sagemaker:*region*:*account-id*:model-package-group/*modelPackageGroupName*
InvokeEndpoint	sagemaker:InvokeEndpoint	arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName*
ListAppImageConfigs	sagemaker:ListAppImageConfigs	arn:aws:sagemaker:*region*:*account-id*:app-image-config/*
ListApps	sagemaker:ListApps	arn:aws:sagemaker:*region*:*account-id*:app/*domain-id*/*user-profile-name*/*
ListDomains	sagemaker:ListDomains	arn:aws:sagemaker:*region*:*account-id*:domain/*
ListEndpointConfigs	sagemaker:ListEndpointConfigs	*
ListEndpoints	sagemaker:ListEndpoints	*
ListFlowDefinitions	sagemaker:ListFlowDefinitions	*
ListHumanLoops	sagemaker:ListHumanLoops	*
ListHumanTaskUis	sagemaker:ListHumanTaskUis	*
ListHyperParameterTuningJobs	sagemaker:ListHyperParameterTuningJobs	arn:aws:sagemaker:*region*:*account-id*:hyper-parameter-tuning-job/*hyperParameterTuningJob*
ListImages	sagemaker:ListImages	*
ListImageVersions	sagemaker:ListImageVersions	arn:aws:sagemaker:*region*:*account-id*:image/*
ListLabelingJobs	sagemaker:ListLabelingJobs	*
ListLabelingJobsForWorkteam	sagemaker:ListLabelingJobForWorkteam	*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
ListModelPackageGroups	sagemaker:ListModelPackageGroups	arn:aws:sagemaker:*region*:*account-id* :model-package-group/*ModelPackageGroupName*
ListModelPackages	sagemaker:ListModelPackages	arn:aws:sagemaker:*region*:*account-id* :model-package/*ModelPackageName*
ListModels	sagemaker:ListModels	*
ListNotebookInstances	sagemaker:ListNotebookInstances	*
ListPipelineExecutions	sagemaker:ListPipelineExecutions	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*
ListPipelineExecutionSteps	sagemaker:ListPipelineExecutionSteps	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*/execution/*execution-id*
ListPipelineParametersForExecution	sagemaker:ListPipelineParametersForExecution	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*/execution/*execution-id*
ListPipelines	sagemaker:ListPipelines	*
ListProcessingJobs	sagemaker:ListProcessingJobs	*
ListSubscribedWorkteams	sagemaker:ListSubscribedWorkteams aws-marketplace:ViewSubscriptions	*
ListTags	sagemaker:ListTags	arn:aws:sagemaker:*region*:*account-id*:*
ListTrainingJobs	sagemaker:ListTrainingJobs	*
ListTrainingJobsForHyperParameterTuningJob	sagemaker:ListTrainingJobsForHyperParameterTuningJob	arn:aws:sagemaker:*region*:*account-id*:hyper-parameter-tuning-job/*hyperParameterTuningJob*
ListTransformJobs	sagemaker:ListTransformJobs	*
ListUserProfiles	sagemaker:ListUserProfiles	arn:aws:sagemaker:*region*:*account-id*:user-profile/domain-id/*
ListWorkforces	sagemaker:ListWorkforces	*
ListWorkteams	sagemaker:ListWorkteams	*
PutModelPackageGroupPolicy	sagemaker:PutModelPackageGroupPolicy	arn:aws:sagemaker:*region*:*account-id*:model-package-group/*modelPackageGroupName*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
RetryPipelineExecution	sagemaker:RetryPipelineExecution	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*/execution/*execution-id*
SendPipelineExecutionStepFailure	sagemaker:SendPipelineExecutionStepFailure	*
SendPipelineExecutionStepSuccess	sagemaker:SendPipelineExecutionStepSuccess	*
StartHumanLoop	sagemaker:StartHumanLoop	arn:aws:sagemaker:*region*:*account-id*:human-loop/*humanLoopName*
StartNotebookInstance	sagemaker:StartNotebookInstance iam:PassRole 仅当您在创建笔记本实例时指定了 VPC 的情况下，才需要以下权限： ec2:CreateNetworkInterface ec2:DescribeNetworkInterfaces ec2:DescribeSecurityGroups ec2:DescribeSubnets ec2:DescribeVpcs 仅当您为您的笔记本实例指定 VPC 和弹性推理加速器时，才需要以下权限： ec2:DescribeVpcEndpoints 仅当您在创建笔记本实例时指定了加密密钥的情况下，才需要以下权限： kms:DescribeKey kms:CreateGrant 仅当您在创建笔记本实例时指定了 Amazon Secrets Manager 密钥以访问私有 Git 存储库时，才需要以下权限： secretsmanager:GetSecretValue	arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*
StartPipelineExecution	sagemaker:StartPipelineExecution	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*
StopHumanLoop	sagemaker:StopHumanLoop	arn:aws:sagemaker:*region*:*account-id*:human-loop/*humanLoopName*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
StopHyperParameterTuningJob	sagemaker:StopHyperParameterTuningJob	arn:aws:sagemaker:*region*:*account-id*:hyper-parameter-tuning-job/*hyperParameterTuningJob*
StopLabelingJob	sagemaker:StopLabelingJob	arn:aws:sagemaker:*region*:*account-id*:labeling-job/*labelingJobName*
StopNotebookInstance	sagemaker:StopNotebookInstance	arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*
StopPipelineExecution	sagemaker:StopPipelineExecution	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*/execution/*execution-id*
StopProcessingJob	sagemaker:StopProcessingJob	arn:aws:sagemaker:*region*:*account-id*:processing-job/*processingJobName*
StopTrainingJob	sagemaker:StopTrainingJob	arn:aws:sagemaker:*region*:*account-id*:training-job/*trainingJobName*
StopTransformJob	sagemaker:StopTransformJob	arn:aws:sagemaker:*region*:*account-id*:transform-job/*transformJobName*
UpdateAppImageConfig	sagemaker:UpdateAppImageConfig	arn:aws:sagemaker:*region*:*account-id*:app-image-config/*appImageConfigName*
UpdateDomain	sagemaker:UpdateDomain	arn:aws:sagemaker:*region*:*account-id*:domain/*domainId*
UpdateEndpoint	sagemaker:UpdateEndpoint	arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName*
UpdateEndpointWeightsAndCapacities	sagemaker:UpdateEndpointWeightsAndCapacities	arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName*
UpdateImage	sagemaker:UpdateImage iam:PassRole	arn:aws:sagemaker:*region*:*account-id*:image/*imageName*
UpdateModelPackage	sagemaker:UpdateModelPackage	arn:aws:sagemaker:*region*:*account-id*:model-package/*modelPackageName*
UpdateNotebookInstance	sagemaker:UpdateNotebookInstance iam:PassRole	arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*

亚马逊 SageMaker API 操作	所需权限（API 操作）	资源
UpdatePipeline	sagemaker:UpdatePipeline iam:PassRole	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name* arn:*aws-partition*:iam::*account-id*:role/*role-name*
UpdatePipelineExecution	sagemaker:UpdatePipelineExecution	arn:*aws-partition*:sagemaker:*region*:*account-id*:pipeline/*pipeline-name*/execution/*execution-id*
UpdateUserProfile	sagemaker:UpdateUserProfile	arn:aws:sagemaker:*region*:*account-id*:user-profile/domain-id/*userProfileName*
UpdateWorkforce	sagemaker:UpdateWorkforce	arn:aws:sagemaker:*region*:*account-id*:workforce/*
UpdateWorkteam	sagemaker:UpdateWorkteam	arn:aws:sagemaker:*region*:*account-id*:workteam/private-crowd/*

亚马逊 SageMaker API 和必需的操作权限

API 操作： AddTags

所需权限（API 操作）：sagemaker:AddTags

资源: *

API 操作： CreateEndpoint

所需权限（API 操作）：sagemaker:CreateEndpoint

资源: arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName*

API 操作： CreateEndpointConfig

所需权限（API 操作）：sagemaker:CreateEndpointConfig

资源: arn:aws:sagemaker:*region*:*account-id*:endpoint-config/*endpointConfigName*

API 操作： CreateModel

所需权限（API 操作）：sagemaker:CreateModel, iam:PassRole

资源: arn:aws:sagemaker:*region*:*account-id*:model/*modelName*

API 操作： CreateLabelingJob

所需权限（API 操作）：sagemaker:CreateLabelingJob, iam:PassRole

资源: arn:aws:sagemaker:*region*:*account-id*:labeling-job/*labelingJobName*

API 操作： CreateNotebookInstance

所需权限（API 操作）：sagemaker:CreateNotebookInstance, iam:PassRole, ec2:CreateNetworkInterface, ec2:AttachNetworkInterface, ec2:ModifyNetworkInterfaceAttribute, ec2:DescribeAvailabilityZones,

ec2:DescribeInternetGateways, ec2:DescribeSecurityGroups, ec2:DescribeSubnets, ec2:DescribeVpcs, kms:CreateGrant

资源: arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*

API 操作： CreateTrainingJob

所需权限（API 操作）：sagemaker:CreateTrainingJob, iam:PassRole

资源: arn:aws:sagemaker:*region*:*account-id*:training-job/*trainingJobName*

API 操作： CreateWorkforce

所需权限（API 操作）：sagemaker:CreateWorkforce、cognito-idp:DescribeUserPoolClient、cognito-idp:UpdateUserPool、cognito-idp:DescribeUserPool、cognito-idp:UpdateUserPoolClient

资源: arn:aws:sagemaker:*region*:*account-id*:workforce/*

API 操作： CreateWorkteam

所需权限（API 操作）：sagemaker:CreateWorkteam、cognito-idp:DescribeUserPoolClient、cognito-idp:UpdateUserPool、cognito-idp:DescribeUserPool、cognito-idp:UpdateUserPoolClient

资源:arn:aws:sagemaker:*region*:*account-id*:workteam/private-crowd/*work team name*

API 操作： DeleteEndpoint

所需权限（API 操作）：sagemaker:DeleteEndpoint

资源: arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName*

API 操作： DeleteEndpointConfig

所需权限（API 操作）：sagemaker:DeleteEndpointConfig

资源: arn:aws:sagemaker:*region*:*account-id*:endpoint-config/*endpointConfigName*

API 操作： DeleteModel

所需权限（API 操作）：sagemaker:DeleteModel

资源: arn:aws:sagemaker:*region*:*account-id*:model/*modelName*

API 操作： DeleteNotebookInstance

所需权限（API 操作）：sagemaker:DeleteNotebookInstance, ec2:DeleteNetworkInterface, ec2:DetachNetworkInterface, ec2:DescribeAvailabilityZones, ec2:DescribeInternetGateways, ec2:DescribeSecurityGroups, ec2:DescribeSubnets, ec2:DescribeVpcs

资源: arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*

API 操作： DeleteTags

所需权限（API 操作）：sagemaker:DeleteTags

资源: *

API 操作： DeleteWorkteam

所需权限（API 操作）：sagemaker:DeleteWorkforce

资源: arn:aws:sagemaker:*region*:*account-id*:workforce/private-crowd/*

API 操作： DeleteWorkteam

所需权限（API 操作）：sagemaker:DeleteWorkteam

资源: arn:aws:sagemaker:*region*:*account-id*:workteam/private-crowd/*

API 操作： DescribeEndpoint

所需权限（API 操作）：sagemaker:DescribeEndpoint

资源: arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName*

API 操作： DescribeEndpointConfig

所需权限（API 操作）：sagemaker:DescribeEndpointConfig

资源: arn:aws:sagemaker:*region*:*account-id*:endpoint-config/*endpointConfigName*

API 操作： DescribeLabelingJob

所需权限（API 操作）：sagemaker:DescribeLabelingJob

资源: arn:aws:sagemaker:*region*:*account-id*:labeling-job/*labelingJobName*

API 操作： DescribeModel

所需权限（API 操作）：sagemaker:DescribeModel

资源: arn:aws:sagemaker:*region*:*account-id*:model/*modelName*

API 操作： DescribeNotebookInstance

所需权限（API 操作）：sagemaker:DescribeNotebookInstance

资源: arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*

API 操作： DescribeSubscribedWorkforce

所需权限 (API 操作)：sagemaker:DescribeSubscribedWorkforce、aws-marketplace:ViewSubscriptions

资源: arn:aws:sagemaker:*region*:*account-id*:workforce/*

API 操作： DescribeSubscribedWorkteam

所需权限 (API 操作)：sagemaker:DescribeSubscribedWorkteam、aws-marketplace:ViewSubscriptions

资源: arn:aws:sagemaker:*region*:*account-id*:workteam/vendor-crowd/*

API 操作： DescribeTrainingJob

所需权限（API 操作）：sagemaker:DescribeTrainingJob

资源: arn:aws:sagemaker:*region*:*account-id*:training-job/*trainingJobName*

API 操作： DescribeWorkteam

所需权限（API 操作）：sagemaker:DescribeWorkteam

资源: arn:aws:sagemaker:*region*:*account-id*:workteam/private-crowd/*

API 操作： CreatePresignedNotebookInstanceUrl

所需权限（API 操作）：sagemaker:CreatePresignedNotebookInstanceUrl

资源: arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*

API 操作： runtime_InvokeEndpoint

所需权限（API 操作）：sagemaker:InvokeEndpoint

资源: arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName*

API 操作： ListEndpointConfigs

所需权限（API 操作）：sagemaker:ListEndpointConfigs

资源: *

API 操作： ListEndpoints

所需权限（API 操作）：sagemaker:ListEndpoints

资源: *

API 操作： ListLabelingJobs

所需权限（API 操作）：sagemaker:ListLabelingJobs

资源: *

API 操作： ListLabelingJobsForWorkteam

所需权限（API 操作）：sagemaker:ListLabelingJobsForWorkteam

资源: *

API 操作： ListModels

所需权限（API 操作）：sagemaker:ListModels

资源: *

API 操作： ListNotebookInstances

所需权限（API 操作）：sagemaker:ListNotebookInstances

资源: *

API 操作： ListSubscribedWorkteams

所需权限 (API 操作)：sagemaker:ListSubscribedWorkteam、aws-marketplace:ViewSubscriptions

资源: *

API 操作： ListTags

所需权限（API 操作）：sagemaker:ListTags

资源: *

API 操作： ListTrainingJobs

所需权限（API 操作）：sagemaker:ListTrainingJobs

资源: *

API 操作： ListWorkteams

所需权限（API 操作）：sagemaker:ListWorkforces

资源: *

API 操作： ListWorkteams

所需权限（API 操作）：sagemaker:ListWorkteams

资源: *

API 操作： StartNotebookInstance

所需权限（API 操作）：sagemaker:StartNotebookInstance, iam:PassRole, ec2:CreateNetworkInterface, ec2:AttachNetworkInterface, ec2:ModifyNetworkInterfaceAttribute, ec2:DescribeAvailabilityZones, ec2:DescribeInternetGateways, ec2:DescribeSecurityGroups, ec2:DescribeSubnets, ec2:DescribeVpcs, kms:CreateGrant

资源: arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*

API 操作： StopLabelingJob

所需权限（API 操作）：sagemaker:StopLabelingJob

资源: arn:aws:sagemaker:*region*:*account-id*:labeling-job/*labelingJobName*

API 操作： StopNotebookInstance

所需权限（API 操作）：sagemaker:StopNotebookInstance

资源: arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*

API 操作： StopTrainingJob

所需权限（API 操作）：sagemaker:StopTrainingJob

资源: arn:aws:sagemaker:*region*:*account-id*:training-job/*trainingJobName*

API 操作： UpdateEndpoint

所需权限（API 操作）：sagemaker:UpdateEndpoints

资源: arn:aws:sagemaker:*region*:*account-id*:endpoint/*endpointName*

API 操作： UpdateNotebookInstance

所需权限（API 操作）：sagemaker:UpdateNotebookInstance, iam:PassRole

资源: arn:aws:sagemaker:*region*:*account-id*:notebook-instance/*notebookInstanceName*

API 操作： UpdateWorkteam

所需权限（API 操作）：sagemaker:UpdateWorkteam

资源: arn:aws:sagemaker:*region*:*account-id*:workteam/private-crowd/*

AmazonAmazon SageMaker 的托管式策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 Amazon 托管策略更简单。创建仅为团队提供所需权限的 IAM 客户托管策略需要时间和专业知识。要快速入门，您可以使用我们的 Amazon 托管策略。这些策略涵盖常见使用案例，可在您的 Amazon 账户中使用。有关 Amazon 托管策略的更多信息，请参阅《IAM 用户指南》中的 Amazon 托管策略。

Amazon 服务负责维护和更新 Amazon 托管策略。您无法更改 Amazon 托管策略中的权限。服务偶尔会向 Amazon 托管式策略添加额外权限以支持新功能。此类更新会影响附加到策略的所有身份（用户、组和角色）。当启动新功能或新操作可用时，服务最有可能会更新 Amazon 托管式策略。服务不会从 Amazon 托管式策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，Amazon 还支持跨多种服务的工作职能的托管式策略。例如，`ReadOnlyAccess` Amazon托管策略提供对所有的只读访问Amazon服务和资源。当服务启动新功能时，Amazon 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅 IAM 用户指南中的适用于工作职能的 Amazon 托管策略。

> **Important**
>
> 我们建议您使用最受限制的策略，以便您执行使用案例。

以下Amazon您可以将托管策略附加到您账户中的用户，这些策略特定于 Amazon SageMaker：

- **`AmazonSageMakerFullAccess`**— 授予对 Amazon 的完全访问权限 SageMaker 资源以及受支持的操作。这不提供无限制的 Amazon S3 访问，但支持具有特定特定的存储桶和对象。`sagemaker`标签。此策略允许将所有 IAM 角色传递给 Amazon SageMaker，但只允许将其中包含 “AmazonSageMaker” 的 IAM 角色传递给Amazon Glue、Amazon Step Functions, 和Amazon RoboMaker 服务。
- **`AmazonSageMakerReadOnly`**— 授予对 Amazon 的只读访问权限 SageMaker 资源的费用。

以下Amazon您可以将托管策略附加到您账户中的用户，但不建议使用：

- `AdministratorAccess`— 授予所有操作Amazon服务和账户中的所有资源。
- `DataScientist`— 授予广泛的权限以涵盖数据科学家所遇到的大多数使用案例（主要用于分析和商业智能）。

您可以通过登录到 IAM 控制台并搜索这些权限策略来查看它们。

您还可以创建自己的自定义 IAM 策略，以授予 Amazon SageMaker 操作和资源的相关权限（根据需要它们）。您可以将这些自定义策略附加到需要它们的 IAM 用户或组。

主题

- Amazon托管策略：AmazonSageMakerFullAccess (p. 2755)
- Amazon托管策略：AmazonSageMakerReadOnly (p. 2764)
- Amazon适用于 Amazon 的托管策略 SageMaker Ground Truth (p. 2765)
- Amazon适用于 的 托管策略 SageMaker 管道 (p. 2768)
- Amazon适用于 的 托管策略 SageMaker 项目和 JumpStart (p. 2769)
- SageMaker 更新Amazon管理的策略 (p. 2777)

Amazon托管策略：AmazonSageMakerFullAccess

此策略授予管理权限，允许委托人完全访问所有 Amazon。 SageMaker 资源和操作。该策略还提供了对相关服务的选择访问权限。此策略允许将所有 IAM 角色传递给 Amazon SageMaker，但只允许将其中包含 “AmazonSageMaker” 的 IAM 角色传递给Amazon Glue、Amazon Step Functions, 和Amazon RoboMaker 服务。此策略不包括创建亚马逊的权限 SageMaker 域。有关创建域所需的策略的信息，请参阅创建 IAM 管理员用户和组 (p. 31).

权限详细信息

此策略包含以下权限。

- `application-autoscaling`— 允许委托人自动扩展 SageMaker 实时推理终端节点。
- `athena`— 允许承担者查询来自的数据目录、数据库和表元数据的列表Amazon Athena.
- `aws-marketplace`— 允许委托人查看Amazon订阅 AI Marketplace。如果你想访问订阅的 SageMaker 软件，你需要这个Amazon Web Services Marketplace.
- `cloudformation`— 允许校长获得Amazon CloudFormation使用的模板 SageMaker JumpStart 解决方案和管道。 SageMaker JumpStart 创建运行所需的资源 end-to-end 紧密的机器学习解决方案 SageMaker 到其他Amazon服务。 SageMaker Pipeline 创建了由以下支持的新项目Amazon Service Catalog.

- `cloudwatch`— 允许校长发布 CloudWatch 指标、与警报交互以及将日志上传到 CloudWatch 登录你的账户。
- `codebuild`— 允许委托人存储Amazon CodeBuild的构件 SageMaker 管道和项目。
- `codecommit`— 必需Amazon CodeCommit与 SageMaker 笔记本实例相集成。
- `cognito-idp`— 亚马逊需要 SageMaker 定义私有人力和工作团队的 Ground Truth。
- `ec2`— 必需 SageMaker 在为您指定 Amazon VPC 时管理 Amazon EC2 资源和网络接口 SageMaker 作业、模型、终端节点和笔记本实例。
- `ecr`— 需要为亚马逊提取和存储 Docker 文物 SageMaker Studio（自定义图像）、训练、处理、批量推理和推理端点。在 SageMaker 中使用您自己的容器也需要此项目。其他权限 SageMaker 需要 JumpStart 解决方案来代表用户创建和删除自定义映像。
- `elastic-inference`— 允许委托人连接到 Amazon Elastic Inference 以便使用 SageMaker 笔记本实例和终端节点。
- `elasticfilesystem`— 允许委托人访问 Amazon Elastic File System。需要使用此信息 SageMaker 使用 Amazon Elastic File System 中的数据源来训练机器学习模型。
- `fsx`— 允许委托人访问亚马逊 FSx。需要使用此信息 SageMaker 使用 Amazon FSx 中的数据源来训练机器学习模型。
- `glue`— 从内部进行推理管道预处理所需 SageMaker 笔记本实例。
- `groundtruthlabeling`— 为 Ground Truth 贴标工作所需。这些区域有：`groundtruthlabeling`终端节点由 Ground Truth 控制台访问。
- `iam`— 需要给 SageMaker 对可用 IAM 角色的控制台访问权限并创建服务相关角色。
- `kms`— 需要给 SageMaker 对可用的控制台访问Amazon KMS密钥并检索任何指定的Amazon KMS作业和终端中的别名。
- `lambda`— 允许委托人调用并获取列表Amazon Lambda函数。
- `logs`— 需要允许 SageMaker 作业和终端节点来发布日志流。
- `redshift`— 允许委托人访问 Amazon Redshift 群集凭证。
- `redshift-data`— 允许委托人使用来自 Amazon Redshift 的数据来运行、描述和取消语句；获取语句结果；以及列出架构和表格。
- `robomaker`— 允许委托人拥有创建、获取描述和删除的完全访问权限Amazon RoboMaker 模拟应用程序和作业。在笔记本实例上运行强化学习示例也需要这样做。
- `s3`— 允许委托人完全访问与 SageMaker 相关的 Amazon S3 资源，但不是所有 Amazon S3。
- `sagemaker`— 允许委托人在亚马逊上发布标签 SageMaker 用户配置文件。
- `secretsmanager`— 允许委托人拥有完全访问权限Amazon Secrets Manager. 委托人可以安全地加密、存储和检索数据库和其他服务的凭证。这也是必需的 SageMaker 带有笔记本实例 SageMaker 使用 GitHub 的代码存储库。
- `servicecatalog`— 允许委托人使用Amazon Service Catalog. 委托人可以创建、获取列表、更新或终止使用部署的服务器、数据库、网站或应用程序等预配置产品Amazon资源的费用。需要使用此信息 SageMaker JumpStart 和 Projects 用于查找和阅读服务目录产品并启动Amazon用户帐户中的资源。
- `sns`— 允许委托人获取 Amazon SNS 主题列表。启用了异步推理的端点需要这样做，才能通知用户他们的推理已完成。
- `states`— 必需 SageMaker JumpStart 和 Pipeline 使用服务目录创建步骤函数资源。
- `tag`— 必需 SageMaker 要在 Studio 中渲染的管道。Studio 需要特别标记的资源`sagemaker:project-id`标签键。这需要`tag:GetResources`权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:*"
            ],
```

```
            "NotResource": [
                "arn:aws:sagemaker:*:*:domain/*",
                "arn:aws:sagemaker:*:*:user-profile/*",
                "arn:aws:sagemaker:*:*:app/*",
                "arn:aws:sagemaker:*:*:flow-definition/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedDomainUrl",
                "sagemaker:DescribeDomain",
                "sagemaker:ListDomains",
                "sagemaker:DescribeUserProfile",
                "sagemaker:ListUserProfiles",
                "sagemaker:*App",
                "sagemaker:ListApps"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "sagemaker:*",
            "Resource": [
                "arn:aws:sagemaker:*:*:flow-definition/*"
            ],
            "Condition": {
                "StringEqualsIfExists": {
                    "sagemaker:WorkteamType": [
                        "private-crowd",
                        "vendor-crowd"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DeleteScheduledAction",
                "application-autoscaling:DeregisterScalableTarget",
                "application-autoscaling:DescribeScalableTargets",
                "application-autoscaling:DescribeScalingActivities",
                "application-autoscaling:DescribeScalingPolicies",
                "application-autoscaling:DescribeScheduledActions",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:PutScheduledAction",
                "application-autoscaling:RegisterScalableTarget",
                "aws-marketplace:ViewSubscriptions",
                "cloudformation:GetTemplateSummary",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:PutMetricData",
                "codecommit:BatchGetRepositories",
                "codecommit:CreateRepository",
                "codecommit:GetRepository",
                "codecommit:List*",
                "cognito-idp:AdminAddUserToGroup",
                "cognito-idp:AdminCreateUser",
                "cognito-idp:AdminDeleteUser",
                "cognito-idp:AdminDisableUser",
                "cognito-idp:AdminEnableUser",
```

```
                "cognito-idp:AdminRemoveUserFromGroup",
                "cognito-idp:CreateGroup",
                "cognito-idp:CreateUserPool",
                "cognito-idp:CreateUserPoolClient",
                "cognito-idp:CreateUserPoolDomain",
                "cognito-idp:DescribeUserPool",
                "cognito-idp:DescribeUserPoolClient",
                "cognito-idp:List*",
                "cognito-idp:UpdateUserPool",
                "cognito-idp:UpdateUserPoolClient",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:CreateVpcEndpoint",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:CreateRepository",
                "ecr:Describe*",
                "ecr:GetAuthorizationToken",
                "ecr:GetDownloadUrlForLayer",
                "ecr:StartImageScan",
                "elastic-inference:Connect",
                "elasticfilesystem:DescribeFileSystems",
                "elasticfilesystem:DescribeMountTargets",
                "fsx:DescribeFileSystems",
                "glue:CreateJob",
                "glue:DeleteJob",
                "glue:GetJob*",
                "glue:GetTable*",
                "glue:GetWorkflowRun",
                "glue:ResetJobBookmark",
                "glue:StartJobRun",
                "glue:StartWorkflowRun",
                "glue:UpdateJob",
                "groundtruthlabeling:*",
                "iam:ListRoles",
                "kms:DescribeKey",
                "kms:ListAliases",
                "lambda:ListFunctions",
                "logs:CreateLogDelivery",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DeleteLogDelivery",
                "logs:Describe*",
                "logs:GetLogDelivery",
                "logs:GetLogEvents",
                "logs:ListLogDeliveries",
                "logs:PutLogEvents",
                "logs:PutResourcePolicy",
                "logs:UpdateLogDelivery",
                "robomaker:CreateSimulationApplication",
                "robomaker:DescribeSimulationApplication",
                "robomaker:DeleteSimulationApplication",
                "robomaker:CreateSimulationJob",
                "robomaker:DescribeSimulationJob",
                "robomaker:CancelSimulationJob",
                "secretsmanager:ListSecrets",
                "servicecatalog:Describe*",
```

```
            "servicecatalog:List*",
            "servicecatalog:ScanProvisionedProducts",
            "servicecatalog:SearchProducts",
            "servicecatalog:SearchProvisionedProducts",
            "sns:ListTopics",
            "tag:GetResources"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "ecr:SetRepositoryPolicy",
            "ecr:CompleteLayerUpload",
            "ecr:BatchDeleteImage",
            "ecr:UploadLayerPart",
            "ecr:DeleteRepositoryPolicy",
            "ecr:InitiateLayerUpload",
            "ecr:DeleteRepository",
            "ecr:PutImage"
        ],
        "Resource": [
            "arn:aws:ecr:*:*:repository/*sagemaker*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "codecommit:GitPull",
            "codecommit:GitPush"
        ],
        "Resource": [
            "arn:aws:codecommit:*:*:*sagemaker*",
            "arn:aws:codecommit:*:*:*SageMaker*",
            "arn:aws:codecommit:*:*:*Sagemaker*"
        ]
    },
    {
        "Action": [
            "codebuild:BatchGetBuilds",
            "codebuild:StartBuild"
        ],
        "Resource": [
            "arn:aws:codebuild:*:*:project/sagemaker*",
            "arn:aws:codebuild:*:*:build/*"
        ],
        "Effect": "Allow"
    },
    {
        "Action": [
            "states:DescribeExecution",
            "states:GetExecutionHistory",
            "states:StartExecution",
            "states:StopExecution",
            "states:UpdateStateMachine"
        ],
        "Resource": [
            "arn:aws:states:*:*:statemachine:*sagemaker*",
            "arn:aws:states:*:*:execution:*sagemaker*:*"
        ],
        "Effect": "Allow"
    },
    {
        "Effect": "Allow",
        "Action": [
            "secretsmanager:DescribeSecret",
```

```
            "secretsmanager:GetSecretValue",
            "secretsmanager:CreateSecret"
        ],
        "Resource": [
            "arn:aws:secretsmanager:*:*:secret:AmazonSageMaker-*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "secretsmanager:DescribeSecret",
            "secretsmanager:GetSecretValue"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "secretsmanager:ResourceTag/SageMaker": "true"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "servicecatalog:ProvisionProduct"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "servicecatalog:TerminateProvisionedProduct",
            "servicecatalog:UpdateProvisionedProduct"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "servicecatalog:userLevel": "self"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:PutObject",
            "s3:DeleteObject",
            "s3:AbortMultipartUpload"
        ],
        "Resource": [
            "arn:aws:s3:::*SageMaker*",
            "arn:aws:s3:::*Sagemaker*",
            "arn:aws:s3:::*sagemaker*",
            "arn:aws:s3:::*aws-glue*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject"
        ],
        "Resource": "*",
        "Condition": {
            "StringEqualsIgnoreCase": {
                "s3:ExistingObjectTag/SageMaker": "true"
            }
        }
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/servicecatalog:provisioning": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetBucketCors",
                "s3:PutBucketCors"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketAcl",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::*SageMaker*",
                "arn:aws:s3:::*Sagemaker*",
                "arn:aws:s3:::*sagemaker*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:function:*SageMaker*",
                "arn:aws:lambda:*:*:function:*sagemaker*",
                "arn:aws:lambda:*:*:function:*Sagemaker*",
                "arn:aws:lambda:*:*:function:*LabelingFunction*"
            ]
        },
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/aws-service-role/sagemaker.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "sagemaker.application-autoscaling.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                "iam:AWSServiceName": "robomaker.amazonaws.com"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "sns:Subscribe",
            "sns:CreateTopic",
            "sns:Publish"
        ],
        "Resource": [
            "arn:aws:sns:*:*:*SageMaker*",
            "arn:aws:sns:*:*:*Sagemaker*",
            "arn:aws:sns:*:*:*sagemaker*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": "arn:aws:iam::*:role/*AmazonSageMaker*",
        "Condition": {
            "StringEquals": {
                "iam:PassedToService": [
                    "glue.amazonaws.com",
                    "robomaker.amazonaws.com",
                    "states.amazonaws.com"
                ]
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": "arn:aws:iam::*:role/*",
        "Condition": {
            "StringEquals": {
                "iam:PassedToService": "sagemaker.amazonaws.com"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "athena:ListDataCatalogs",
            "athena:ListDatabases",
            "athena:ListTableMetadata",
            "athena:GetQueryExecution",
            "athena:GetQueryResults",
            "athena:StartQueryExecution",
            "athena:StopQueryExecution"
        ],
        "Resource": [
            "*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:CreateTable"
        ],
        "Resource": [
```

```
            "arn:aws:glue:*:*:table/*/sagemaker_tmp_*",
            "arn:aws:glue:*:*:table/sagemaker_featurestore/*",
            "arn:aws:glue:*:*:catalog",
            "arn:aws:glue:*:*:database/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:DeleteTable"
        ],
        "Resource": [
            "arn:aws:glue:*:*:table/*/sagemaker_tmp_*",
            "arn:aws:glue:*:*:catalog",
            "arn:aws:glue:*:*:database/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:GetDatabases",
            "glue:GetTable",
            "glue:GetTables"
        ],
        "Resource": [
            "arn:aws:glue:*:*:table/*",
            "arn:aws:glue:*:*:catalog",
            "arn:aws:glue:*:*:database/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:CreateDatabase",
            "glue:GetDatabase"
        ],
        "Resource": [
            "arn:aws:glue:*:*:catalog",
            "arn:aws:glue:*:*:database/sagemaker_featurestore",
            "arn:aws:glue:*:*:database/sagemaker_processing",
            "arn:aws:glue:*:*:database/default",
            "arn:aws:glue:*:*:database/sagemaker_data_wrangler"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "redshift-data:ExecuteStatement",
            "redshift-data:DescribeStatement",
            "redshift-data:CancelStatement",
            "redshift-data:GetStatementResult",
            "redshift-data:ListSchemas",
            "redshift-data:ListTables"
        ],
        "Resource": [
            "*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "redshift:GetClusterCredentials"
        ],
        "Resource": [
            "arn:aws:redshift:*:*:dbuser:*/sagemaker_access*",
            "arn:aws:redshift:*:*:dbname:*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:ListStackResources"
            ],
            "Resource": "arn:aws:cloudformation:*:*:stack/SC-*"
        }
    ]
}
```

Amazon托管策略：AmazonSageMakerReadOnly

此策略授予对亚马逊的只读访问 SageMaker 通过Amazon Web Services Management Console和开发工具包。

权限详细信息

此策略包含以下权限。

- `application-autoscaling`— 允许用户浏览可扩展的描述 SageMaker 实时推理终端节点。
- `aws-marketplace`— 允许用户查看Amazon订阅 AI Marketplace。
- `cloudwatch`— 允许用户接收 CloudWatch 警报。
- `cognito-idp`— 亚马逊需要 SageMaker Ground Truth 浏览私有人力和工作团队的描述和列表。
- `ecr`— 需要阅读 Docker 项目以进行培训和推理。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:Describe*",
                "sagemaker:List*",
                "sagemaker:BatchGetMetrics",
                "sagemaker:GetDeviceRegistration",
                "sagemaker:GetDeviceFleetReport",
                "sagemaker:GetSearchSuggestions",
                "sagemaker:BatchGetRecord",
                "sagemaker:GetRecord",
                "sagemaker:Search",
                "sagemaker:QueryLineage",
                "sagemaker:GetLineageGroupPolicy",
                "sagemaker:BatchDescribeModelPackage",
                "sagemaker:GetModelPackageGroupPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DescribeScalableTargets",
                "application-autoscaling:DescribeScalingActivities",
                "application-autoscaling:DescribeScalingPolicies",
                "application-autoscaling:DescribeScheduledActions",
                "aws-marketplace:ViewSubscriptions",
                "cloudwatch:DescribeAlarms",
                "cognito-idp:DescribeUserPool",
                "cognito-idp:DescribeUserPoolClient",
```

```
                "cognito-idp:ListGroups",
                "cognito-idp:ListIdentityProviders",
                "cognito-idp:ListUserPoolClients",
                "cognito-idp:ListUserPools",
                "cognito-idp:ListUsers",
                "cognito-idp:ListUsersInGroup",
                "ecr:Describe*"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon适用于 Amazon 的托管策略 SageMaker Ground Truth

这些Amazon托管策略添加使用所需的权限 SageMaker Ground Truth。这些策略可在您的中找到Amazon账户，并由从 SageMaker 控制台。

主题

- Amazon托管策略：AmazonSageMakerGroundTruthExecution (p. 2765)
- 亚马逊 SageMaker SageMaker Ground Truth 托管策略的更新 (p. 2768)

Amazon托管策略：AmazonSageMakerGroundTruthExecution

该Amazon托管策略授予通常需要使用的权限 SageMaker Ground Truth。

权限详细信息

此策略包含以下权限。

- `lambda`— 允许委托人调用名称包括 “sagemaker”（不区分大小写）、“GTPefic” 或 “labelingFunction” 的 Lambda 函数。
- `s3`— 允许委托人从 Amazon S3 存储桶添加和检索对象。这些对象仅限于那些不区分大小写的名称包含 “groundtruth” 或 “sagemaker” 或者标记为 “SageMaker” 的对象。
- `cloudwatch`— 允许校长发布 CloudWatch 指标。
- `logs`— 允许委托人创建和访问日志流以及发布日志事件。
- `sqs`— 允许委托人创建 Amazon SQS 队列，以及发送和接收 Amazon SQS 消息。这些权限仅限于名称包含 “GroundTruth” 的队列。
- `sns`— 允许委托人订阅并向其发布消息，其名称不区分大小写的名称包含 “groundtruth” 或 “sagemaker” 的 Amazon SNS 主题。
- `ec2`— 创建、描述和删除 VPC 终端节点服务名称包含 “sagemaker-Task-资源” 或 “标签” 的 Amazon VPC 终端节点的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CustomLabelingJobs",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:function:*GtRecipe*",
                "arn:aws:lambda:*:*:function:*LabelingFunction*",
                "arn:aws:lambda:*:*:function:*SageMaker*",
```

```
            "arn:aws:lambda:*:*:function:*sagemaker*",
            "arn:aws:lambda:*:*:function:*Sagemaker*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:AbortMultipartUpload",
            "s3:GetObject",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::*GroundTruth*",
            "arn:aws:s3:::*Groundtruth*",
            "arn:aws:s3:::*groundtruth*",
            "arn:aws:s3:::*SageMaker*",
            "arn:aws:s3:::*Sagemaker*",
            "arn:aws:s3:::*sagemaker*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject"
        ],
        "Resource": "*",
        "Condition": {
            "StringEqualsIgnoreCase": {
                "s3:ExistingObjectTag/SageMaker": "true"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3:ListBucket"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatch",
        "Effect": "Allow",
        "Action": [
            "cloudwatch:PutMetricData",
            "logs:CreateLogStream",
            "logs:CreateLogGroup",
            "logs:DescribeLogStreams",
            "logs:PutLogEvents"
        ],
        "Resource": "*"
    },
    {
        "Sid": "StreamingQueue",
        "Effect": "Allow",
        "Action": [
            "sqs:CreateQueue",
            "sqs:DeleteMessage",
            "sqs:GetQueueAttributes",
            "sqs:GetQueueUrl",
            "sqs:ReceiveMessage",
            "sqs:SendMessage",
            "sqs:SetQueueAttributes"
        ],
        "Resource": "arn:aws:sqs:*:*:*GroundTruth*"
    },
```

```
        {
            "Sid": "StreamingTopicSubscribe",
            "Effect": "Allow",
            "Action": "sns:Subscribe",
            "Resource": [
                "arn:aws:sns:*:*:*GroundTruth*",
                "arn:aws:sns:*:*:*Groundtruth*",
                "arn:aws:sns:*:*:*groundTruth*",
                "arn:aws:sns:*:*:*groundtruth*",
                "arn:aws:sns:*:*:*SageMaker*",
                "arn:aws:sns:*:*:*Sagemaker*",
                "arn:aws:sns:*:*:*sageMaker*",
                "arn:aws:sns:*:*:*sagemaker*"
            ],
            "Condition": {
                "StringEquals": {
                    "sns:Protocol": "sqs"
                },
                "StringLike": {
                    "sns:Endpoint": "arn:aws:sqs:*:*:*GroundTruth*"
                }
            }
        },
        {
            "Sid": "StreamingTopic",
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:*:*:*GroundTruth*",
                "arn:aws:sns:*:*:*Groundtruth*",
                "arn:aws:sns:*:*:*groundTruth*",
                "arn:aws:sns:*:*:*groundtruth*",
                "arn:aws:sns:*:*:*SageMaker*",
                "arn:aws:sns:*:*:*Sagemaker*",
                "arn:aws:sns:*:*:*sageMaker*",
                "arn:aws:sns:*:*:*sagemaker*"
            ]
        },
        {
            "Sid": "StreamingTopicUnsubscribe",
            "Effect": "Allow",
            "Action": [
                "sns:Unsubscribe"
            ],
            "Resource": "*"
        },
        {
            "Sid": "WorkforceVPC",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeVpcEndpoints",
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "*",
            "Condition": {
                "StringLikeIfExists": {
                    "ec2:VpceServiceName": [
                        "*sagemaker-task-resources*",
                        "aws.sagemaker*labeling*"
                    ]
                }
            }
        }
```

```
    ]
}
```

亚马逊 SageMaker SageMaker Ground Truth 托管策略的更新

查看有关更新的详细信息Amazon适用于 Amazon 的托管策略 SageMaker 从该服务开始跟踪这些更改开始，Ground Truth True。要获得有关此页面更改的自动提示，请订阅上的 RSS 源。 SageMaker 文档历史记录页面。 (p. 2837)

策略	版本	更改	日期
AmazonSageMakerGroundTruthExecution	3	Add`ec2:CreateVpcEndpoint`、`ec2:DescribeVpcEndpoints`和`ec2:DeleteVpcEndpoints`权限。	2022 年 5 月 29 日
AmazonSageMakerGroundTruthExecution	2	Remove`sqs:SendMessageBatch`权限。	2022 年 4 月 11 日
AmazonSageMakerGroundTruthExecution	1	初始策略	2020 年 7 月 20 日

Amazon适用于 的 托管策略 SageMaker 管道

这些Amazon托管策略添加使用所需的权限 SageMaker 管道。这些策略可在您的中找到Amazon账户，并由从 SageMaker 控制台。

主题

- Amazon托管策略：AmazonSageMakerPipelinesIntegrations (p. 2768)
- 亚马逊 SageMaker SageMaker Pipeline 托管策略的更新 (p. 2769)

Amazon托管策略：AmazonSageMakerPipelinesIntegrations

该Amazon托管策略授予在中使用回调步骤和 Lambda 步骤所需的权限 SageMaker 管道。该策略已添加到`AmazonSageMaker-ExecutionRole`这是您在亚马逊上机时创建的 SageMaker Studio。可以将策略附加到用于创作或执行管道的任何角色。

此政策授予适当的Amazon构建调用 Lambda 函数或包含回调步骤的管道时所需的 Lambda、Amazon Simple Queue Service (Amazon SQS) 和 IAM 权限，这些管道可用于手动批准步骤或运行自定义工作负载。

Amazon SQS 权限允许您创建接收回调消息所需的 Amazon SQS 队列，也可以将消息发送到该队列。

Lambda 权限允许您创建、更新和删除管道步骤中使用的 Lambda 函数，还可以调用这些 Lambda 函数。

权限详细信息

此策略包含以下权限。

- `lambda`— 创建、删除和调用 Lambda 函数；更新 Lambda 函数代码。这些权限仅限于名称包括 “sagemaker” 的函数。
- `sqs`— 创建 Amazon SQS 队列；发送 Amazon SQS 消息。这些权限仅限于名称包括 “sagemaker” 的队列。
- `iam`— 将 IAM 角色传递给AmazonLambda 服务。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:CreateFunction",
                "lambda:DeleteFunction",
                "lambda:InvokeFunction",
                "lambda:UpdateFunctionCode"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:function:*sagemaker*",
                "arn:aws:lambda:*:*:function:*sageMaker*",
                "arn:aws:lambda:*:*:function:*SageMaker*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:CreateQueue",
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:*:*:*sagemaker*",
                "arn:aws:sqs:*:*:*sageMaker*",
                "arn:aws:sqs:*:*:*SageMaker*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lambda.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

亚马逊 SageMaker SageMaker Pipeline 托管策略的更新

查看有关更新的详细信息Amazon适用于 Amazon 的托管策略 SageMaker 因为该服务开始跟踪这些更改。要获得有关此页面更改的自动提示，请订阅上的 RSS 源。 SageMaker 文档历史记录页面。 (p. 2837)

策略	版本	更改	日期
AmazonSageMakerPipelinesIntegrations	1	初始策略	2021 年 7 月 30 日

Amazon适用于 的 托管策略 SageMaker 项目和 JumpStart

这些Amazon托管策略添加使用内置亚马逊的权限 SageMaker 项目模板和 JumpStart 解决方案。这些策略可在您的中找到Amazon账户，并由从 SageMaker 控制台。

SageMaker 项目和 JumpStart 使用Amazon要配置的 Service CatalogAmazon客户账户中的资源。一些创建的资源需要担任执行角色。例如，如果Amazon创建 “Service Catalog” CodePipeline 管道代表客户获得 SageMaker 机器学习 CI/CD 项目，那么该管道需要 IAM 角色。

这些区域有：AmazonSageMaker 服务目录产品发布角色角色具有启动 SageMaker 来自的产品组合AmazonService Catalog。这些区域有：AmazonSageMaker 服务目录产品 Ssuserole角色具有使用所需的权限 SageMaker 来自的产品组合AmazonService Catalog。这些区域有：`AmazonSageMakerServiceCatalogProductsLaunchRole`传递角色传递`AmazonSageMakerServiceCatalogProductsUseRole`角色到已置备AmazonService Catalog 产品资源。

主题

- Amazon托管策略：AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy (p. 2770)
- 亚马逊 SageMaker 更新到AmazonService CatalogAmazon托管策略 (p. 2777)

Amazon托管策略：AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy

此服务角色策略由Amazon Service Catalog预配亚马逊商品的服务 SageMaker 产品组合。策略向一组相关授予权限Amazon服务包括Amazon CodePipeline、Amazon CodeBuild、Amazon CodeCommit、AmazonGlue、Amazon CloudFront 等。

这些区域有：`AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy`策略旨在供`AmazonSageMakerServiceCatalogProductsLaunchRole`从创建的角色 SageMaker 控制台。该策略添加了资源调配的权限Amazon用于的资源 SageMaker 项目和 JumpStart 使用Amazon Service Catalog转入客户的账户。

权限详细信息

此策略包含以下权限。

- `apigateway`— 允许角色调用带有标记的 API Gateway 终端节点`sagemaker:launch-source`.
- `cloudformation`— 允许Amazon Service Catalog创建、更新和删除 CloudFormation 堆栈。
- `codebuild`— 允许担任的角色Amazon Service Catalog并传递到 CloudFormation 创建、更新和删除 CodeBuild 项目。
- `codecommit`— 允许担任的角色Amazon Service Catalog并传递到 CloudFormation 创建、更新和删除 CodeCommit 存储库。
- `codepipeline`— 允许担任的角色Amazon Service Catalog并传递到 CloudFormation 以创建、更新和删除 CodePipeline。
- `codestar-connections`— 允许角色通过 AWS CodeStar 连接。
- `cognito-idp`— 允许该角色创建、更新和删除组和用户池。还允许为资源添加标记。
- `ecr`— 允许担任的角色Amazon Service Catalog并传递到 CloudFormation 以创建和删除亚马逊 ECR 存储库。还允许为资源添加标记。
- `events`— 允许担任的角色Amazon Service Catalog并传递到 CloudFormation 要创建和删除 EventBridge 规则。用于将 CICD 管道的各个组件捆绑在一起。
- `firehose`— 允许角色与 Kinesis Data Firehose 流进行交互。
- `glue`— 允许角色与Amazon Glue.
- `iam`— 允许角色传递前面的角色`AmazonSageMakerServiceCatalog`. 当项目提供Amazon Service Catalog产品，因为角色需要传递给Amazon Service Catalog.
- `lambda`— 允许角色与Amazon Lambda.
- `logs`— 允许角色创建、删除和访问日志流。
- `s3`— 允许担任的角色Amazon Service Catalog并传递到 CloudFormation 以访问存储项目模板代码的 Amazon S3 存储桶。
- `sagemaker`— 允许角色与各种各样的互动 SageMaker 服务。这两种方式都是完成的 CloudFormation 在模板置备期间以及在 CodeBuild 在 CICD 管道执行期间。
- `states`— 允许角色创建、删除和更新随附的 Step Functions 创建、删除和更新步骤函数。`sagemaker`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:POST",
        "apigateway:PUT",
        "apigateway:PATCH",
        "apigateway:DELETE"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/sagemaker:launch-source": "*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:POST"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": [
            "sagemaker:launch-source"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/account"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:UpdateStack",
        "cloudformation:DeleteStack"
      ],
      "Resource": "arn:aws:cloudformation:*:*:stack/SC-*",
      "Condition": {
        "ArnLikeIfExists": {
          "cloudformation:RoleArn": [
            "arn:aws:sts::*:assumed-role/AmazonSageMakerServiceCatalog*"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStacks"
      ],
      "Resource": "arn:aws:cloudformation:*:*:stack/SC-*"
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:GetTemplateSummary",
        "cloudformation:ValidateTemplate"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:CreateProject",
        "codebuild:DeleteProject",
        "codebuild:UpdateProject"
      ],
      "Resource": [
        "arn:aws:codebuild:*:*:project/sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:CreateCommit",
        "codecommit:CreateRepository",
        "codecommit:DeleteRepository",
        "codecommit:GetRepository",
        "codecommit:TagResource"
      ],
      "Resource": [
        "arn:aws:codecommit:*:*:sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:ListRepositories"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:CreatePipeline",
        "codepipeline:DeletePipeline",
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:StartPipelineExecution",
        "codepipeline:TagResource",
        "codepipeline:UpdatePipeline"
      ],
      "Resource": [
        "arn:aws:codepipeline:*:*:sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "cognito-idp:CreateUserPool",
        "cognito-idp:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": [
            "sagemaker:launch-source"
```

```
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "cognito-idp:CreateGroup",
        "cognito-idp:CreateUserPoolDomain",
        "cognito-idp:CreateUserPoolClient",
        "cognito-idp:DeleteGroup",
        "cognito-idp:DeleteUserPool",
        "cognito-idp:DeleteUserPoolClient",
        "cognito-idp:DeleteUserPoolDomain",
        "cognito-idp:DescribeUserPool",
        "cognito-idp:DescribeUserPoolClient",
        "cognito-idp:UpdateUserPool",
        "cognito-idp:UpdateUserPoolClient"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/sagemaker:launch-source": "*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecr:CreateRepository",
        "ecr:DeleteRepository",
        "ecr:TagResource"
      ],
      "Resource": [
        "arn:aws:ecr:*:*:repository/sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule",
        "events:DeleteRule",
        "events:DisableRule",
        "events:EnableRule",
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets"
      ],
      "Resource": [
        "arn:aws:events:*:*:rule/sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "firehose:CreateDeliveryStream",
        "firehose:DeleteDeliveryStream",
        "firehose:DescribeDeliveryStream",
        "firehose:StartDeliveryStreamEncryption",
        "firehose:StopDeliveryStreamEncryption",
        "firehose:UpdateDestination"
      ],
      "Resource": "arn:aws:firehose:*:*:deliverystream/sagemaker-*"
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "glue:CreateDatabase",
        "glue:DeleteDatabase"
      ],
      "Resource": [
        "arn:aws:glue:*:*:catalog",
        "arn:aws:glue:*:*:database/sagemaker-*",
        "arn:aws:glue:*:*:table/sagemaker-*",
        "arn:aws:glue:*:*:userDefinedFunction/sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "glue:CreateClassifier",
        "glue:DeleteClassifier",
        "glue:DeleteCrawler",
        "glue:DeleteJob",
        "glue:DeleteTrigger",
        "glue:DeleteWorkflow",
        "glue:StopCrawler"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "glue:CreateWorkflow"
      ],
      "Resource": [
        "arn:aws:glue:*:*:workflow/sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "glue:CreateJob"
      ],
      "Resource": [
        "arn:aws:glue:*:*:job/sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "glue:CreateCrawler",
        "glue:GetCrawler"
      ],
      "Resource": [
        "arn:aws:glue:*:*:crawler/sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "glue:CreateTrigger",
        "glue:GetTrigger"
      ],
      "Resource": [
        "arn:aws:glue:*:*:trigger/sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/service-role/AmazonSageMakerServiceCatalog*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "lambda:AddPermission",
        "lambda:CreateFunction",
        "lambda:DeleteFunction",
        "lambda:GetFunction",
        "lambda:GetFunctionConfiguration",
        "lambda:InvokeFunction",
        "lambda:RemovePermission"
      ],
      "Resource": [
        "arn:aws:lambda:*:*:function:sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DeleteLogGroup",
        "logs:DeleteLogStream",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams",
        "logs:PutRetentionPolicy"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/apigateway/AccessLogs/*",
        "arn:aws:logs:*:*:log-group::log-stream:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "s3:ExistingObjectTag/servicecatalog:provisioning": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": [
        "arn:aws:s3:::sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:DeleteBucketPolicy",
        "s3:GetBucketPolicy",
        "s3:PutBucketAcl",
        "s3:PutBucketNotification",
        "s3:PutBucketPolicy",
        "s3:PutBucketPublicAccessBlock",
```

```
        "s3:PutBucketLogging",
        "s3:PutEncryptionConfiguration",
        "s3:PutBucketTagging",
        "s3:PutObjectTagging",
        "s3:PutBucketCORS"
      ],
      "Resource": "arn:aws:s3:::sagemaker-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateEndpoint",
        "sagemaker:CreateEndpointConfig",
        "sagemaker:CreateModel",
        "sagemaker:CreateWorkteam",
        "sagemaker:DeleteEndpoint",
        "sagemaker:DeleteEndpointConfig",
        "sagemaker:DeleteModel",
        "sagemaker:DeleteWorkteam",
        "sagemaker:DescribeModel",
        "sagemaker:DescribeEndpointConfig",
        "sagemaker:DescribeEndpoint",
        "sagemaker:DescribeWorkteam",
        "sagemaker:CreateCodeRepository",
        "sagemaker:DescribeCodeRepository",
        "sagemaker:UpdateCodeRepository",
        "sagemaker:DeleteCodeRepository"
      ],
      "Resource": [
        "arn:aws:sagemaker:*:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateImage",
        "sagemaker:DeleteImage",
        "sagemaker:DescribeImage",
        "sagemaker:UpdateImage",
        "sagemaker:ListTags"
      ],
      "Resource": [
        "arn:aws:sagemaker:*:*:image/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:CreateStateMachine",
        "states:DeleteStateMachine",
        "states:UpdateStateMachine"
      ],
      "Resource": [
        "arn:aws:states:*:*:stateMachine:sagemaker-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "codestar-connections:PassConnection",
      "Resource": "arn:aws:codestar-connections:*:*:connection/*",
      "Condition": {
        "StringEquals": {
          "codestar-connections:PassedToService": "codepipeline.amazonaws.com"
        }
      }
    }
```

```
  ]
}
```

亚马逊 SageMaker 更新到AmazonService CatalogAmazon托管策略

查看有关更新的详细信息Amazon适用于 Amazon 的托管策略 SageMaker 因为该服务开始跟踪这些更改。要获得有关此页面更改的自动提示，请订阅上的 RSS 源。 SageMaker 文档历史记录页面。 (p. 2837)

策略	版本	更改	日期
AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy (p. 2770)	5	添加新权限`ecr-idp:TagResource`.	2022 年 3 月 21 日
AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy	4	添加的新权限`cognito-idp:TagResource`和`s3:PutBucketCORS`.	2022 年 2 月 16 日
AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy	3	添加的新权限`sagemaker`. 创建、读取、更新和删除 SageMaker 映像。	2021 年 9 月 15 日
AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy	2	添加的新权限`sagemaker`和`codestar-connections`. 创建、读取、更新和删除代码存储库。 传递 AWS CodeStar 连接到 Amazon CodePipeline.	2021 年 7 月 1 日
AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy	1	初始策略	2020 年 11 月 27 日

SageMaker 更新Amazon管理的策略

查看有关更新的详细信息Amazon适用于 的托管策略 SageMaker 因为该服务开始跟踪这些更改。要获得有关此页面更改的自动提示，请订阅上的 RSS 源。SageMaker 文档记录页.

策略	版本	更改	日期
AmazonSageMakerFullAccess (p. 2755)	22	添加`cloudformation:ListStackResources`.	2022 年 5 月 1 日
AmazonSageMakerReadOnly (p. 2764)	11	Add`sagemaker:QueryLineage`、`sagemaker:GetLineageGroupPol`权限。	2021 年 12 月 1 日
AmazonSageMakerFullAccess	21	Add`sns:Publish`启用了异步推理的终端的权限。	2021 年 9 月 8 日
AmazonSageMakerFullAccess	20	更新`iam:PassRole`资源和权限。	2021 年 7 月 15 日
AmazonSageMakerReadOnly	10	新 API`BatchGetRecord`增加了 SageMaker 功能商店。	2021 年 6 月 10 日

策略	版本	更改	日期
		SageMaker 开始跟踪其的更改Amazon托管策略。	2021 年 6 月 1 日

排查 Amazon SageMaker 身份和访问权限

使用以下信息可帮助您诊断和修复在使用时可能遇到的常见问题。 SageMaker 和 IAM。

主题

- 我无权在 SageMaker 中执行操作 (p. 2778)
- 我无权执行 iam:PassRole (p. 2778)
- 我想要查看我的访问密钥 (p. 2778)
- 我是管理员并希望允许其他人访问 SageMaker (p. 2779)
- 我想要允许我的人员在我之外Amazon访问我的 SageMaker 资源的帐户 (p. 2779)

我无权在 SageMaker 中执行操作

如果 Amazon Web Services Management Console 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。

下面的示例错误发生在`mateojackson`IAM 用户尝试使用控制台查看有关训练作业的详细信息，但没有`sagemaker:sagemaker:DescribeTrainingJob`权限。

```
User: arn:aws:iam::123456789012:user/mateojackson is not
            authorized to perform: sagemaker:DescribeTrainingJob on resource: my-example-
widget
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `sagemaker:DescribeTrainingJob` 操作访问 `TrainingJob` 资源。

我无权执行 iam:PassRole

如果您收到错误消息，提示您无权执行 `iam:PassRole` 操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。请求该人员更新您的策略，以便允许您将角色传递给 SageMaker。

有些 Amazon Web Services允许您将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的 IAM 用户时，会发生以下示例错误：`marymajor`尝试使用控制台在 SageMaker 中执行操作。但是，服务必须具有服务角色所授予的权限才可执行操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

在这种情况下，Mary 请求她的管理员来更新其策略，以允许她执行 `iam:PassRole` 操作。

我想要查看我的访问密钥

在创建 IAM 用户访问密钥后，您可以随时查看您的访问密钥 ID。但是，您无法再查看您的秘密访问密钥。如果您丢失了私有密钥，则必须创建一个新的访问密钥对。

访问密钥包含两部分：访问密钥 ID（例如 `AKIAIOSFODNN7EXAMPLE`）和秘密访问密钥（例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

Important

请不要向第三方提供访问密钥，即便是为了帮助找到您的规范用户 ID 也不行。如果您这样做，可能会向某人提供对您的账户的永久访问权限。

当您创建访问密钥对时，系统会提示您将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果丢失了您的秘密访问密钥，您必须为 IAM 用户添加新的访问密钥。您最多可拥有两个访问密钥。如果您已有两个密钥，则必须删除一个密钥对，然后再创建新的密钥。要查看说明，请参阅 IAM 用户指南中的管理访问密钥。

我是管理员并希望允许其他人访问 SageMaker

要允许其他人访问 SageMaker，您必须为需要访问权限的人员或应用程序创建一个 IAM 实体（用户或角色）。它们将使用该实体的凭证访问 Amazon。然后，您必须将策略附加到实体，以便在 SageMaker 中向其授予正确的权限。

要立即开始使用，请参阅《IAM 用户指南》中的创建您的第一个 IAM 委派用户和组。

我想要允许我的人员在我之外Amazon访问我的 SageMaker 资源的帐户

您可以创建一个角色，以便其它账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 了解是否 SageMaker 支持这些功能，请参阅Amazon 的方式 SageMaker 与 IAM 协同工作 (p. 2694).
- 要了解如何为您拥有的 Amazon Web Services 账户 中的资源提供访问权限，请参阅 IAM 用户指南中的为您拥有的另一个 Amazon Web Services 账户 中的 IAM 用户提供访问权限。
- 要了解如何为第三方Amazon Web Services 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的Amazon Web Services 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

日志记录和监控

您可以监控亚马逊 SageMaker 使用 Amazon CloudWatch，后者收集原始数据，并将数据处理为便于读取的近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 使用 Amazon CloudWatch 监控 Amazon SageMaker (p. 2810)。

亚马逊 CloudWatch 使您能够监控、存储和访问来自 Amazon EC2 实例的日志文件。Amazon CloudTrail和其他来源的数据。您可以收集和跟踪指标，创建自定义的控制面板，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。 CloudWatch 日志可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 使用 Amazon CloudWatch 记录 Amazon SageMaker 事件 (p. 2820)。

Amazon CloudTrail提供用户、角色或用户所执行操作的记录。AmazonSageMaker 中的服务。通过使用 CloudTrail 收集的信息，您可以确定向 SageMaker 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。有关更多信息，请参阅 使用记录 Amazon SageMaker API 调用 Amazon CloudTrail (p. 2821)。

Note

CloudTrail 不监控对的调用`runtime_InvokeEndpoint`.

您可以在 Amazon 中创建规则 CloudWatch 对 SageMaker 训练、超参数调整或批转换作业中的状态更改做出反应的事件。有关更多信息，请参阅 使用 Amazon EventBridge 自动执行 Amazon SageMaker (p. 2823)。

Amazon SageMaker 的合规性验证

第三方审计员评估 Amazon 的安全性和合规性。 SageMaker 作为多个组成部分Amazon合规性计划。其中包括 SOC、PCI、FedRAMP、HIPAA 及其它。

有关特定合规性计划范围内的Amazon服务的列表，请参阅合规性计划范围内的Amazon服务。有关一般信息，请参阅Amazon合规性计划。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅下载 Amazon Artifact 中的报告。

使用亚马逊时的合规责任 SageMaker 由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。Amazon提供以下资源以帮助实现合规性：

- 《安全性与合规性快速入门指南》– 这些部署指南讨论了架构注意事项，并提供了在Amazon上部署基于安全性和合规性的基准环境的步骤。
- 设计符合 HIPAA 安全性和合规性要求的架构白皮书 – 此白皮书介绍公司如何使用 Amazon 创建符合 HIPAA 标准的应用程序。
- Amazon 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- Amazon Config – 此Amazon服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- Amazon Security Hub – 此 Amazon 服务提供了 Amazon 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

Amazon SageMaker 中的恢复能力

Amazon全球基础设施围绕Amazon区域和可用区构建。Amazon区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅 Amazon 全球基础设施。

除了Amazon亚马逊全球基础设施 SageMaker 提供了多种功能，以帮助支持您的数据弹性和备份需求。

Amazon SageMaker 的基础设施安全性

作为托管服务，亚马逊 SageMaker 受保护Amazon中描述的全局网络安全程序Amazon Web Services：安全过程概述白皮书。

你用Amazon发布 API 调用以访问亚马逊 SageMaker 通过网络。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon Security Token Service (Amazon STS) 生成临时安全凭证来对请求进行签名。

主题

- SageMaker 扫描Amazon Web Services Marketplace针对安全漏洞的训练和推理容器 (p. 2781)
- 从 VPC 内 Connect 到资源 (p. 2781)
- 在互联网免费模式下运行训练和推理容器 (p. 2784)
- 连接到 SageMaker 通过 VPC 接口终端节点节点 (p. 2785)
- 授予权 SageMaker 访问您的亚马逊 VPC 中的资源 (p. 2793)

SageMaker 扫描Amazon Web Services Marketplace针对安全漏洞的训练和推理容器

为满足我们的安全性要求，将对 Amazon Web Services Marketplace 中列出的算法和模型软件包进行扫描，以发现常见漏洞和风险 (CVE)。CVE 是有关安全漏洞和风险的公共已知信息的列表。国家漏洞数据库 (NVD) 提供了 CVE 详细信息，如严重性、影响评分和修复信息。CVE 和 NVD 都可供公众使用，安全工具和服务使用它们时免费。有关更多信息，请参阅 http://cve.mitre.org/about/faqs.html#what_is_cve。

从 VPC 内 Connect 到资源

亚马逊 SageMaker Studio 和 SageMaker 默认情况下，笔记本实例允许直接互联网 这样，您就可以下载流行的程序包和笔记本，自定义其开发环境并高效工作。不过，这可能为未经授权访问您的数据提供另一种途径。例如，如果您以公开发布的笔记本或公开发布的源代码库形式在计算机上安装恶意代码，则该计算机就可以访问您的数据。您可以选择通过启动亚马逊来限制哪些流量可以访问互联网 SageMaker Studio 和 SageMaker 中的笔记本实例Amazon Virtual Private Cloud (Amazon VPC)你选择的。

Amazon Virtual Private Cloud 是专供您使用的虚拟网络。Amazonaccount. 使用亚马逊 VPC，您可以控制亚马逊的网络访问和互联网连接 SageMaker 工作室和笔记本实例。您可以禁用直接互联网访问以添加额外的安全层。

以下主题介绍如何连接 SageMaker Studio 实例和笔记本实例转移到 VPC 中的资源。

主题

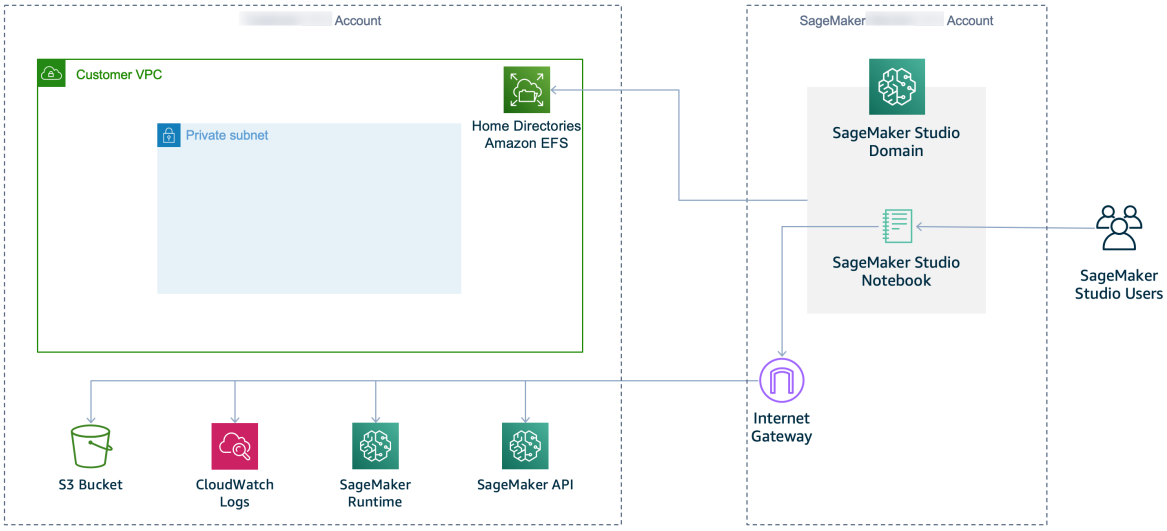
- Connect (连接) SageMaker VPC 中的 Studio 笔记本电脑到外部资源 (p. 2781)
- 将 VPC 中的笔记本实例 Connect 到外部资源 (p. 2783)

Connect (连接) SageMaker VPC 中的 Studio 笔记本电脑到外部资源

以下主题提供了有关如何将 VPC 中的 Studio Notebook 连接到外部资源的信息。

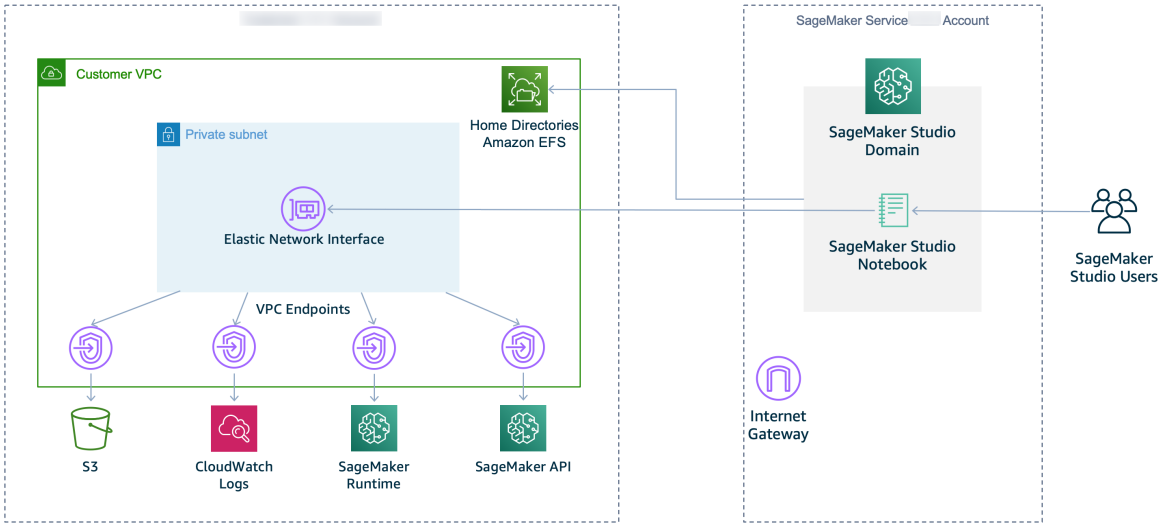
与互联网的默认通信

默认情况下， SageMaker Studio 提供一个网络接口，允许通过 SageMaker 管理的 VPC 与 Internet 通信。流量AmazonAmazon S3 和 CloudWatch 通过互联网网关，访问 SageMaker API 的流量也是如此 SageMaker 运行时间。域与 Amazon EFS 卷之间的流量通过您在加载到 Studio 时指定的 VPC 或调用CreateDomainAPI。下图显示了默认配置。



VPC only与互联网沟通

为了防止 SageMaker 从提供对 Studio 笔记本电脑的互联网访问，您可以通过指定`VPC only`当你时的网络访问类型注册到 Studio或者调用CreateDomainAPI。因此，除非您的 VPC 具有指向 SageMaker API 和运行时，或者可以访问互联网的 NAT 网关，并且您的安全组允许出站连接。下图显示了使用仅 VPC 模式的配置。



使用的要求VPC only模式

选择的时候`VpcOnly`，请按照以下步骤操作：

1. 确保您的子网每个实例都有一个 IP 地址。有关更多信息，请参阅 。针对 IPv4 的 VPC 和子网大小调整.

   Note

   您只能使用默认租赁 VPC 配置子网，其中您的实例在共享硬件上运行。有关 VPC 租赁属性的更多信息，请参阅专用实例。
2. 使用入站和出站规则设置一个或多个安全组，这些规则共同允许以下流量：

   - 2049 端口上通过 TCP 的 NFS 流量在域和 Amazon EFS 卷之间。

- 安全组内的 TCP 流量. 这对于在 JupyterServer 应用程序和 KernelGateway 应用程序。您必须允许访问范围内的至少端口`8192-65535`.

3. 如果要允许互联网访问，必须使用NAT 网关可以访问互联网，例如通过互联网网关.
4. 如果您不希望允许访问互联网，创建接口 VPC 终端节点(AmazonPrivateLink) 允许 Studio 使用相应的服务名称访问以下服务。您还必须将 VPC 的安全组与这些终端节点关联起来。

   - SageMaker API：`com.amazonaws.us-east-1.sagemaker.api`
   - SageMaker 运行时：`com.amazonaws.us-east-1.sagemaker.runtime`. 运行 Studio 笔记本以及训练和托管模型是必需的。
   - Amazon S3：`com.amazonaws.us-east-1.s3`.
   - 使用 SageMaker 项目：`com.amazonaws.us-east-1.servicecatalog`.
   - 任何格式Amazon你需要的服务。

   Note

   对于在 VPC 模式下工作的客户，公司防火墙可能会导致连接问题 SageMaker Studio 或介于 Studio JupyterServer 和 KernelGateway。使用时是否遇到以下问题之一，请进行以下检查： SageMaker 工作室来自防火墙后面。

   - 检查 Studio URL 是否在网络允许列表中。
   - 检查 websocket 连接是否未被阻止。Jupyter 在引擎盖下使用 websocket。如果 KernelGateway 应用程序 InService， JupyterServer 可能无法连接到 KernelGateway。打开系统终端时也应该看到这个问题。

有关更多信息

- 保护亚马逊 SageMaker 使用私有 VPC 进行工作室连接.
- 您的 VPC 的安全组
- 连接到 SageMaker 通过 VPC 接口终端节点节点 (p. 2785)
- 带有公有和私有子网的 VPC（NAT）

将 VPC 中的笔记本实例 Connect 到外部资源

以下主题提供了有关如何将 VPC 中的笔记本实例连接到外部资源的信息。

与互联网的默认通信

当你的笔记本允许直接 Internet 访问，SageMaker 提供了一个网络接口，允许笔记本通过 SageMaker 管理的 VPC 与 Internet 通信。VPC CIDR 中的流量将通过在 VPC 中创建的 elastic network interface。所有其他流量通过 SageMaker 创建的网络接口，该接口本质上是通过公有 Internet。到 Amazon S3 和 DynamoDB 等网关 VPC 终端节点的流量通过公有 Internet，而到接口 VPC 终端节点的流量仍通过您的 VPC。如果要使用网关 VPC 终端节点，您可能希望禁用直接 Internet 访问。

VPC 与互联网的通信

要禁用直接 Internet 访问，您可以为笔记本实例指定一个 VPC。通过这样做，你可以防止 SageMaker 从提供对笔记本实例的互联网访问。因此，除非您的 VPC 具有接口终端节点，否则，笔记本实例无法训练或托管模型 (Amazon PrivateLink) 或者 NAT 网关，您的安全组允许出站连接。

有关创建 VPC 接口终端节点以使用的信息Amazon PrivateLink对于你的笔记本实例，请参阅通过 VPC 接口终端节点连接到笔记本实例 (p. 2790). 有关为 VPC 设置 NAT 网关的信息，请参阅。带有公有和私有子网的 VPC (NAT)中的Amazon Virtual Private Cloud 用户指南. 有关安全组的信息，请参阅您的 VPC 的安全组。有关每种网络模式中的网络配置和在本地配置网络的更多信息，请参阅了解 Amazon SageMaker 笔记本实例网络配置和高级路由选项.

安全性和共享笔记本实例

一个 SageMaker 笔记本实例最适合单个用户。它旨在为数据科学家和其他用户提供管理开发环境的最强大功能。

笔记本实例用户具有安装程序包和其他相关软件的根访问权限。对于连接到包含敏感信息的 VPC 的笔记本实例，建议您在授予其相关的个人访问权限时谨慎处理。例如，您可以授予用户访问具有 IAM 策略的笔记本实例的权限，如以下示例所示：

```
{
  "Version": "2012-10-17",
  "Statement": [
   {
      "Effect": "Allow",
      "Action": "sagemaker:CreatePresignedNotebookInstanceUrl",
      "Resource": "arn:aws:sagemaker:region:account-id:notebook-instance/
myNotebookInstance"
   }
  ]
}
```

在互联网免费模式下运行训练和推理容器

默认情况下，SageMaker 训练和部署的推理容器启用 Internet。这允许容器访问外部服务和公共 Internet 上的资源，以作为训练和部署工作负载的一部分。但是，这可以为未经授权访问您的数据提供一个通道。例如，恶意用户或无意中安装在容器上的恶意代码（以公开发布的源代码库形式）就可以访问您的数据并将数据传输到远程主机。

如果 VPC 通过为`VpcConfig`调用时的参数`CreateTrainingJob`、`CreateHyperParameterTuningJob`，或者`CreateModel`，您可以通过管理安全组和限制从 VPC 访问 Internet 来保护您的数据和资源。但是，其代价是进行额外网络配置的成本，并且配置网络时有出错的风险。如果你不想 SageMaker 要提供对训练或推理容器的外部网络访问权限，您可以启用网络隔离。

网络隔离

您可以在创建训练作业或模型时通过设置`EnableNetworkIsolation`到参数`True`当你打电话`CreateTrainingJob`、`CreateHyperParameterTuningJob`，或者`CreateModel`.

> **Note**
>
> 使用以下的资源运行训练作业和模型时，需要网络隔离。Amazon Web Services Marketplace.

如果您启用网络隔离，容器无法进行任何出站网络调用，甚至对于其他容器也是如此Amazon诸如 Amazon S3 之类的服务。此外，没有任何 Amazon 凭证可用于容器运行时环境。对于具有多个实例的训练作业，入站和出站网络流量限制为每个训练容器的对等项。 SageMaker 仍然使用您的对 Amazon S3 执行下载和上传操作 SageMaker 独立于训练或推理容器的执行角色。

以下管理 SageMaker 容器不支持网络隔离，因为它们需要访问 Amazon S3：

- Chainer
- SageMaker 强化学习

使用 VPC 进行网络隔离

网络隔离可与 VPC 结合使用。在这种情况下，客户数据和模型构件的下载和上传将通过您的 VPC 子网进行路由。但是，训练和推理容器本身继续与网络隔离，并且对 VPC 内或 Internet 上的任何资源都没有访问权限。

连接到 SageMaker 通过 VPC 接口终端节点节点

你可以直接连接到 SageMaker API 或 SageMaker 运行时接口终端节点在您的 Virtual Private Cloud (VPC) 中，而不是通过 Internet 进行连接。当您使用 VPC 接口终端节点时，您的 VPC 与 SageMaker API 或运行时完全安全地在Amazon网络。

这些区域有： SageMaker API 和运行时支持Amazon Virtual Private Cloud(Amazon VPC) 接口终端节点由提供支持AmazonPrivateLink. 每个 VPC 终端节点均由一个或多个表示。弹性网络接口您的 VPC 子网中具有私有 IP 地址。

VPC 接口终端节点将您的 VPC 直接连接到 SageMaker 没有互联网网关、NAT 设备、VPN 连接的 API 或运行时Amazon Direct Connect连接。VPC 中的实例不需要公有 IP 地址便可与 SageMaker API 或运行时。

您可以创建接口终端节点以连接到 SageMaker 或者去 SageMaker 运行时使用Amazon控制台或Amazon Command Line Interface(Amazon CLI) 命令。有关说明，请参阅创建接口终端节点。

创建 VPC 终端节点后，您可以使用以下示例 CLI 命令，这些命令使用`endpoint-url`参数来指定接口终端节点 SageMaker API 或运行时：

```
aws sagemaker list-notebook-instances --endpoint-
url VPC_Endpoint_ID.api.sagemaker.Region.vpce.amazonaws.com

aws sagemaker list-training-jobs --endpoint-
url VPC_Endpoint_ID.api.sagemaker.Region.vpce.amazonaws.com

aws sagemaker-runtime invoke-endpoint --endpoint-
url VPC_Endpoint_ID.runtime.sagemaker.Region.vpce.amazonaws.com  \
    --endpoint-name Endpoint_Name \
    --body "Endpoint_Body" \
    --content-type "Content_Type" \
            Output_File
```

如果为 VPC 终端节点启用专用 DNS 主机名，您不需要指定终端节点 URL。这些区域有： SageMaker CLI 和的 API DNS 主机名 SageMaker 默认情况下 SDK 使用（ https://api.sagemaker.##.amazon.com) 解析为 VPC 终端节点。同样， SageMaker CLI 和 SageMaker 运行时开发工具包在默认情况下使用的运行时 DNS 主机名 (https://runtime.sagemaker.##.amazonaws.com) 解析为您的 VPC 终端节点。

这些区域有： SageMaker API 和运行时都支持 VPC 终端节点Amazon两者的区域Amazon VPC和SageMaker都可用。 SageMaker 支持打电话给它的所有`Operations`在您的 VPC 内。结果`AuthorizedUrl`来自 的`CreatePresignedNotebookInstanceUrl`不支持Amazon PrivateLink. 有关如何启用的信息Amazon PrivateLink有关用户用于连接到笔记本实例的授权 URL，请参阅。通过 VPC 接口终端节点连接到笔记本实例 (p. 2790).

了解相关更多信息Amazon PrivateLink，请参阅Amazon PrivateLink文档. 请参阅VPC 定价定价对 VPC 终端节点的价格。要了解有关 VPC 和终端节点的更多信息，请参阅 Amazon VPC。有关如何使用基于身份的信息Amazon Identity and Access Management使用策略来限制对 SageMaker API 和运行时，请参阅控制对的访问 SageMaker 使用基于身份的策略 API (p. 2713).

为 SageMaker 创建 VPC 终端节点策略

您可以为 Amazon VPC 终端节点创建一个策略， SageMaker 要指定以下内容：

- 可执行操作的委托人。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 终端节点控制对服务的访问。

Note

联邦信息处理标准 (FIPS) 不支持 VPC 终端节点策略 SageMaker 运行时终端节点`runtime_InvokeEndpoint`.

以下示例 VPC 终端节点策略指定有权访问 VPC 接口终端节点的所有用户都可以调用 SageMaker 名为的托管端点`myEndpoint`.

```
{
  "Statement": [
      {
          "Action": "sagemaker:InvokeEndpoint",
          "Effect": "Allow",
          "Resource": "arn:aws:sagemaker:us-west-2:123456789012:endpoint/myEndpoint",
          "Principal": "*"
      }
  ]
}
```

在本示例中，会拒绝以下操作：

- 其他 SageMaker 如 API 操作`sagemaker:CreateEndpoint`和`sagemaker:CreateTrainingJob`.
- 调用 SageMaker 以外的托管终端节点`myEndpoint`.

Note

在本示例中，用户仍然可以使用其他 SageMaker 来自 VPC 外部的 API 操作。有关如何将 API 调用限制为该 VPC 中执行的那些调用的信息，请参阅 控制对的访问 SageMaker 使用基于身份的策略 API (p. 2713)。

为 Amazon 创建 VPC 终端节点策略 SageMaker 功能商店

为 Amazon 创建 VPC 终端节点 SageMaker 要素商店，使用以下终端节点模板，替换*`vpc_endpoint_ID.api`*和*`##`*：

`VPC_Endpoint_ID.api``.featurestore-runtime.sagemaker.`*`Region`*`.vpce.amazonaws.com`

连接到 SageMaker Studio 通过接口 VPC 终端节点

你可以连接到亚马逊 SageMaker 来自你的工作室Amazon Virtual Private Cloud（亚马逊 VPC）通过接口终端节点在 VPC 中，而不是通过互联网连接。当您使用接口 VPC 终端节点（接口终端节点）时，VPC 与 Studio 之间的通信完全在Amazon网络。

支持 SageMaker Studio 支持接口终端节点，这些端点Amazon PrivateLink. 每个接口终端节点均由一个或多个表示弹性网络接口您的 VPC 子网中具有私有 IP 地址。

您可以创建接口终端节点以连接到 Studio，使用Amazon控制台或Amazon Command Line Interface(Amazon CLI)。有关说明，请参阅创建接口终端节点. 确保为 VPC 中要连接到 Studio 的所有子网创建接口终端节点。

创建接口终端节点时，请确保终端节点上的安全组允许来自 SageMaker Studio 关联的安全组访问 HTTPS 流量的入站访问。有关更多信息，请参阅 。使用 VPC 终端节点控制对服务的访问权限.

Note

除了创建要连接到的接口终端节点 SageMaker Studio，创建接口终端节点以连接到 Amazon SageMaker API。当用户打电话`CreatePresignedDomainUrl`要获取连接到 Studio 的 URL，该调用将通过用于连接到 SageMaker API 的接口终端节点。

创建接口终端节点时，请指定**aws.sagemaker.*Region*.studio**作为服务名称。在创建接口终端节点后，为您的终端节点启用私有 DNS。任何使用 SageMaker API，Amazon CLI，或者要连接到的控制台 SageMaker VPC 内的 Studio 通过接口终端节点而不是公共互联网连接到 Studio。您还需要为 Amazon VPC 终端节点设置带有私有托管区域的自定义 DNS，因此 SageMaker 工作室可以访问 SageMaker 使用 APIapi.sagemaker.$region.amazonaws.com终端节点而不是使用 VPC 终端节点 URL。有关设置私有托管区域的说明，请参阅。使用私有托管区域.

Studio 总的来说支持接口端点Amazon两者的区域Amazon SageMaker和Amazon VPC都可用。

主题

- 为 SageMaker Studio 创建 VPC 终端节点策略 (p. 2787)
- 仅允许从 VPC 中访问 (p. 2787)

为 SageMaker Studio 创建 VPC 终端节点策略

您可以将 Amazon VPC 终端节点策略附加到用于连接到的接口 VPC 终端节点 SageMaker Studio。端点策略控制对 Studio 的访问。您可以指定：

- 可执行操作的委托人。
- 可执行的操作。
- 可对其执行操作的资源。

将 VPC 终端节点与 SageMaker Studio，您的终端节点策略必须允许`CreateApp`在上操作 KernelGateway 应用程序类型。这允许通过 VPC 终端节点路由到的流量调用`CreateApp`API。以下示例 VPC 终端节点策略介绍了如何允许`CreateApp`operation.

```
{
 "Statement": [
   {
     "Action": "sagemaker:CreateApp",
     "Effect": "Allow",
     "Resource": "arn:aws:sagemaker:us-west-2:acct-id:user-profile/domain-id/*",
     "Principal": "*"
   }
 ]
}
```

有关更多信息，请参阅使用 VPC 终端节点控制对服务的访问。

以下 VPC 终端节点策略示例指定有权访问该终端节点的所有用户都可以访问 SageMaker 具有指定域 ID 的 Studio 域名。访问其他域名时会被拒绝。

```
{
  "Statement": [
      {
          "Action": "sagemaker:CreatePresignedDomainUrl",
          "Effect": "Allow",
          "Resource": "arn:aws:sagemaker:us-west-2:acct-id:user-profile/domain-id/*",
          "Principal": "*"
      }
  ]
}
```

仅允许从 VPC 中访问

VPC 之外的用户可以连接到 SageMaker 即使您在 VPC 中设置了接口终端节点，您也可以通过 Internet 使用 Studio。

要仅允许访问从您 VPC 中进行的连接，请创建Amazon Identity and Access Management这方面的 (IAM) 政策。将该策略添加到用于访问 Studio 的每个 IAM 用户、组或角色。此功能仅在 IAM 模式下受支持，在 SSO 模式下不支持。以下示例说明如何创建此类策略。

> **Important**
>
> 如果您应用类似于以下示例之一的 IAM 策略，则用户无法访问 SageMaker 工作室或指定 SageMaker 通过 API 通过 SageMaker 控制台。要访问 Studio，用户必须使用预签名 URL 或调用 SageMaker 直接 API。

示例 1：仅允许在接口终端节点的子网内进行连接

以下策略只允许连接到创建接口终端节点的子网中的调用方。

```
{
    "Id": "sagemaker-studio-example-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable SageMaker Studio Access",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedDomainUrl",
                "sagemaker:DescribeUserProfile"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpc": "vpc-111bbaaa"
                }
            }
        }
    ]
}
```

示例 2：仅允许通过接口端点进行连接**aws:sourceVpce**

以下策略仅允许连接到通过由aws:sourceVpce条件键。例如，第一个接口终端节点可以允许通过 SageMaker Studio 控制面板。第二个接口终端节点可以允许通过 SageMaker API。

```
{
    "Id": "sagemaker-studio-example-2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable SageMaker Studio Access",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedDomainUrl",
                "sagemaker:DescribeUserProfile"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:sourceVpce": [
                        "vpce-111bbccc",
                        "vpce-111bbddd"
                    ]
                }
            }
        }
    ]
```

```
}
```

此策略包含DescribeUserProfileaction. 通常你打电话DescribeUserProfile以确保用户配置文件的状态为InService在尝试连接到域之前。例如：

```
aws sagemaker describe-user-profile \
    --domain-id domain-id \
    --user-profile-name profile-name
```

响应:

```
{
    "DomainId": "domain-id",
    "UserProfileArn": "arn:aws:sagemaker:us-west-2:acct-id:user-profile/domain-id/profile-
name",
    "UserProfileName": "profile-name",
    "HomeEfsFileSystemUid": "200001",
    "Status": "InService",
    "LastModifiedTime": 1605418785.555,
    "CreationTime": 1605418477.297
}
```

```
aws sagemaker create-presigned-domain-url
    --domain-id domain-id \
    --user-profile-name profile-name
```

响应:

```
{
    "AuthorizedUrl": "https://domain-id.studio.us-west-2.sagemaker.aws/auth?
token=AuthToken"
}
```

对于这两个呼叫，如果你使用的是Amazon在 2018 年 8 月 13 日之前发布的 SDK，您必须在调用中指定终端节点 URL。例如，以下示例显示对以下示例的调用：create-presigned-domain-url：

```
aws sagemaker create-presigned-domain-url
    --domain-id domain-id \
    --user-profile-name profile-name \
    --endpoint-url vpc-endpoint-id.api.sagemaker.Region.vpce.amazonaws.com
```

示例 3：允许来自 IP 地址的连接**aws:SourceIp**

以下策略仅允许使用来自指定 IP 地址范围的连接aws:SourceIp条件键。

```
{
    "Id": "sagemaker-studio-example-3",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable SageMaker Studio Access",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedDomainUrl",
                "sagemaker:DescribeUserProfile"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "IpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
                }
            }
        }
    ]
}
```

示例 4：允许通过接口终端节点从 IP 地址进行连接**aws:VpcSourceIp**

如果你正在访问 SageMaker Studio 通过接口终端节点，您可以使用`aws:VpcSourceIp`条件键，用于仅允许来自创建接口终端节点的子网内指定 IP 地址范围的连接，如以下策略所示：

```
{
    "Id": "sagemaker-studio-example-4",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable SageMaker Studio Access",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedDomainUrl",
                "sagemaker:DescribeUserProfile"
            ],
            "Resource": "*",
            "Condition": {
                "IpAddress": {
                    "aws:VpcSourceIp": [
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
                },
                "StringEquals": {
                    "aws:SourceVpc": "vpc-111bbaaa"
                }
            }
        }
    ]
}
```

通过 VPC 接口终端节点连接到笔记本实例

您可以通过 VPC 连接到笔记本实例接口终端节点在您的 Virtual Private Cloud (VPC) 中，而不是通过公共互联网进行连接。当您使用 VPC 接口终端节点时，VPC 与笔记本实例之间的通信完全在Amazon网络。

SageMaker 笔记本实例支持Amazon Virtual Private Cloud(Amazon VPC) 接口终端节点由提供支持AmazonPrivateLink. 每个 VPC 终端节点均由一个或多个表示。弹性网络接口您的 VPC 子网中具有私有 IP 地址。

> Note
>
> 在创建接口 VPC 终端节点以连接到笔记本实例之前，请创建一个接口 VPC 终端节点以连接到 SageMaker API。这样，在用户调用 `CreatePresignedNotebookInstanceUrl` 以获取 URL 来连接到笔记本实例时，该调用还会通过接口 VPC 终端节点。有关信息，请参阅连接到 SageMaker 通过 VPC 接口终端节点节点 (p. 2785)。

您可以使用Amazon控制台或Amazon Command Line Interface(Amazon CLI) 命令。有关说明，请参阅创建接口终端节点。确保为 VPC 中要连接到笔记本实例的所有子网创建接口终端节点。

在您创建接口终端节点时，请指定 aws.sagemaker.*region*.notebook 作为服务名。在创建 VPC 终端节点后，为您的 VPC 终端节点启用私有 DNS。任何使用 SageMaker API，Amazon CLI或者从 VPC 中连接到笔记本实例的控制台通过 VPC 终端节点而不是公有 Internet 连接到笔记本实例。

SageMaker 笔记本实例支持 VPC 终端节点Amazon两者的区域Amazon VPC和SageMaker都可用。

主题

- 将您的专用网络连接到 VPC (p. 2791)
- 为 SageMaker 笔记本实例创建 VPC 终端节点策略 (p. 2791)
- 将访问限制为来自您 VPC 中的连接 (p. 2791)

将您的专用网络连接到 VPC

要通过 VPC 连接到笔记本实例，您必须从位于 VPC 中的实例进行连接，或者使用Amazon Virtual Private Network(Amazon VPN) 或Amazon Direct Connect. 有关的信息Amazon VPN，请参阅VPN 连接中的Amazon Virtual Private Cloud 用户指南. 有关的信息Amazon Direct Connect，请参阅创建连接中的AmazonDirect Connect 用户指南.

为 SageMaker 笔记本实例创建 VPC 终端节点策略

您可以为 Amazon VPC 终端节点创建一个策略， SageMaker 笔记本实例指定以下内容：

- 可执行操作的委托人。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 终端节点控制对服务的访问。

以下 VPC 终端节点策略示例指定，所有有权访问该终端节点的用户都可以访问名为 `myNotebookInstance` 的笔记本实例。

```
{
  "Statement": [
      {
          "Action": "sagemaker:CreatePresignedNotebookInstanceUrl",
          "Effect": "Allow",
          "Resource": "arn:aws:sagemaker:us-west-2:123456789012:notebook-instance/
myNotebookInstance",
          "Principal": "*"
      }
  ]
}
```

访问其他笔记本实例时会被拒绝。

将访问限制为来自您 VPC 中的连接

即使您在 VPC 中设置了接口终端节点，在 VPC 之外的用户仍可以通过 Internet 访问笔记本实例。

> **Important**
>
> 如果您应用类似于以下内容之一的 IAM 策略，则用户无法访问指定的 SageMaker 通过控制台进行 API 或笔记本实例。

要将访问限制为仅限从您 VPC 中进行的连接，请创建 Amazon Identity and Access Management 策略，将访问限制为仅允许来自您 VPC 中调用。然后，将该策略添加到用于访问笔记本实例的每个 Amazon Identity and Access Management 用户、组或角色。

Note

此策略只允许连接到创建接口终端节点的子网中的调用方。

```
{
    "Id": "notebook-example-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable Notebook Access",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedNotebookInstanceUrl",
                "sagemaker:DescribeNotebookInstance"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpc": "vpc-111bbaaa"
                }
            }
        }
    ]
}
```

如果您想要将对笔记本实例的访问限制为仅允许使用接口终端节点进行的连接，请使用 `aws:SourceVpce` 条件键而不是 `aws:SourceVpc`：

```
{
    "Id": "notebook-example-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable Notebook Access",
            "Effect": "Allow",
            "Action": [
                "sagemaker:CreatePresignedNotebookInstanceUrl",
                "sagemaker:DescribeNotebookInstance"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:sourceVpce": [
                        "vpce-111bbccc",
                        "vpce-111bbddd"
                    ]
                }
            }
        }
    ]
}
```

这些策略示例均假定您已为 SageMaker API。有关更多信息，请参阅 连接到 SageMaker 通过 VPC 接口终端节点节点 (p. 2785)。在第二个示例中，`aws:SourceVpce` 的值是笔记本实例的接口终端节点的 ID。另一个是接口终端节点的 ID SageMaker API。

这里的政策示例包括 `DescribeNotebookInstance`，因为通常你会打电话`DescribeNotebookInstance`以确保`NotebookInstanceStatus`是`InService`在你尝试连接到它之前。例如：

```
aws sagemaker describe-notebook-instance \
```

```
                    --notebook-instance-name myNotebookInstance


{
   "NotebookInstanceArn":
   "arn:aws:sagemaker:us-west-2:1234567890ab:notebook-instance/mynotebookinstance",
   "NotebookInstanceName": "myNotebookInstance",
   "NotebookInstanceStatus": "InService",
   "Url": "mynotebookinstance.notebook.us-west-2.sagemaker.aws",
   "InstanceType": "ml.m4.xlarge",
   "RoleArn":
   "arn:aws:iam::1234567890ab:role/service-role/AmazonSageMaker-
ExecutionRole-12345678T123456",
   "LastModifiedTime": 1540334777.501,
   "CreationTime": 1523050674.078,
   "DirectInternetAccess": "Disabled"
}
aws sagemaker create-presigned-notebook-instance-url --notebook-instance-name
 myNotebookInstance


{
   "AuthorizedUrl": "https://mynotebookinstance.notebook.us-west-2.sagemaker.aws?
authToken=AuthToken
}
```

对于这些调用，如果您没有为 VPC 终端节点启用专用 DNS 主机名，或者您使用的Amazon在 2018 年 8 月 13 日之前发布的 SDK，您必须在调用中指定终端节点 URL。例如，对的调用`create-presigned-notebook-instance-url`是：

```
aws sagemaker create-presigned-notebook-instance-url
    --notebook-instance-name myNotebookInstance --endpoint-url
    VPC_Endpoint_ID.api.sagemaker.Region.vpce.amazonaws.com
```

将您的专用网络连接到 VPC

调用 SageMaker 要通过您的 VPC 进行 API 和运行时，您必须从位于 VPC 中的实例进行连接，或者使用 Amazon Virtual Private Network(Amazon VPN) 或Amazon Direct Connect. 有关的信息Amazon VPN，请参阅VPN 连接中的Amazon Virtual Private Cloud 用户指南. 有关的信息Amazon Direct Connect，请参阅创建连接中的AmazonDirect Connect 用户指南.

授予权 SageMaker 访问您的亚马逊 VPC 中的资源

默认情况下，SageMaker 将在 Amazon Virtual Private Cloud 中运行以下作业类型。

- 处理
- 培训
- 模型托管
- Batch 转换
- 亚马逊 SageMaker 澄清
- SageMaker 编译

但是，这些作业的容器可以访问Amazon资源（例如，您可以在其中存储训练数据和模型项目的 Amazon Simple Storage Service (Amazon S3) 存储桶 —— 通过 Internet 进行。

要控制对数据和作业容器的访问，建议您创建一个私有 VPC，并将其配置为不可通过 Internet 访问。有关创建和配置 VPC 的信息，请参阅Amazon VPC 入门中的Amazon VPC User Guide. 使用 VPC 有助于保护您的

作业容器和数据，因为您可以将 VPC 配置为不连接到 Internet。使用 VPC，您还可以通过 VPC 流日志来监控进出作业容器的所有网络流量。有关更多信息，请参阅 Amazon VPC 用户指南 中的 VPC 流日志。

创建作业时，您通过指定子网和安全组来指定私有 VPC 配置。当您指定子网和安全组时， SageMaker 创建弹性网络接口这些信息与其中一个子网中的安全组关联。您的作业接口可将您的作业容器连接到 VPC 中的资源。有关网络接口的信息，请参阅弹性网络接口中的Amazon VPC User Guide.

主题

- 授予权 SageMaker 处理作业访问 Amazon VPC 中的资源的访问权限 (p. 2794)
- 授予权 SageMaker 训练作业对 Amazon VPC 中的资源的访问权限 (p. 2796)
- 授予权 SageMaker 托管终端节点对 Amazon VPC 中的资源的访问权限 (p. 2799)
- 为 Batch 转换作业授予 Amazon VPC 中的资源的访问权限。 (p. 2801)
- 给予亚马逊 SageMaker 澄清作业访问 Amazon VPC 中的资源的访问权限 (p. 2804)
- 授予权 SageMaker 编译作业访问 Amazon VPC 中的资源的访问权限 (p. 2807)

授予权 SageMaker 处理作业访问 Amazon VPC 中的资源的访问权限

要控制对数据和处理作业的访问，建议您创建一个私有 Amazon VPC，并将其配置为不可通过公有 Internet 访问。有关创建和配置 VPC 的信息，请参阅Amazon VPC 入门中的Amazon VPC User Guide. 使用 VPC 有助于保护您的处理容器和数据，因为您可以对 VPC 进行配置，使其不连接到 Internet。使用 VPC，您还可以通过 VPC 流日志来监控进出处理容器的所有网络流量。有关更多信息，请参阅 Amazon VPC 用户指南 中的 VPC 流日志。

本文档介绍了如何添加 Amazon VPC 配置以处理作业。

为 Amazon VPC 访问配置处理 Job

要在私有 VPC 中指定子网和安全组，请使用`NetworkConfig.VpcConfig`的请求参数`CreateProcessingJob`API，或在中创建处理作业时需要提供此信息。 SageMaker 控制台。SageMaker 使用此信息创建网络接口并将其附加到您的处理容器。网络接口在未连接到 Internet 的 VPC 中为您的处理容器提供网络连接。它们还可将您的处理作业连接到私有 VPC 中的资源。

以下是您在调用 `CreateProcessingJob` 时将包含的 `VpcConfig` 参数的示例：

```
VpcConfig: {
      "Subnets": [
          "subnet-0123456789abcdef0",
          "subnet-0123456789abcdef1",
          "subnet-0123456789abcdef2"
          ],
      "SecurityGroupIds": [
          "sg-0123456789abcdef0"
          ]
        }
```

配置私有 VPC 以进行 SageMaker 处理

为您的配置私有 VPC 时 SageMaker 处理作业时，请遵守以下准则。有关设置 VPC 的信息，请参阅使用 VPC 和子网中的Amazon VPC User Guide.

主题

- 确保子网拥有足够的 IP 地址 (p. 2795)
- 创建 Amazon S3 VPC 终端节点 (p. 2795)
- 使用自定义终端节点策略限制 S3 访问 (p. 2795)

- 配置路由表 (p. 2796)
- 配置 VPC 安全组 (p. 2796)
- 连接到 VPC 之外的资源 (p. 2796)

确保子网拥有足够的 IP 地址

对于处理作业中的每个实例，您的 VPC 子网应至少拥有两个私有 IP 地址。有关更多信息，请参阅 。针对 IPv4 的 VPC 和子网大小调整中的Amazon VPC User Guide.

创建 Amazon S3 VPC 终端节点

如果您将 VPC 配置为使处理容器不具有 Internet 访问权限，则这些容器无法连接到包含数据的 Amazon S3 存储桶，除非您创建一个允许访问的 VPC 终端节点。通过创建 VPC 终端节点，您允许处理容器访问用于存储数据的存储桶。我们还建议您创建自定义策略，只允许来自私有 VPC 的请求访问您的 S3 存储桶。有关更多信息，请参阅用于 Simple Storage Service (Amazon S3) 的端点。

要创建 S3 VPC 终端节点，请执行以下操作：

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在导航窗格中，选择 Endpoints (终端节点)，然后选择 Create Endpoint (创建终端节点)。
3. 对于 Service Name (服务名称)，请选择 com.amazonaws.***region***.s3，其中 *region* 是您的 VPC 所在的区域。
4. 对于 VPC，请选择要用于该终端节点的 VPC。
5. 对于 Configure route tables，选择终端节点要使用的路由表。VPC 服务自动将一个路由添加到您选择的每个路由表中，它将任何 S3 流量传送到新的终端节点。
6. 对于 Policy (策略)，请选择 Full Access (完全访问) 以允许 VPC 中的任何用户或服务完全访问 S3 服务。选择 Custom (自定义) 以进一步限制访问。有关信息，请参阅使用自定义终端节点策略限制 S3 访问 (p. 2795)。

使用自定义终端节点策略限制 S3 访问

默认终端节点策略允许您的 VPC 中的任何用户或服务完全访问 S3。要进一步限制 S3 访问，请创建一个自定义终端节点策略。有关更多信息，请参阅 。为 Amazon S3 使用终端节点策略. 您也可以使用存储桶策略将 S3 存储桶访问限制为仅来自 Amazon VPC 的流量。想要了解有关信息，请参阅使用 Amazon S3 存储桶策略.

限制在处理容器上安装包

默认终端节点策略允许用户在处理容器中安装来自 Amazon Linux 和 Amazon Linux 2 存储库的包。如果您不希望用户安装来自该存储库的包，则创建一个自定义终端节点策略，明确拒绝访问 Amazon Linux 和 Amazon Linux 2 存储库。以下是拒绝访问这些存储库的策略示例：

```
{
    "Statement": [
      {
        "Sid": "AmazonLinuxAMIRepositoryAccess",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::packages.*.amazonaws.com/*",
            "arn:aws:s3:::repo.*.amazonaws.com/*"
        ]
      }
```

```
    ]
}

{
    "Statement": [
        { "Sid": "AmazonLinux2AMIRepositoryAccess",
          "Principal": "*",
          "Action": [
              "s3:GetObject"
              ],
          "Effect": "Deny",
          "Resource": [
              "arn:aws:s3:::amazonlinux.*.amazonaws.com/*"
              ]
         }
    ]
}
```

配置路由表

对终端节点路由表使用默认的 DNS 设置，以便使用标准 Amazon S3 URL (例如`http://s3-aws-region.amazonaws.com/MyBucket`) 解析。如果您不使用默认 DNS 设置，则确保通过配置终端节点路由表来解析用于指定处理作业中数据位置的 URL。有关 VPC 终端节点路由表的信息，请参阅网关终端节点路由中的Amazon VPC User Guide.

配置 VPC 安全组

在分布式处理中，您必须允许在同一处理作业中的不同容器之间进行通信。为此，请为您的安全组配置规则，以允许同一安全组的成员之间实现入站连接。有关更多信息，请参阅 。安全组规则.

连接到 VPC 之外的资源

如果您将 VPC 配置为不具有 Internet 访问权限，则使用该 VPC 的处理作业无权访问 VPC 之外的资源。如果您的处理作业需要访问您的 VPC 外部的资源，请使用以下选项之一提供访问权限：

- 如果您的处理作业需要访问支持接口 VPC 终端节点的 Amazon 服务，请创建连接到该服务的终端节点。有关支持接口终端节点的服务的列表，请参阅VPC 终端节点中的Amazon VPC User Guide. 有关创建接口 VPC 端点的信息，请参阅。接口 VPC 终端节点 (Amazon) PrivateLink中的Amazon VPC User Guide.
- 如果您的处理作业需要访问不支持接口 VPC 终端节点的 Amazon 服务，或需要访问 Amazon 之外的资源，请创建一个 NAT 网关，并将安全组配置为允许出站连接。有关为 VPC 设置 NAT 网关的信息，请参阅。方案 2: 带有公有和私有子网的 VPC (NAT)中的Amazon Virtual Private Cloud 用户指南.

授予权 SageMaker 训练作业对 Amazon VPC 中的资源的访问权限

> **Note**
>
> 对于训练作业，您只能为子网配置默认租赁 VPC，其中您的实例在共享硬件上运行。有关 VPC 租赁属性的更多信息，请参阅专用实例。

为 Amazon VPC 访问配置训练 Job

要在私有 VPC 中指定子网和安全组，请使用`VpcConfig`的请求参数`CreateTrainingJob`API，或者当在中创建训练任务时提供此信息。 SageMaker 控制台。 SageMaker 使用此信息创建网络接口并将其附加到您的训练容器。网络接口在未连接到 Internet 的 VPC 中为您的训练容器提供网络连接。它们还可将您的训练作业连接到私有 VPC 中的资源。

以下是您在调用 `CreateTrainingJob` 时将包含的 `VpcConfig` 参数的示例：

```
VpcConfig: {
      "Subnets": [
          "subnet-0123456789abcdef0",
          "subnet-0123456789abcdef1",
          "subnet-0123456789abcdef2"
          ],
      "SecurityGroupIds": [
          "sg-0123456789abcdef0"
          ]
        }
```

配置私有 VPC SageMaker 训练

为您的配置私有 VPC 时 SageMaker 培训工作，请使用以下准则。有关设置 VPC 的信息，请参阅使用 VPC 和子网中的Amazon VPC User Guide.

主题

- 确保子网拥有足够的 IP 地址 (p. 2797)
- 创建 Amazon S3 VPC 终端节点 (p. 2797)
- 使用自定义终端节点策略限制 S3 访问 (p. 2797)
- 配置路由表 (p. 2798)
- 配置 VPC 安全组 (p. 2798)
- 连接到 VPC 之外的资源 (p. 2798)

确保子网拥有足够的 IP 地址

对于训练作业中的每个实例，您的 VPC 子网应至少拥有两个私有 IP 地址。有关更多信息，请参阅 。针对 IPv4 的 VPC 和子网大小调整中的Amazon VPC User Guide.

创建 Amazon S3 VPC 终端节点

如果您将 VPC 配置为使训练容器不具有 Internet 访问权限，则这些容器无法连接到包含训练数据的 Amazon S3 存储桶，除非您创建一个允许访问的 VPC 终端节点。通过创建 VPC 终端节点，您允许训练容器访问用于存储数据和模型构件的存储桶。我们还建议您创建自定义策略，只允许来自私有 VPC 的请求访问您的 S3 存储桶。有关更多信息，请参阅用于 Simple Storage Service (Amazon S3) 的端点。

要创建 S3 VPC 终端节点，请执行以下操作：

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在导航窗格中，选择 Endpoints (终端节点)，然后选择 Create Endpoint (创建终端节点)。
3. 适用于服务名称，搜索com.amazonaws.*##*.s3其中，*##*是您的 VPC 所在的区域的名称。
4. 选择网关。
5. 对于 VPC，请选择要用于该终端节点的 VPC。
6. 对于 Configure route tables，选择终端节点要使用的路由表。VPC 服务自动将一个路由添加到您选择的每个路由表中，它将任何 S3 流量传送到新的终端节点。
7. 对于 Policy (策略)，请选择 Full Access (完全访问) 以允许 VPC 中的任何用户或服务完全访问 S3 服务。选择 Custom (自定义) 以进一步限制访问。有关信息，请参阅使用自定义终端节点策略限制 S3 访问 (p. 2797)。

使用自定义终端节点策略限制 S3 访问

默认终端节点策略允许您的 VPC 中的任何用户或服务完全访问 S3。要进一步限制 S3 访问，请创建一个自定义终端节点策略。有关更多信息，请参阅 。为 Amazon S3 使用终端节点策略. 您也可以使用存储桶策略将

S3 存储桶访问限制为仅来自 Amazon VPC 的流量。想要了解有关信息，请参阅使用 Amazon S3 存储桶策略.

限制在训练容器上安装包

默认终端节点策略允许用户在训练容器中安装来自 Amazon Linux 和 Amazon Linux 2 存储库的包。如果您不希望用户安装来自该存储库的包，则创建一个自定义终端节点策略，明确拒绝访问 Amazon Linux 和 Amazon Linux 2 存储库。以下是拒绝访问这些存储库的策略示例：

```
{
    "Statement": [
      {
        "Sid": "AmazonLinuxAMIRepositoryAccess",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::packages.*.amazonaws.com/*",
            "arn:aws:s3:::repo.*.amazonaws.com/*"
        ]
      }
    ]
}

{
    "Statement": [
        { "Sid": "AmazonLinux2AMIRepositoryAccess",
          "Principal": "*",
          "Action": [
              "s3:GetObject"
              ],
          "Effect": "Deny",
          "Resource": [
              "arn:aws:s3:::amazonlinux.*.amazonaws.com/*"
              ]
         }
    ]
}
```

配置路由表

对终端节点路由表使用默认的 DNS 设置，以便使用标准 Amazon S3 URL (例如`http://s3-aws-region.amazonaws.com/MyBucket`) 解析。如果您不使用默认 DNS 设置，则确保通过配置终端节点路由表来解析用于指定训练作业中数据位置的 URL。有关 VPC 终端节点路由表的信息，请参阅网关终端节点路由中的Amazon VPC User Guide.

配置 VPC 安全组

在分布式训练中，您必须允许同一训练作业中的不同容器之间进行通信。为此，请为您的安全组配置规则，以允许同一安全组的成员之间实现入站连接。有关信息，请参阅安全组规则。

连接到 VPC 之外的资源

如果您将 VPC 配置为不具有 Internet 访问权限，则使用该 VPC 的训练作业无权访问 VPC 之外的资源。如果您的训练作业需要访问您的 VPC 外部的资源，请使用以下选项之一提供访问权限：

- 如果您的训练作业需要访问支持接口 VPC 终端节点的 Amazon 服务，请创建连接到该服务的终端节点。有关支持接口终端节点的服务的列表，请参阅VPC 终端节点中的Amazon VPC User Guide. 有关创建接口 VPC 端点的信息，请参阅。接口 VPC 终端节点 (Amazon) PrivateLink中的Amazon VPC User Guide.

- 如果您的训练作业需要访问不支持接口 VPC 终端节点的 Amazon 服务，或需要访问 Amazon 之外的资源，请创建一个 NAT 网关，并将安全组配置为允许出站连接。有关为 VPC 设置 NAT 网关的信息，请参阅。方案 2: 带有公有和私有子网的 VPC (NAT)中的Amazon Virtual Private Cloud 用户指南.

授予权 SageMaker 托管终端节点对 Amazon VPC 中的资源的访问权限

为 Amazon VPC 访问配置模型

要在私有 VPC 中指定子网和安全组，请使用`VpcConfig`的请求参数`CreateModel`API，或在中创建模型时需要提供此信息。 SageMaker 控制台。 SageMaker 使用此信息创建网络接口并将其附加到您的模型容器。网络接口在未连接到 Internet 的 VPC 中为您的模型容器提供网络连接。它们还允许您的模型连接到私有 VPC 中的资源。

> **Note**
>
> 您必须在私有 VPC 的不同可用区中至少创建两个子网 (即使只有一个托管实例)。

以下是您在调用 `CreateModel` 时将包含的 `VpcConfig` 参数的示例：

```
VpcConfig: {
      "Subnets": [
          "subnet-0123456789abcdef0",
          "subnet-0123456789abcdef1",
          "subnet-0123456789abcdef2"
          ],
      "SecurityGroupIds": [
          "sg-0123456789abcdef0"
          ]
       }
```

配置私有 VPC SageMaker 托管

为您的配置私有 VPC 时 SageMaker 模型，请遵守以下准则。有关设置 VPC 的信息，请参阅使用 VPC 和子网中的Amazon VPC User Guide.

主题

- 确保子网拥有足够的 IP 地址 (p. 2799)
- 创建 Amazon S3 VPC 终端节点 (p. 2799)
- 使用自定义终端节点策略限制对 Amazon S3 的访问 (p. 2800)
- 将终端节点访问在 VPC 中运行的容器的权限添加到策略 (p. 2801)
- 配置路由表 (p. 2801)
- 连接到 VPC 之外的资源 (p. 2801)

确保子网拥有足够的 IP 地址

对于每个模型实例，您的 VPC 子网应至少具有两个私有 IP 地址。有关更多信息，请参阅 。针对 IPv4 的 VPC 和子网大小调整中的Amazon VPC User Guide.

创建 Amazon S3 VPC 终端节点

如果您将 VPC 配置为使模型容器不具有 Internet 访问权限，则这些容器无法连接到包含数据的 Amazon S3 存储桶，除非您创建一个允许访问的 VPC 终端节点。通过创建 VPC 终端节点，您允许模型容器访问用于存

储数据和模型构件的存储桶。我们还建议您创建自定义策略，只允许来自私有 VPC 的请求访问您的 S3 存储桶。有关更多信息，请参阅用于 Simple Storage Service (Amazon S3) 的端点。

要创建 Amazon S3 VPC VPC 终端节点，请执行以

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在导航窗格中，选择 Endpoints (终端节点)，然后选择 Create Endpoint (创建终端节点)。
3. 适用于服务名称，选择com.amazonaws.*##*.s3其中，*##*是的名称Amazon您的 VPC 所在的区域。
4. 对于 VPC，请选择要用于该终端节点的 VPC。
5. 适用于配置路由表中，选择要使用的终端节点的路由表。VPC 服务自动将一个路由添加到您选择的每个路由表中，以将 Amazon S3 流量指向新的终端节点。
6. 适用于策略，选择完全访问权限以便允许 VPC 中的任何用户或服务完全访问 Amazon S3 服务。要进一步限制访问，请选择 Custom (自定义)。有关更多信息，请参阅 使用自定义终端节点策略限制对 Amazon S3 的访问 (p. 2800)。

使用自定义终端节点策略限制对 Amazon S3 的访问

默认终端节点策略允许您的 VPC 中的任何用户或服务完全访问 Amazon Simple Storage Service (Amazon S3)。要进一步限制 Amazon S3 访问，请创建一个自定义终端节点策略。有关更多信息，请参阅 。为 Amazon S3 使用终端节点策略.

您也可以使用存储桶策略将 S3 存储桶访问限制为仅来自 Amazon VPC 的流量。想要了解有关信息，请参阅使用 Amazon S3 存储桶策略.

使用自定义终端节点策略限制在模型容器上安装包

默认终端节点策略允许用户在模型容器中安装来自 Amazon Linux 和 Amazon Linux 2 存储库的包。如果您不希望用户安装来自这些存储库的包，则创建一个自定义终端节点策略，明确拒绝访问 Amazon Linux 和 Amazon Linux 2 存储库。以下是拒绝访问这些存储库的策略示例：

```
{
    "Statement": [
      {
        "Sid": "AmazonLinuxAMIRepositoryAccess",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::packages.*.amazonaws.com/*",
            "arn:aws:s3:::repo.*.amazonaws.com/*"
        ]
      }
    ]
}

{
    "Statement": [
        { "Sid": "AmazonLinux2AMIRepositoryAccess",
          "Principal": "*",
          "Action": [
              "s3:GetObject"
              ],
          "Effect": "Deny",
          "Resource": [
              "arn:aws:s3:::amazonlinux.*.amazonaws.com/*"
              ]
         }
    ]
```

```
}
```

将终端节点访问在 VPC 中运行的容器的权限添加到策略

这些区域有：`SageMakerFullAccess`托管策略包含相应的权限，您需要具有这些权限才能将为 Amazon VPC 访问配置的模型与终端节点 这些权限允许 SageMaker 创建 elastic network interface 并将其附加到在 VPC 中运行的模型容器。如果您使用自己的 IAM 策略，您必须向该策略添加以下权限才能使用为 VPC 访问配置的模型。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:CreateNetworkInterface"
            ],
            "Resource": "*"
        }
    ]
}
```

有关 `SageMakerFullAccess` 托管策略的更多信息，请参阅 Amazon托管策略：AmazonSageMakerFullAccess (p. 2755)。

配置路由表

对终端节点路由表使用默认的 DNS 设置，以便使用标准 Amazon S3 URL (例如`http://s3-aws-region.amazonaws.com/MyBucket`) 解析。如果您不使用默认 DNS 设置，则确保通过配置终端节点路由表来解析用于指定模型中数据位置的 URL。有关 VPC 终端节点路由表的信息，请参阅网关终端节点路由中的Amazon VPC User Guide.

连接到 VPC 之外的资源

如果您将 VPC 配置为不具有 Internet 访问权限，则使用该 VPC 的模型无权访问 VPC 之外的资源。如果您的模型需要访问您的 VPC 外部的资源，请使用以下选项之一提供访问权限：

- 如果您的模型需要访问支持接口 VPC 终端节点的 Amazon 服务，请创建连接到该服务的终端节点。有关支持接口终端节点的服务的列表，请参阅VPC 终端节点中的Amazon VPC User Guide. 有关创建接口 VPC 端点的信息，请参阅。接口 VPC 终端节点 (Amazon) PrivateLink中的Amazon VPC User Guide.
- 如果您的模型需要访问不支持接口 VPC 终端节点的 Amazon 服务，或需要访问 Amazon 之外的资源，请创建一个 NAT 网关，并将安全组配置为允许出站连接。有关为 VPC 设置 NAT 网关的信息，请参阅。方案 2: 带有公有和私有子网的 VPC (NAT)中的Amazon Virtual Private Cloud 用户指南.

为 Batch 转换作业授予 Amazon VPC 中的资源的访问权限。

要控制对数据和批量转换作业的访问，建议您创建一个私有 Amazon VPC，并将其配置为不可通过公有 Internet 访问。在创建模型时，您可以指定子网和安全组以指定私有 VPC 配置。然后，您可以在创建批量转换作业时指定相同的模型。当您指定子网和安全组时， SageMaker 创建弹性网络接口这些信息与其中一个

子网中的安全组关联。您的模型接口可将您的模型容器连接到 VPC 中的资源。有关网络接口的信息，请参阅弹性网络接口中的Amazon VPC User Guide.

本文档介绍了如何为批量转换作业添加 Amazon VPC 配置。

为 Amazon VPC 访问配置 Batch 转换 Job

要在私有 VPC 中指定子网和安全组，请使用 `CreateModel` API 的 `VpcConfig` 请求参数，或者当在 SageMaker 控制台中创建模型时提供此信息。然后在`ModelName`的请求参数`CreateTransformJob`API，或者在模型名称在中创建转换作业时的字段 SageMaker 控制台。 SageMaker 使用此信息创建网络接口并将其附加到您的模型容器。网络接口在未连接到 Internet 的 VPC 中为您的模型容器提供网络连接。它们还允许您的转换作业连接到私有 VPC 中的资源。

以下是您在调用 `CreateModel` 时将包含的 `VpcConfig` 参数的示例：

```
VpcConfig: {
      "Subnets": [
          "subnet-0123456789abcdef0",
          "subnet-0123456789abcdef1",
          "subnet-0123456789abcdef2"
          ],
      "SecurityGroupIds": [
          "sg-0123456789abcdef0"
          ]
        }
```

如果您使用创建模型`CreateModel`API 操作，用于创建模型的 IAM 执行角色必须包含中描述的权限。CreateModel API：执行角色权限 (p. 2737)，包括私有 VPC 所需的以下权限。

在控制台中创建模型时，如果选择创建新角色中的模型设置部分中，AmazonSageMakerFullAccess用于创建角色的策略已包含这些权限。如果您选择输入自定义 IAM 角色 ARN要么使用现有角色，则指定的角色 ARN 必须具有附加具有以下权限的执行策略。

```
{
            "Effect": "Allow",
            "Action": [
            "ec2:CreateNetworkInterface",
            "ec2:CreateNetworkInterfacePermission",
            "ec2:DeleteNetworkInterface",
            "ec2:DeleteNetworkInterfacePermission",
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeVpcs",
            "ec2:DescribeDhcpOptions",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
```

配置私有 VPC SageMaker 批量转换

为您的配置私有 VPC 时 SageMaker 请按照以下准则执行批量转换作业。有关设置 VPC 的信息，请参阅使用 VPC 和子网中的Amazon VPC User Guide.

主题

- 确保子网拥有足够的 IP 地址 (p. 2803)
- 创建 Amazon S3 VPC 终端节点 (p. 2803)
- 使用自定义终端节点策略限制 S3 访问 (p. 2803)
- 配置路由表 (p. 2804)
- 配置 VPC 安全组 (p. 2804)
- 连接到 VPC 之外的资源 (p. 2804)

确保子网拥有足够的 IP 地址

对于转换作业中的每个实例，您的 VPC 子网应至少具有两个私有 IP 地址。有关更多信息，请参阅 。针对 IPv4 的 VPC 和子网大小调整中的Amazon VPC User Guide.

创建 Amazon S3 VPC 终端节点

如果您将 VPC 配置为使模型容器不具有 Internet 访问权限，则这些容器无法连接到包含数据的 Amazon S3 存储桶，除非您创建一个允许访问的 VPC 终端节点。通过创建 VPC 终端节点，您允许模型容器访问用于存储数据和模型构件的存储桶。我们还建议您创建自定义策略，只允许来自私有 VPC 的请求访问您的 S3 存储桶。有关更多信息，请参阅用于 Simple Storage Service (Amazon S3) 的端点。

要创建 S3 VPC 终端节点，请执行以下操作：

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在导航窗格中，选择 Endpoints (终端节点)，然后选择 Create Endpoint (创建终端节点)。
3. 对于 Service Name (服务名称)，请选择 com.amazonaws.***region***.s3，其中 *region* 是您的 VPC 所在的区域。
4. 对于 VPC，请选择要用于该终端节点的 VPC。
5. 对于 Configure route tables，选择终端节点要使用的路由表。VPC 服务自动将一个路由添加到您选择的每个路由表中，它将任何 S3 流量传送到新的终端节点。
6. 对于 Policy (策略)，请选择 Full Access (完全访问) 以允许 VPC 中的任何用户或服务完全访问 S3 服务。选择 Custom (自定义) 以进一步限制访问。有关信息，请参阅使用自定义终端节点策略限制 S3 访问 (p. 2803)。

使用自定义终端节点策略限制 S3 访问

默认终端节点策略允许您的 VPC 中的任何用户或服务完全访问 S3。要进一步限制 S3 访问，请创建一个自定义终端节点策略。有关更多信息，请参阅 。为 Amazon S3 使用终端节点策略. 您也可以使用存储桶策略将 S3 存储桶访问限制为仅来自 Amazon VPC 的流量。想要了解有关信息，请参阅使用 Amazon S3 存储桶策略.

限制在模型容器上安装包

默认终端节点策略允许用户在训练容器中安装来自 Amazon Linux 和 Amazon Linux 2 存储库的包。如果您不希望用户安装来自该存储库的包，则创建一个自定义终端节点策略，明确拒绝访问 Amazon Linux 和 Amazon Linux 2 存储库。以下是拒绝访问这些存储库的策略示例：

```
{
    "Statement": [
      {
        "Sid": "AmazonLinuxAMIRepositoryAccess",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::packages.*.amazonaws.com/*",
            "arn:aws:s3:::repo.*.amazonaws.com/*"
        ]
      }
    ]
}

{
    "Statement": [
        { "Sid": "AmazonLinux2AMIRepositoryAccess",
```

```
            "Principal": "*",
            "Action": [
                "s3:GetObject"
                ],
            "Effect": "Deny",
            "Resource": [
                "arn:aws:s3:::amazonlinux.*.amazonaws.com/*"
                ]
           }
    ]
}
```

配置路由表

对终端节点路由表使用默认的 DNS 设置，以便使用标准 Amazon S3 URL (例如`http://s3-aws-region.amazonaws.com/MyBucket`) 解析。如果您不使用默认 DNS 设置，则确保通过配置终端节点路由表来解析用于指定批量转换作业中数据位置的 URL。有关 VPC 终端节点路由表的信息，请参阅网关终端节点路由中的Amazon VPC User Guide.

配置 VPC 安全组

在分布式批量转换中，您必须允许同一批量转换作业中的不同容器之间进行通信。为此，请为您的安全组配置规则，以允许同一安全组的成员之间实现入站连接。有关信息，请参阅安全组规则。

连接到 VPC 之外的资源

如果您将 VPC 配置为不具有 Internet 访问权限，则使用该 VPC 的批量转换作业无权访问 VPC 之外的资源。如果您的批量转换作业需要访问您的 VPC 外部的资源，请使用以下选项之一提供访问权限：

- 如果您的批量转换作业需要访问支持接口 VPC 终端节点的 Amazon 服务，请创建连接到该服务的终端节点。有关支持接口终端节点的服务的列表，请参阅VPC 终端节点中的Amazon VPC User Guide. 有关创建接口 VPC 端点的信息，请参阅。接口 VPC 终端节点 (Amazon) PrivateLink中的Amazon VPC User Guide.
- 如果您的批量转换作业需要访问不支持接口 VPC 终端节点的 Amazon 服务，或需要访问 Amazon 之外的资源，请创建一个 NAT 网关，并将安全组配置为允许出站连接。有关为 VPC 设置 NAT 网关的信息，请参阅。方案 2: 带有公有和私有子网的 VPC (NAT)中的Amazon Virtual Private Cloud 用户指南.

给予亚马逊 SageMaker 澄清作业访问 Amazon VPC 中的资源的访问权限

要控制对数据的访问， SageMaker 要澄清作业，建议您创建一个私有 Amazon VPC，并将其配置为不可通过公有 Internet 访问您的作业。有关创建和配置 Amazon VPC 以处理作业的信息，请参阅授予权 SageMaker 处理作业访问 Amazon VPC 中的资源的访问权限.

本文档介绍了如何添加满足以下要求的其他 Amazon VPC 配置 SageMaker 澄清工作。

主题

- 配置 SageMaker 澄清 Job VPC 访问权限的作业 (p. 2804)
- 为配置私有 Amazon VPC SageMaker 澄清工作 (p. 2806)

配置 SageMaker 澄清 Job VPC 访问权限的作业

在为其配置私有 Amazon VPC 时，您需要指定子网和安全组 SageMaker 澄清工作并使工作能够从 SageMaker 计算训练后偏差指标和功能贡献时的模型，以帮助解释模型预测。

主题

- SageMaker 澄清 Job Amazon VPC 子网和安全组 (p. 2805)
- 配置模型亚马逊 VPC 以进行推理 (p. 2805)

SageMaker 澄清 Job Amazon VPC 子网和安全组

您的私有 Amazon VPC 中的子网和安全组可以分配给 SageMaker 根据你创造工作的方式，以各种方式澄清工作。

- SageMaker 控制台：在中创建作业时需要提供此信息SageMaker 控制板. 从处理菜单，选择处理任务，然后选择创建处理任务. 选择VPC中的选项网络面板并使用下拉列表提供子网和安全组。确保此面板中提供的网络隔离选项设置为禁用。
- SageMaker API：使用`NetworkConfig.VpcConfig`的请求参数`CreateProcessingJob`API，如以下示例所示：

```
"NetworkConfig": {
    "VpcConfig": {
        "Subnets": [
            "subnet-0123456789abcdef0",
            "subnet-0123456789abcdef1",
            "subnet-0123456789abcdef2"
        ],
        "SecurityGroupIds": [
            "sg-0123456789abcdef0"
        ]
    }
}
```

- Python 开发工具包：使用`NetworkConfig`的参数`SageMakerClarifyProcessor`API或`Processor`API，如以下示例所示：

```
from sagemaker.network import NetworkConfig
network_config = NetworkConfig(
    subnets=[
        "subnet-0123456789abcdef0",
        "subnet-0123456789abcdef1",
        "subnet-0123456789abcdef2",
    ],
    security_group_ids=[
        "sg-0123456789abcdef0",
    ],
)
```

SageMaker 使用信息创建网络接口并将其附加到 SageMaker 澄清工作。网络接口提供 SageMaker 使用 Amazon VPC 中未连接到公共 Internet 的网络连接来澄清工作。他们还启用 SageMaker 澄清连接到私有 Amazon VPC 中的资源的作业。

配置模型亚马逊 VPC 以进行推理

为了计算训练后的偏差指标和可解释性， SageMaker 澄清工作需要从中获取推理 SageMaker 由`model_name`的参数分析配置(对于) SageMaker 澄清处理作业。或者，如果你使用`SageMakerClarifyProcessor`中的 API SageMaker Python SDK，工作需要获得`model_name`由ModelConfig类。为此， SageMaker 澄清作业用模型创建了一个短暂的端点，称为阴影端点，然后将模型的 Amazon VPC 配置应用于影子终端节点。

> **Note**
>
> 两者的网络隔离选项 SageMaker 澄清作业并且模型必须被禁用（默认情况下该选项处于禁用状态），以便 SageMaker 澄清作业可以与影子端点进行通信。

要将您的私有 Amazon VPC 中的子网和安全组指定到 SageMaker 模型，请使用VpcConfig的请求参数CreateModel使用 API 或提供此信息在创建模型时提供此信息。 SageMaker 控制台中的控制面板。以下是您在调用 CreateModel 时将包含的 VpcConfig 参数的示例：

```
"VpcConfig": {
    "Subnets": [
        "subnet-0123456789abcdef0",
        "subnet-0123456789abcdef1",
        "subnet-0123456789abcdef2"
    ],
    "SecurityGroupIds": [
        "sg-0123456789abcdef0"
    ]
}
```

您可以指定要使用启动的影子终端节点的实例数量。initial_instance_count的参数分析配置(对于) SageMaker 澄清处理作业。或者，如果你使用SageMakerClarifyProcessor中的 API SageMaker Python SDK，工作需要获得instance_count由ModelConfig类。

> Note
>
> 即使在创建影子终端节点时只请求一个实例，模型中至少需要两个子网ModelConfig在不同的可用区域中。否则，创建影子端点将失败，并显示如下错误：
> ClientError：托管端点 SageMaker-CLARIFY 端点-XXX 时出错：失败。原因：无法找到至少 2 个具有请求实例类型 YYY 且与 SageMaker 子网。

如果您的模型需要 Amazon S3 中的模型文件，则模型 Amazon VPC 需要具有 Amazon S3 VPC 终端节点。有关创建和配置 Amazon VPC 的更多信息。 SageMaker 模型，请参阅授予权 SageMaker 托管终端节点对 Amazon VPC 中的资源的访问权限 (p. 2799).

为配置私有 Amazon VPC SageMaker 澄清工作

通常，您可以按照中的步骤进行操作配置私有 VPC SageMaker 处理为您的私有 Amazon VPC 配置 SageMaker 澄清工作。下面是一些亮点内容和特殊要求 SageMaker 澄清工作。

主题

- Connect 到 Amazon VPC 之外的资源 (p. 2806)
- 配置 Amazon VPC 安全组 (p. 2807)

Connect 到 Amazon VPC 之外的资源

如果您将 Amazon VPC 配置为其不具有公共互联网访问权限，则需要进行一些额外的设置才能授予 SageMaker 澄清作业对亚马逊 VPC 外部资源和服务的访问权限。例如，需要 Amazon S3 VPC 终端节点，因为 SageMaker 澄清任务需要从 S3 存储桶加载数据集以及将分析结果保存到 S3 存储桶。有关更多信息，请参阅 。创建 Amazon S3 VPC 终端节点用于创作指南。另外，如果 SageMaker 澄清工作需要从影子端点获得推断，然后它需要再调用几个Amazon服务。

- 创建 Amazon SageMaker API 服务 VPC 终端节点：这些区域有： SageMaker 澄清工作需要致电亚马逊 SageMaker API 服务来操纵影子终端节点或描述 SageMaker 亚马逊 VPC 验证的模型。您可以按照中提供的指南进行操作保护所有亚马逊 SageMaker 使用 API 调用AmazonPrivateLink创建亚马逊的博客 SageMaker API VPC 终端节点，允许 SageMaker 澄清工作以拨打服务电话。请注意，亚马逊的服务名称 SageMaker API 服务是com.amazonaws.*region*.sagemaker.api其中，*##*是您的亚马逊 VPC 所在地区的名称。
- 创建 Amazon SageMaker 运行时 VPC 端节点：这些区域有： SageMaker 澄清任务需要调用 Amazon SageMaker 运行时服务，该服务将调用路由到影子终端节点。设置步骤与亚马逊的设置步骤类似 SageMaker API 服务。请注意，亚马逊的服务名称 SageMaker Runtime 服务com.amazonaws.*region*.sagemaker.runtime其中，*##*是您的亚马逊 VPC 所在地区的名称。

配置 Amazon VPC 安全组

使用以下方式之一指定两个或多个处理实例时，SageMaker Clarity 作业支持分布式处理：

- SageMaker 控制台：这些区域有：实例计数已在资源配置的一部分Job 设置上的面板创建处理任务页.
- SageMaker API：这些区域有：`InstanceCount`在使用创建作业时指定`CreateProcessingJob`API。
- Python 开发工具包：这些区域有：`instance_count`在使用SageMakerClarify 处理器API 或处理器API。

在分布式处理中，您必须允许同一处理作业中的不同实例之间进行通信。为此，请为您的安全组配置规则，以允许同一安全组的成员之间实现入站连接。想要了解有关信息，请参阅安全组规则.

授予权 SageMaker 编译作业访问 Amazon VPC 中的资源的访问权限

> Note
>
> 对于编译作业，您只能使用默认租赁 VPC 配置子网，其中您的作业在共享硬件上运行。有关 VPC 租赁属性的更多信息，请参阅专用实例。

为 Amazon VPC 访问配置编译 Job

要在私有 VPC 中指定子网和安全组，请使用`VpcConfig`的请求参数`CreateCompilationJob`API，或在中创建编译作业时需要提供此信息。 SageMaker 控制台。 SageMaker Neo 使用此信息创建网络接口并将其附加到您的编译作业。网络接口在未连接到 Internet 的 VPC 中为网络连接提供编译作业。它们还可将您的编译作业连接到私有 VPC 中的资源。以下是您在调用 `CreateCompilationJob` 时将包含的 `VpcConfig` 参数的示例：

```
VpcConfig: {"Subnets": [
          "subnet-0123456789abcdef0",
          "subnet-0123456789abcdef1",
          "subnet-0123456789abcdef2"
          ],
      "SecurityGroupIds": [
          "sg-0123456789abcdef0"
          ]
        }
```

配置私有 VPC SageMaker 汇编

为您的配置私有 VPC 时 SageMaker 编译作业，请使用以下准则。有关设置 VPC 的信息，请参阅使用 VPC 和子网中的Amazon VPC User Guide.

主题

- 确保子网拥有足够的 IP 地址 (p. 2807)
- 创建 Amazon S3 VPC 终端节点 (p. 2808)
- 使用自定义终端节点策略限制 S3 访问 (p. 2808)
- 配置路由表 (p. 2809)
- 配置 VPC 安全组 (p. 2809)

确保子网拥有足够的 IP 地址

对于编译作业中的每个实例，您的 VPC 子网应至少拥有两个私有 IP 地址。有关更多信息，请参阅 。针对 IPv4 的 VPC 和子网大小调整中的Amazon VPC User Guide.

创建 Amazon S3 VPC 终端节点

如果您将 VPC 配置为阻止访问互联网， SageMaker 除非您创建一个允许访问的 VPC 终端节点，否则 Neo 无法连接到包含模型的 Amazon S3 存储桶。通过创建 VPC 终端节点，您允许 SageMaker Neo 编译作业，以访问用于存储数据和模型构件的存储桶。我们还建议您创建自定义策略，只允许来自私有 VPC 的请求访问您的 S3 存储桶。有关更多信息，请参阅用于 Simple Storage Service (Amazon S3) 的端点。

要创建 S3 VPC 终端节点，请执行以下操作：

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在导航窗格中，选择 Endpoints (终端节点)，然后选择 Create Endpoint (创建终端节点)。
3. 适用于服务名称，搜索com.amazonaws.*##*.s3其中，*##*是您的 VPC 所在的区域的名称。
4. 选择网关。
5. 对于 VPC，请选择要用于该终端节点的 VPC。
6. 对于 Configure route tables，选择终端节点要使用的路由表。VPC 服务自动将一个路由添加到您选择的每个路由表中，它将任何 S3 流量传送到新的终端节点。
7. 对于 Policy (策略)，请选择 Full Access (完全访问) 以允许 VPC 中的任何用户或服务完全访问 S3 服务。选择 Custom (自定义) 以进一步限制访问。有关信息，请参阅使用自定义终端节点策略限制 S3 访问 (p. 2797)。

使用自定义终端节点策略限制 S3 访问

默认终端节点策略允许您的 VPC 中的任何用户或服务完全访问 S3。要进一步限制 S3 访问，请创建一个自定义终端节点策略。有关更多信息，请参阅 。为 Amazon S3 使用终端节点策略. 您也可以使用存储桶策略将 S3 存储桶访问限制为仅来自 Amazon VPC 的流量。想要了解有关信息，请参阅使用 Amazon S3 存储桶策略. 以下是一个自定义策略的示例：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": {
                "AWS": "*"
            },
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::your-sample-bucket",
                "arn:aws:s3:::your-sample-bucket/*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:SourceVpce": [
                        "vpce-01234567890123456"
                    ]
                }
            }
        }
    ]
}
```

将在 Amazon VPC 中运行编译 Job 的权限添加到自定义 IAM 策略

这些区域有：`SageMakerFullAccess`托管策略包含相应的权限，您需要具有这些权限才能将为 Amazon VPC 访问配置的模型与终端节点 这些权限允许 SageMaker Neo 创建 elastic network interface 并将其附加到 Amazon VPC 中运行的编译作业。如果您使用自己的 IAM 策略，您必须向该策略添加以下权限以使用为 Amazon VPC 访问配置的模型。

```
{"Version": "2012-10-17",
    "Statement": [
        {"Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:CreateNetworkInterface",
                "ec2:ModifyNetworkInterfaceAttribute"
            ],
            "Resource": "*"
        }
    ]
}
```

有关 `SageMakerFullAccess` 托管策略的更多信息，请参阅 Amazon托管策略：AmazonSageMakerFullAccess (p. 2755)。

配置路由表

对终端节点路由表使用默认的 DNS 设置，以便使用标准 Amazon S3 URL (例如`http://s3-aws-region.amazonaws.com/MyBucket`) 解析。如果您不使用默认 DNS 设置，则确保通过配置终端节点路由表来解析用于指定编译作业中数据位置的 URL。有关 VPC 终端节点路由表的信息，请参阅网关终端节点路由中的Amazon VPC User Guide.

配置 VPC 安全组

在编译作业的安全组中，您必须允许与 Amazon S3 Amazon VPC 终端节点和用于编译作业的子网 CIDR 范围进行出站通信。想要了解有关信息，请参阅安全组规则和使用 Amazon VPC 终端节点控制对服务的访问.

监控 Amazon SageMaker

监控是保持 SageMaker 和其他人的可靠性、可用性和性能的重要方面。Amazon解决方案。Amazon提供以下监控工具来监控 SageMaker、在出现错误时进行报告并在适当的时候采取自动化措施：

- Amazon CloudWatch 实时监控您的 Amazon 资源以及在 Amazon 上运行的应用程序。您可以收集和跟踪指标，创建自定义的控制面板，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以具有 Amazon EC2 实例的 CloudWatch 跟踪 CPU 使用率或其他指标并且在需要时自动启动新实例。有关更多信息，请参阅 Amazon CloudWatch 用户指南。
- Amazon CloudWatch Logs使您能够监控、存储和访问来自 EC2 实例的日志文件，Amazon CloudTrail和其他来源的数据。CloudWatch Logs 可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南。
- Amazon CloudTrail 捕获由您的 Amazon 账户或代表该账户发出的 API 调用和相关事件，并将日志文件传输到您指定的 Simple Storage Service（Amazon S3）存储桶。您可以标识哪些用户和账户调用了 Amazon、从中发出调用的源 IP 地址以及调用的发生时间。有关更多信息，请参阅 Amazon CloudTrail 用户指南。
- CloudWatch Events提供近乎实时的系统事件流以描述Amazon资源的费用。创建 CloudWatch Event 规则可应对 SageMaker 训练、超参数优化或批量转换作业中的状态更改。

主题

- 使用 Amazon CloudWatch 监控 Amazon SageMaker (p. 2810)
- 使用 Amazon CloudWatch 记录 Amazon SageMaker 事件 (p. 2820)
- 使用记录 Amazon SageMaker API 调用Amazon CloudTrail (p. 2821)
- 使用 Amazon EventBridge 自动执行 Amazon SageMaker (p. 2823)

使用 Amazon CloudWatch 监控 Amazon SageMaker

您可以使用 Amazon CloudWatch 监控 Amazon SageMaker，此工具可收集原始数据，并将其处理为便于读取的近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。但是，Amazon CloudWatch 控制台将搜索限制最近 2 周内更新过的指标。此限制可确保显示您命名空间中最新的作业。要列出指标图形而不使用搜索，请在源视图中指定其确切名称。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

SageMaker 指标与维度

- SageMaker 终端节点调用指标 (p. 2811)
- SageMaker 多模型终端节点指标 (p. 2812)
- SageMaker 作业和端点指标 (p. 2813)
- SageMaker Ground Truth 指标 (p. 2816)
- SageMaker Feature Store 指标 (p. 2818)
- SageMaker 管道指标 (p. 2818)

SageMaker 终端节点调用指标

这些区域有：`AWS/SageMaker`命名空间包括调用到的以下请求指标InvokeEndpoint.

指标按 1 分钟一次的频率提供。

有关 CloudWatch 指标的保留时间长度的信息，请参阅。GetMetricStatistics中的Amazon CloudWatch API 参考.

终端节点调用指标

指标	描述
`Invocation4XXErrors`	模型在其中返回 4xx HTTP 响应代码的 `InvokeEndpoint` 请求的数量。对于每个 4xx 响应，发送 1；否则，发送 0。 单位：无 有效统计数据：Average、Sum
`Invocation5XXErrors`	模型在其中返回 5xx HTTP 响应代码的 `InvokeEndpoint` 请求的数量。对于每个 5xx 响应，发送 1；否则，发送 0。 单位：无 有效统计数据：Average、Sum
`Invocations`	的数量`InvokeEndpoint`发送到模型终端节点的请求。 要获取发送到模型终端节点的请求总数，请使用 Sum 统计数据。 单位：无 有效统计数据：总计
`InvocationsPerInstance`	发送到模型的调用数量，按`InstanceCount`在每个 ProductionVariant 中。1/`numberOfInstances`作为每个请求的值发送，其中`numberOfInstances`是发出请求时终端节点后面的 ProductionVariant 的活动实例数。 单位：无 有效统计数据：总计
`ModelLatency`	从 SageMaker 中查看的模型做出响应所用的时间隔。此时间间隔包括发送请求以及从模型容器提取响应的本地通信时间，以及在容器中完成推理所用的时间。 单位：微秒 有效统计数据：Average、Sum、Min、Max、Sample Count
`OverheadLatency`	SageMaker 开销为响应客户端的请求所用时间隔，此时间隔增加到 SageMaker 的客户端请求。此时间间隔的计算从 SageMaker 收到请求开始，直至它返回响应到客户端，减去`ModelLatency`. 开销延迟可能会由于多种因素发生而变化，包括请求和响应负载大小、请求频率以及请求的身份验证/授权。 单位：微秒 有效统计数据：Average、Sum、Min、Max、Sample Count

指标	描述
ModelSetupTime	为无服务器端点启动新的计算资源所需的时间。时间可能会因模型大小、下载模型所需的时间以及容器的启动时间而有所不同。 单位：微秒 有效统计数据：Average、Min、Max、Sample Count

终端节点调用指标的维度

维度	描述
EndpointName, VariantName	针对指定终端节点和变体的 ProductionVariant 筛选终端节点调用指标。

SageMaker 多模型终端节点指标

这些区域有：AWS/SageMaker命名空间包括从调用到的以下模型加载指标：InvokeEndpoint.

指标按 1 分钟一次的频率提供。

有关 CloudWatch 指标的保留时间长度的信息，请参阅。GetMetricStatistics中的Amazon CloudWatch API 参考.

Multi-Model Endpoint Model Loading Metrics (多模型终端节点模型加载指标)

指标	描述
ModelLoadingWaitTime	调用请求等待下载和/或加载目标模型以执行推理的间隔时间。 单位：微秒 有效统计数据：Average、Sum、Min、Max、Sample Count
ModelUnloadingTime	通过容器的 UnloadModel API 调用卸载模型所用的间隔时间。 单位：微秒 有效统计数据：Average、Sum、Min、Max、Sample Count
ModelDownloadingTime	从 Amazon Simple Storage Service (Amazon S3) 下载模型所用的间隔时间。 单位：微秒 有效统计数据：Average、Sum、Min、Max、Sample Count
ModelLoadingTime	通过容器的 LoadModel API 调用加载模型所用的间隔时间。 单位：微秒 有效统计数据：Average、Sum、Min、Max、Sample Count
ModelCacheHit	发送到已加载模型的多模型终端节点的 InvokeEndpoint 请求数。 “Average”统计数据显示已加载模型的请求的比率。

指标	描述
	单位：无 有效统计数据：Sum、Sample Count

Dimensions for Multi-Model Endpoint Model Loading Metrics (多模型终端节点模型加载指标的维度)

维度	描述
`EndpointName, VariantName`	针对指定终端节点和变体的 `ProductionVariant` 筛选终端节点调用指标。

这些区域有：`/aws/sagemaker/Endpoints`命名空间包括通过调用获得的以下实例指标：InvokeEndpoint.

指标按 1 分钟一次的频率提供。

有关 CloudWatch 指标的保留时间长度的信息，请参阅。GetMetricStatistics中的Amazon CloudWatch API 参考.

Multi-Model Endpoint Model Instance Metrics (多模型终端节点模型实例指标)

指标	描述
`LoadedModelCount`	多模型终端节点的容器中加载的模型数。此指标是按实例发射的。 周期为 1 分钟的“Average”统计数据指示每个实例加载的平均模型数。 “Sum”统计数据指示在终端节点中的所有实例上加载的模型总数。 此指标跟踪的模型不一定是唯一的，因为可能在终端节点的多个容器中加载模型。 单位：无 有效统计数据：Average、Sum、Min、Max、Sample Count

Dimensions for Multi-Model Endpoint Model Loading Metrics (多模型终端节点模型加载指标的维度)

维度	描述
`EndpointName, VariantName`	针对指定终端节点和变体的 `ProductionVariant` 筛选终端节点调用指标。

SageMaker 作业和端点指标

这些区域有：`/aws/sagemaker/ProcessingJobs`、`/aws/sagemaker/TrainingJobs`、`/aws/sagemaker/TransformJobs`, 和`/aws/sagemaker/Endpoints`命名空间包括以下用于训练作业和终端节点实例的指标。

指标按 1 分钟一次的频率提供。

Note

Amazon CloudWatch 支持高清自定义指标最好的分辨率是 1 秒钟。但是，分辨率越好，CloudWatch 指标的使用寿命就越短。对于 1 秒的频率分辨率，CloudWatch 指标可用期为 3 小时。有关 CloudWatch 指标的分辨率和寿命周期的更多信息，请参阅。GetMetricStatistics中的Amazon CloudWatch API 参考.

Tip

如果您想以更精细的分辨率分析您的训练工作，并随时将训练指标无限期存储在 Amazon S3 中以进行自定义分析，请考虑使用Amazon SageMaker 调试器. SageMaker Debug 提供了自动检测常见训练问题的内置规则；它可以检测硬件资源利用率问题（例如 CPU、GPU 和 I/O 瓶颈）和非融合模型问题（例如过度拟合、消失渐变和爆炸张量）。SageMaker 调试器还通过 Studio 及其性能分析报告提供可视化效果。要探索调试器的可视化效果，请参阅SageMaker 调试器见解仪表板演练、调试器性能分析报告演练, 和使用 smDebug 客户端库分析数据.

处理作业、训练作业、批量转换作业和终端节点实例指标

指标	描述
`CPUUtilization`	每个 CPU 核心的利用率总和。每个核心范围的 CPU 使用率为 0—100。例如，如果有四个 CPU，`CPUUtilization`范围为 0%-400%。对于处理作业，该值是实例上的处理容器的 CPU 利用率。 对于训练作业，该值是实例上的算法容器的 CPU 利用率。 对于批量转换作业，该值是实例上的转换容器的 CPU 利用率。 对于终端节点变体，该值是实例上的主容器和辅助容器的 CPU 利用率的总和。 Note 对于多个实例作业，每个实例都会报告 CPU 利用率指标。但是，CloudWatch 中的默认视图显示跨所有实例的平均 CPU 利用率。 单位：百分比
`MemoryUtilization`	实例上的容器所使用的内存的百分比。该值范围为 0%-100%。 对于处理作业，该值是实例上的处理容器的内存利用率。 对于训练作业，该值是实例上的算法容器的内存利用率。 对于批量转换作业，该值是实例上的转换容器的内存利用率。 对于终端节点变体，该值是实例上的主容器和辅助容器的内存利用率的总和。 单位：百分比 Note 对于多个实例作业，每个实例会报告内存利用率指标。但是，CloudWatch 中的默认视图显示跨所有实例的平均内存利用率。
`GPUUtilization`	实例上的容器所使用的 GPU 单位的百分比。该值可以介于范围之间，为 0 到 100，并且乘以 CPU 数目。例如，如果有四个 GPU，`GPUUtilization`范围为 0%-400%。 对于处理作业，该值是实例上的处理容器的 GPU 利用率。 对于训练作业，该值是实例上的算法容器的 GPU 利用率。 对于批量转换作业，该值是实例上的转换容器的 GPU 利用率。

指标	描述
	对于终端节点变体，该值是实例上的主容器和辅助容器的 GPU 利用率的总和。 Note 对于多个实例作业，每个实例都会报告 GPU 利用率指标。但是，CloudWatch 中的默认视图显示跨所有实例的平均 GPU 利用率。 单位：百分比
GPUMemoryUtilization	实例上的容器所使用的 GPU 内存的百分比。该值范围为 0 到 100，并且乘以 GPU 数目。例如，如果有四个 GPU，GPUMemoryUtilization范围为 0%-400%。 对于处理作业，该值是实例上的处理容器的 GPU 内存利用率。 对于训练作业，该值是实例上的算法容器的 GPU 内存利用率。 对于批量转换作业，该值是实例上的转换容器的 GPU 内存利用率。 对于终端节点变体，该值是实例上的主容器和辅助容器的 GPU 内存利用率的总和。 Note 对于多个实例作业，每个实例都会报告 GPU 内存利用率指标。但是，CloudWatch 中的默认视图显示跨所有实例的平均 GPU 内存利用率。 单位：百分比
DiskUtilization	实例上的容器所使用的磁盘空间的百分比。该值范围为 0%-100%。批量转换作业不支持此指标。 对于处理作业，该值是实例上的处理容器的磁盘空间利用率。 对于训练作业，该值是实例上的算法容器的磁盘空间利用率。 对于终端节点变体，该值是实例上的主容器和辅助容器的磁盘空间利用率的总和。 单位：百分比 Note 对于多个实例作业，每个实例会报告磁盘利用率指标。但是，CloudWatch 中的默认视图显示跨所有实例的平均磁盘利用率。

处理作业、训练作业和批量转换作业实例指标的维度

维度	描述
Host	对于处理作业，该维度的值具有格式 [processing-job-name]/algo-[instance-number-in-cluster]。使用此维度可筛选指定处理作业和实例的实例指标。此维度格式仅存在于 /aws/sagemaker/ProcessingJobs 命名空间中。 对于训练作业，该维度的值具有格式 [training-job-name]/algo-[instance-number-in-cluster]。使用此维度可筛选指定训练作业和实例的实例指标。此维度格式仅存在于 /aws/sagemaker/TrainingJobs 命名空间中。

维度	描述
	对于批量转换作业，该维度的值的格式为 `[transform-job-name]/[instance-id]`。使用此维度可筛选指定批量转换作业和实例的实例指标。此维度格式仅存在于 `/aws/sagemaker/TransformJobs` 命名空间中。

SageMaker Ground Truth 指标

Ground Truth 指标

指标	描述
`ActiveWorkers`	私人工作团队中的单个活跃工作人员已提交、发布或拒绝任务。要获取在职员工总数，请使用 Sum 统计数据。Ground Truth 试图交付每个人`ActiveWorkers`一次事件。如果此交付不成功，则此指标可能无法报告在职员工的总数 单位：无 有效统计数据：Sum、Sample Count
`DatasetObjectsAutoAnnotated`	在标记作业中自动注释的数据集对象数。此指标仅在启用自动标记时发出。要查看标记作业进度，请使用 Max 指标。 单位：无 有效统计数据：最大值
`DatasetObjectsHumanAnnotated`	在标记作业中人工注释的数据集对象数。要查看标记作业进度，请使用 Max 指标。 单位：无 有效统计数据：最大值
`DatasetObjectsLabelingFailed`	在标记作业中未能标记的数据集对象数。要查看标记作业进度，请使用 Max 指标。 单位：无 有效统计数据：最大值
`JobsFailed`	单个标签作业失败。要获取失败的标记作业总数，请使用 Sum 统计数据。 单位：无 有效统计数据：Sum、Sample Count
`JobsSucceeded`	单个标签作业成功。要获取成功的标记作业总数，请使用 Sum 统计数据。 单位：无 有效统计数据：Sum、Sample Count
`JobsStopped`	单个标签作业已停止。要获取停止的标记作业总数，请使用 Sum 统计数据。 单位：无 有效统计数据：Sum、Sample Count

指标	描述
TasksAccepted	一个工作人员接受了一项任务。要获取工作人员接受的任务总数，请使用 Sum 统计数据。Ground Truth 试图交付每个人TaskAccepted一次事件。如果此交付不成功，则此指标可能无法报告接受的任务总数。 单位：无 有效统计数据：Sum、Sample Count
TasksDeclined	工作人员拒绝了一项任务。要获取工作人员拒绝的任务总数，请使用 Sum 统计数据。Ground Truth 试图交付每个人TasksDeclined一次事件。如果此交付不成功，则此指标可能无法报告被拒绝的任务总数。 单位：无 有效统计数据：Sum、Sample Count
TasksReturned	返回了一个任务。要获取返回的任务总数，请使用 Sum 统计数据。Ground Truth 试图交付每个人TasksReturned一次事件。如果此交付不成功，则此指标可能无法报告返回的任务总数。 单位：无 有效统计数据：Sum、Sample Count
TasksSubmitted	私人员工提交/完成了单个任务。要获取工作人员提交的任务总数，请使用 Sum 统计数据。Ground Truth 试图交付每个人TasksSubmitted一次事件。如果此交付不成功，则此指标可能不会报告提交的任务总数。 单位：无 有效统计数据：Sum、Sample Count
TimeSpent	私人员工完成任务所用的时间。此指标不包括工作人员暂停或休息的时间。Ground Truth 试图交付每个TimeSpent一次事件。如果此交付不成功，则此指标可能无法报告花费的总时间。 单位：秒 有效统计数据：Sum、Sample Count
TotalDatasetObjectsLabeled	在标记作业中成功标记的数据集对象数。要查看标记作业进度，请使用 Max 指标。 单位：无 有效统计数据：最大值

数据集对象指标的维度

维度	描述
LabelingJobName	筛选标签作业的数据集对象计数指标。

SageMaker Feature Store 指标

功能商店指标

指标	描述
ConsumedReadRequestsUnits	在指定时间段内使用的读取单位数。您可以检索要 feature store 运行时操作所使用的读取单位及其对应的功能组。 单位：无 有效统计数据：全部
ConsumedWriteRequestsUnits	在指定时间段内使用的写入单元数。您可以检索要 feature store 运行时操作所使用的写入单元及其对应的功能组。 单位：无 有效统计数据：全部

功能商店指标的维度

维度	描述
FeatureGroupName, OperationName	筛选指定功能组的要素存储运行时操作指标。

SageMaker 管道指标

这些区域有：AWS/Sagemaker/ModelBuildingPipeline命名空间包括管道执行的以下指标。

提供了两类 Pipeline 执行指标：

- 所有管道的执行指标— 账户级管道执行指标（适用于当前账户中的所有管道）
- 管道执行指标— 每个管道的管道执行指标

指标按 1 分钟一次的频率提供。

管道执行指标

指标	描述
ExecutionStarted	启动的管道执行数。 单位：计数 有效统计数据：Average、Sum
ExecutionFailed	失败的管道执行数。 单位：计数 有效统计数据：Average、Sum

指标	描述
ExecutionSucceeded	成功的管道执行数。 单位：计数 有效统计数据：Average、Sum
ExecutionStopped	停止的管道执行的数量。 单位：计数 有效统计数据：Average、Sum
ExecutionDuration	管道执行运行的持续时间（以毫秒为单位）。 单位：毫秒 有效统计数据：Average、Sum、Min、Max、Sample Count

按管道分类的执行指标的维度

维度	描述
PipelineName	筛选指定管道的管道执行指标。

管道步骤指标

这些区域有：AWS/Sagemaker/ModelBuildingPipeline命名空间包括管道步骤的以下指标。

指标按 1 分钟一次的频率提供。

指标	描述
StepStarted	启动的步骤数。 单位：计数 有效统计数据：Average、Sum
StepFailed	失败的步骤数。 单位：计数 有效统计数据：Average、Sum
StepSucceeded	成功的步骤数。 单位：计数 有效统计数据：Average、Sum
StepStopped	停止的步骤数。 单位：计数 有效统计数据：Average、Sum

指标	描述
StepDuration	步骤运行的持续时长（以毫秒为单位）。 单位：毫秒 有效统计数据：Average、Sum、Min、Max、Sample Count

管道步骤指标的维度

维度	描述
PipelineName, StepName	筛选指定管道和步骤的步骤指标。

使用 Amazon CloudWatch 记录 Amazon SageMaker 事件

为帮助您调试编译作业、处理作业、训练作业、终端节点、转换作业、笔记本实例和笔记本实例生命周期配置，算法容器、模型容器或笔记本实例生命周期配置发送到的任何内容也会发送到。stdout要么stderr还发送到 Amazon CloudWatch Logs 中。除了调试之外，您还可以使用它们进行进度分析。

日志

下表列出了 Amazon SageMaker 提供的所有日志。

日志

日志组名称	日志流名称
/aws/sagemaker/CompilationJobs	[compilation-job-name]
/aws/sagemaker/Endpoints/[EndpointName]	[production-variant-name]/[instance-id]
	[production-variant-name]/[instance-id]/[container-name provided in SageMaker model] (For Inference Pipelines)
/aws/sagemaker/groundtruth/WorkerActivity	aws/sagemaker/groundtruth/worker-activity/[requester-AWS-Id]-[region]/[timestamp]
/aws/sagemaker/LabelingJobs	[labeling-job-name]
/aws/sagemaker/NotebookInstances	[notebook-instance-name]/[LifecycleConfigHook]
	[notebook-instance-name]/jupyter.log
/aws/sagemaker/ProcessingJobs	[processing-job-name]/[hostname]-[epoch_timestamp]
/aws/sagemaker/studio	[domain-id]/[user-profile-name]/[app-type]/[app-name]

日志组名称	日志流名称
	[domain-id]/domain-shared/rstudioserverpro/default
/aws/sagemaker/TrainingJobs	[training-job-name]/algo-[instance-number-in-cluster]-[epoch_timestamp]
/aws/sagemaker/TransformJobs	[transform-job-name]/[instance-id]-[epoch_timestamp]
	[transform-job-name]/[instance-id]-[epoch_timestamp]/data-log
	[transform-job-name]/[instance-id]-[epoch_timestamp]/[container-name provided in SageMaker model] (For Inference Pipelines)

Note

1. 在您使用生命周期配置创建笔记本实例时，将创建 /aws/sagemaker/NotebookInstances/[LifecycleConfigHook] 日志流。有关更多信息，请参阅 使用生命周期配置脚本自定义笔记本实例 (p. 152)。
2. 对于推理管道，如果您未提供容器名称，则平台使用 “容器/1”、“容器/2”，以此类推，与 SageMaker 模型中提供的顺序相对应。

有关使用 CloudWatch 日志记录的更多信息，请参阅。什么是 Amazon CloudWatch Logs？中的Amazon CloudWatch 用户指南.

使用记录 Amazon SageMaker API 调用Amazon CloudTrail

Amazon SageMaker 与Amazon CloudTrail，此服务可提供用户、角色或者执行操作的记录的服务AmazonSageMaker 中的服务。CloudTrail 捕获 SageMaker 的所有 API 调用，但InvokeEndpoint和InvokeEndpoint 异步，作为活动。捕获的调用包括来自 SageMaker 控制台和对 SageMaker API 操作的代码调用。如果创建跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 SageMaker 的事件）。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 SageMaker 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《Amazon CloudTrail 用户指南》。

默认情况下，日志数据将无限期存储在 CloudWatch Logs 中。但是，您可以配置要在日志组中存储日志数据多长时间。想要了解有关信息，请参阅更改 CloudWatch Logs 中的日志数据保留中的Amazon CloudWatch Logs 用户指南.

CloudTrail 中的 SageMaker 信息

在您创建 Amazon 账户时，将在该账户上启用 CloudTrail。当 Amazon SageMaker 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他事件一同保存在Amazon中的服务事件事件历史. 您可以在 Amazon 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录您的事件Amazon要创建跟踪（包括 Amazon SageMaker 的事件），请创建跟踪。通过跟踪记录，CloudTrail 可将日志文件传送至 Simple Storage Service（Amazon S3）存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 Amazon 区域。此跟踪记录在 Amazon 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）存储桶。此外，您可以配置其他

Amazon 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 SageMaker 操作，但InvokeEndpoint和InvokeEndpoint 异步，由 CloudTrail 记录下来并记载到Operations. 例如，对 CreateTrainingJob、CreateEndpoint 和 CreateNotebookInstance 操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 Amazon Identity and Access Management (IAM) 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它Amazon服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

由自动模型优化执行的操作

SageMaker 支持将非 API 服务事件记录到您的 CloudTrail 日志文件，以便自动模型优化作业。这些事件与您的优化作业相关，但并非客户请求公共 Amazon API 的直接结果。例如，当您通过调用创建超参数优化作业时CreateHyperParameterTuningJob，SageMaker 创建训练作业以评估超参数的不同组合来找到最佳结果。同样，当你打电话StopHyperParameterTuningJob要停止超级参数优化作业，SageMaker 可能会停止任何关联的正在运行的训练作业。您的优化作业的非 API 事件将记录到 CloudTrail，以帮助您改进对您的管理、合规性以及操作和风险审核。Amazonaccount.

从非 API 服务事件生成的日志条目具有 AwsServiceEvent 而非 AwsApiCall 的 eventType。

了解 SageMaker 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

以下示例为 CreateEndpoint 操作的日志条目，它创建一个终端节点以部署经过训练的模型。

```
{
    "eventVersion":"1.05",
    "userIdentity": {
        "type":"IAMUser",
        "principalId":"AIXDAYQEXAMPLEUMLYNGL",
        "arn":"arn:aws:iam::123456789012:user/intern",
        "accountId":"123456789012",
        "accessKeyId":"ASXIAGXEXAMPLEQULKNXV",
        "userName":"intern"
    },
    "eventTime":"2018-01-02T13:39:06Z",
    "eventSource":"sagemaker.amazonaws.com",
    "eventName":"CreateEndpoint",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"127.0.0.1",
    "userAgent":"USER_AGENT",
    "requestParameters": {
        "endpointName":"ExampleEndpoint",
```

```
        "endpointConfigName":"ExampleEndpointConfig"
    },
    "responseElements": {
        "endpointArn":"arn:aws:sagemaker:us-west-2:123456789012:endpoint/exampleendpoint"
    },
    "requestID":"6b1b42b9-EXAMPLE",
    "eventID":"a6f85b21-EXAMPLE",
    "eventType":"AwsApiCall",
    "recipientAccountId":"444455556666"
}
```

以下示例是 `CreateModel` 操作的日志条目，它创建一个或多个容器来托管以前经过训练的模型。

```
{
    "eventVersion":"1.05",
    "userIdentity": {
        "type":"IAMUser",
        "principalId":"AIXDAYQEXAMPLEUMLYNGL",
        "arn":"arn:aws:iam::123456789012:user/intern",
        "accountId":"123456789012",
        "accessKeyId":"ASXIAGXEXAMPLEQULKNXV",
        "userName":"intern"
    },
    "eventTime":"2018-01-02T15:23:46Z",
    "eventSource":"sagemaker.amazonaws.com",
    "eventName":"CreateModel",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"127.0.0.1",
    "userAgent":"USER_AGENT",
    "requestParameters": {
        "modelName":"ExampleModel",
        "primaryContainer": {
            "image":"174872318107.dkr.ecr.us-west-2.amazonaws.com/kmeans:latest"
        },
        "executionRoleArn":"arn:aws:iam::123456789012:role/EXAMPLEARN"
    },
    "responseElements": {
        "modelArn":"arn:aws:sagemaker:us-west-2:123456789012:model/
barkinghappy2018-01-02t15-23-32-275z-ivrdog"
    },
    "requestID":"417b8dab-EXAMPLE",
    "eventID":"0f2b3e81-EXAMPLE",
    "eventType":"AwsApiCall",
    "recipientAccountId":"444455556666"
}
```

使用 Amazon EventBridge 自动执行 Amazon SageMaker

Amazon EventBridge 监控 Amazon SageMaker 中的状态更改事件。您可以使用 SageMaker 自动执行并自动响应事件，例如训练作业状态更改或终端节点状态更改。SageMaker 的事件近乎实时传输到 EventBridge。您可以编写简单规则来指示您关注的事件，并指示要在事件匹配规则时执行的自动化操作。有关如何创建规则的示例，请参阅使用 Amazon EventBridge 安排管道 (p. 2473).

可自动触发的操作示例包括：

- 调用 Amazon Lambda 函数
- 调用 Amazon EC2 Run Command

- 将事件中继到 Amazon Kinesis Data Streams
- 激活 Amazon Step Functions 状态机
- 通知 Amazon SNS 主题或Amazon SMS排队

SageMaker 事件由 EventBridge 监控

- 训练作业状态更改 (p. 2824)
- 更改超级参数优化作业状态 (p. 2825)
- 转换作业状态更改 (p. 2826)
- 端点状态更改 (p. 2827)
- 功能组状态更改 (p. 2828)
- 模型包状态更改 (p. 2829)
- 管道执行状态更改 (p. 2830)
- 管道步骤状态更改 (p. 2830)
- SageMaker 映像状态更改 (p. 2831)
- SageMaker 映像版本状态更改 (p. 2831)
- 端节点部署状态更改 (p. 2832)

训练作业状态更改

指示 SageMaker 训练作业的状态更改。

如果 `TrainingJobStatus` 的值为 `Failed`，则事件包含 `FailureReason` 字段，此字段描述训练作业为何失败。

```
{
    "version": "0",
    "id": "844e2571-85d4-695f-b930-0153b71dcb42",
    "detail-type": "SageMaker Training Job State Change",
    "source": "aws.sagemaker",
    "account": "123456789012",
    "time": "2018-10-06T12:26:13Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:sagemaker:us-east-1:123456789012:training-job/kmeans-1"
    ],
    "detail": {
        "TrainingJobName": "89c96cc8-dded-4739-afcc-6f1dc936701d",
        "TrainingJobArn": "arn:aws:sagemaker:us-east-1:123456789012:training-job/kmeans-1",
        "TrainingJobStatus": "Completed",
        "SecondaryStatus": "Completed",
        "HyperParameters": {
            "Hyper": "Parameters"
        },
        "AlgorithmSpecification": {
            "TrainingImage": "TrainingImage",
            "TrainingInputMode": "TrainingInputMode"
        },
        "RoleArn": "arn:aws:iam::123456789012:role/SMRole",
        "InputDataConfig": [
            {
                "ChannelName": "Train",
                "DataSource": {
                    "S3DataSource": {
                        "S3DataType": "S3DataType",
                        "S3Uri": "S3Uri",
```

```
                        "S3DataDistributionType": "S3DataDistributionType"
                    }
                },
                "ContentType": "ContentType",
                "CompressionType": "CompressionType",
                "RecordWrapperType": "RecordWrapperType"
            }
        ],
        "OutputDataConfig": {
            "KmsKeyId": "KmsKeyId",
            "S3OutputPath": "S3OutputPath"
        },
        "ResourceConfig": {
            "InstanceType": "InstanceType",
            "InstanceCount": 3,
            "VolumeSizeInGB": 20,
            "VolumeKmsKeyId": "VolumeKmsKeyId"
        },
        "VpcConfig": {

        },
        "StoppingCondition": {
            "MaxRuntimeInSeconds": 60
        },
        "CreationTime": "1583831889050",
        "TrainingStartTime": "1583831889050",
        "TrainingEndTime": "1583831889050",
        "LastModifiedTime": "1583831889050",
        "SecondaryStatusTransitions": [

        ],
        "Tags": {

        }
    }
}
```

更改超级参数优化作业状态

指示 SageMaker 超参数优化作业的状态更改。

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "SageMaker HyperParameter Tuning Job State Change",
  "source": "aws.sagemaker",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:sagemaker:us-east-1:123456789012:tuningJob/x"
  ],
  "detail": {
    "HyperParameterTuningJobName": "016bffd3-6d71-4d3a-9710-0a332b2759fc",
    "HyperParameterTuningJobArn": "arn:aws:sagemaker:us-east-1:123456789012:tuningJob/x",
    "TrainingJobDefinition": {
      "StaticHyperParameters": {},
      "AlgorithmSpecification": {
        "TrainingImage": "trainingImageName",
        "TrainingInputMode": "inputModeFile",
        "MetricDefinitions": [
          {
            "Name": "metricName",
```

```
            "Regex": "regex"
          }
        ]
      },
      "RoleArn": "roleArn",
      "InputDataConfig": [
        {
          "ChannelName": "channelName",
          "DataSource": {
            "S3DataSource": {
              "S3DataType": "s3DataType",
              "S3Uri": "s3Uri",
              "S3DataDistributionType": "s3DistributionType"
            }
          },
          "ContentType": "contentType",
          "CompressionType": "gz",
          "RecordWrapperType": "RecordWrapper"
        }
      ],
      "VpcConfig": {
        "SecurityGroupIds": [
          "securityGroupIds"
        ],
        "Subnets": [
          "subnets"
        ]
      },
      "OutputDataConfig": {
        "KmsKeyId": "kmsKeyId",
        "S3OutputPath": "s3OutputPath"
      },
      "ResourceConfig": {
        "InstanceType": "instanceType",
        "InstanceCount": 10,
        "VolumeSizeInGB": 500,
        "VolumeKmsKeyId": "volumeKeyId"
      },
      "StoppingCondition": {
        "MaxRuntimeInSeconds": 3600
      }
    },
    "HyperParameterTuningJobStatus": "status",
    "CreationTime": "1583831889050",
    "LastModifiedTime": "1583831889050",
    "TrainingJobStatusCounters": {
      "Completed": 1,
      "InProgress": 0,
      "RetryableError": 0,
      "NonRetryableError": 0,
      "Stopped": 0
    },
    "ObjectiveStatusCounters": {
      "Succeeded": 1,
      "Pending": 0,
      "Failed": 0
    },
    "Tags": {}
  }
}
```


转换作业状态更改

指示 SageMaker 批次转换作业的状态更改。

如果 TransformJobStatus 的值为 Failed，则事件包含 FailureReason 字段，此字段描述训练作业为何失败。

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "SageMaker Transform Job State Change",
  "source": "aws.sagemaker",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": ["arn:aws:sagemaker:us-east-1:123456789012:transform-job/myjob"],
  "detail": {
    "TransformJobName": "4b52bd8f-e034-4345-818d-884bdd7c9724",
    "TransformJobArn": "arn:aws:sagemaker:us-east-1:123456789012:transform-job/myjob",
    "TransformJobStatus": "another status... GO",
    "FailureReason": "failed why 1",
    "ModelName": "i am a beautiful model",
    "MaxConcurrentTransforms": 5,
    "MaxPayloadInMB": 10,
    "BatchStrategy": "Strategizing...",
    "Environment": {
      "environment1": "environment2"
    },
    "TransformInput": {
      "DataSource": {
        "S3DataSource": {
          "S3DataType": "s3DataType",
          "S3Uri": "s3Uri"
        }
      },
      "ContentType": "content type",
      "CompressionType": "compression type",
      "SplitType": "split type"
    },
    "TransformOutput": {
      "S3OutputPath": "s3Uri",
      "Accept": "accept",
      "AssembleWith": "assemblyType",
      "KmsKeyId": "kmsKeyId"
    },
    "TransformResources": {
      "InstanceType": "instanceType",
      "InstanceCount": 3
    },
    "CreationTime": "2018-10-06T12:26:13Z",
    "TransformStartTime": "2018-10-06T12:26:13Z",
    "TransformEndTime": "2018-10-06T12:26:13Z",
    "Tags": {}
  }
}
```

端点状态更改

指示 SageMaker 托管的实时推理终端节点的状态更改。

下面显示了中包含终端节点的事件IN_SERVICE状态。

```
{
  "version": "0",
  "id": "d2921b5a-b0ad-cace-a8e3-0f159d018e06",
  "detail-type": "SageMaker Endpoint State Change",
  "source": "aws.sagemaker",
```

```
  "account": "123456789012",
  "time": "1583831889050",
  "region": "us-west-2",
  "resources": [
      "arn:aws:sagemaker:us-west-2:123456789012:endpoint/myendpoint"
  ],
  "detail": {
      "EndpointName": "MyEndpoint",
      "EndpointArn": "arn:aws:sagemaker:us-west-2:123456789012:endpoint/myendpoint",
      "EndpointConfigName": "MyEndpointConfig",
      "ProductionVariants": [
          {
              "DesiredWeight": 1.0,
              "DesiredInstanceCount": 1.0
          }
      ],
      "EndpointStatus": "IN_SERVICE",
      "CreationTime": 1592411992203.0,
      "LastModifiedTime": 1592411994287.0,
      "Tags": {

      }
  }
}
```

功能组状态更改

表示在`FeatureGroupStatus`或者`OfflineStoreStatus`SageMaker 功能组的数据。

```
{
  "version": "0",
  "id": "93201303-abdb-36a4-1b9b-4c1c3e3671c0",
  "detail-type": "SageMaker Feature Group State Change",
  "source": "aws.sagemaker",
  "account": "123456789012",
  "time": "2021-01-26T01:22:01Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:sagemaker:us-east-1:123456789012:feature-group/sample-feature-group"
  ],
  "detail": {
    "FeatureGroupArn": "arn:aws:sagemaker:us-east-1:123456789012:feature-group/sample-
feature-group",
    "FeatureGroupName": "sample-feature-group",
    "RecordIdentifierFeatureName": "RecordIdentifier",
    "EventTimeFeatureName": "EventTime",
    "FeatureDefinitions": [
      {
        "FeatureName": "RecordIdentifier",
        "FeatureType": "Integral"
      },
      {
        "FeatureName": "EventTime",
        "FeatureType": "Fractional"
      }
    ],
    "CreationTime": 1611624059000,
    "OnlineStoreConfig": {
      "EnableOnlineStore": true
    },
    "OfflineStoreConfig": {
      "S3StorageConfig": {
        "S3Uri": "s3://offline/s3/uri"
```

```
      },
      "DisableGlueTableCreation": false,
      "DataCatalogConfig": {
        "TableName": "sample-feature-group-1611624059",
        "Catalog": "AwsDataCatalog",
        "Database": "sagemaker_featurestore"
      }
    },
    "RoleArn": "arn:aws:iam::123456789012:role/SageMakerRole",
    "FeatureGroupStatus": "Active",
    "Tags": {}
  }
}
```

模型包状态更改

指示 SageMaker 模型包的状态更改。

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "SageMaker Model Package State Change",
  "source": "aws.sagemaker",
  "account": "123456789012",
  "time": "2021-02-24T17:00:14Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:sagemaker:us-east-2:123456789012:model-package/versionedmp-p-idy6c3e1fiqj/2"
  ],
  "source": [
    "aws.sagemaker"
  ],
  "detail": {
    "ModelPackageGroupName": "versionedmp-p-idy6c3e1fiqj",
    "ModelPackageVersion": 2,
    "ModelPackageArn": "arn:aws:sagemaker:us-east-2:123456789012:model-package/versionedmp-
p-idy6c3e1fiqj/2",
    "CreationTime": "2021-02-24T17:00:14Z",
    "InferenceSpecification": {
      "Containers": [
        {
          "Image": "257758044811.dkr.ecr.us-east-2.amazonaws.com/sagemaker-xgboost:1.0-1-
cpu-py3",
          "ImageDigest":
 "sha256:4dc8a7e4a010a19bb9e0a6b063f355393f6e623603361bd8b105f554d4f0c004",
          "ModelDataUrl": "s3://sagemaker-project-p-idy6c3e1fiqj/versionedmp-p-
idy6c3e1fiqj/AbaloneTrain/pipelines-4r83jejmhorv-TrainAbaloneModel-xw869y8C4a/output/
model.tar.gz"
        }
      ],
      "SupportedContentTypes": [
        "text/csv"
      ],
      "SupportedResponseMIMETypes": [
        "text/csv"
      ]
    },
    "ModelPackageStatus": "Completed",
    "ModelPackageStatusDetails": {
      "ValidationStatuses": [],
      "ImageScanStatuses": []
    },
    "CertifyForMarketplace": false,
```

```
    "ModelApprovalStatus": "Rejected",
    "MetadataProperties": {
      "GeneratedBy": "arn:aws:sagemaker:us-east-2:123456789012:pipeline/versionedmp-p-
idy6c3e1fiqj/execution/4r83jejmhorv"
    },
    "ModelMetrics": {
      "ModelQuality": {
        "Statistics": {
          "ContentType": "application/json",
          "S3Uri": "s3://sagemaker-project-p-idy6c3e1fiqj/versionedmp-p-idy6c3e1fiqj/
script-2021-02-24-10-55-15-413/output/evaluation/evaluation.json"
        }
      }
    },
    "LastModifiedTime": "2021-02-24T17:00:14Z"
  }
}
```

管道执行状态更改

指示 SageMaker 管道执行的状态更改。

```
{
  "version": "0",
  "id": "315c1398-40ff-a850-213b-158f73kd93ir",
  "detail-type": "SageMaker Model Building Pipeline Execution Status Change",
  "source": "aws.sagemaker",
  "account": "123456789012",
  "time": "2021-03-15T16:10:11Z",
  "region": "us-east-1",
  "resources": ["arn:aws:sagemaker:us-east-1:123456789012:pipeline/myPipeline-123",
 "arn:aws:sagemaker:us-east-1:123456789012:pipeline/myPipeline-123/execution/
p4jn9xou8a8s"],
  "detail": {
    "pipelineExecutionDisplayName": "SomeDisplayName",
    "currentPipelineExecutionStatus": "Succeeded",
    "previousPipelineExecutionStatus": "Executing",
    "executionStartTime": "2021-03-15T16:03:13Z",
    "executionEndTime": "2021-03-15T16:10:10Z",
    "pipelineExecutionDescription": "SomeDescription",
    "pipelineArn": "arn:aws:sagemaker:us-east-1:123456789012:pipeline/myPipeline-123",
    "pipelineExecutionArn": "arn:aws:sagemaker:us-east-1:123456789012:pipeline/
myPipeline-123/execution/p4jn9xou8a8s"
  }
}
```

管道步骤状态更改

指示 SageMaker 管道步骤的状态更改。

如果有缓存命中，该事件将包含`cacheHitResult`字段中返回的子位置类型。

如果`currentStepStatus`是`Failed`，该事件包含`failureReason`字段，提供了有关步骤失败的原因的说明。

```
{
  "version": "0",
  "id": "ea37ccbb-5e2b-05e9-4073-1daazc940304",
  "detail-type": "SageMaker Model Building Pipeline Execution Step Status Change",
  "source": "aws.sagemaker",
```

```
  "account": "123456789012",
  "time": "2021-03-15T16:10:10Z",
  "region": "us-east-1",
  "resources": ["arn:aws:sagemaker:us-east-1:123456789012:pipeline/myPipeline-123",
 "arn:aws:sagemaker:us-east-1:123456789012:pipeline/myPipeline-123/execution/
p4jn9xou8a8s"],
  "detail": {
    "metadata": {
      "processingJob": {
        "arn": "arn:aws:sagemaker:us-east-1:123456789012:processing-job/pipelines-
p4jn9xou8a8s-myprocessingstep1-tmgxry49ug"
      }
    },
    "stepStartTime": "2021-03-15T16:03:14Z",
    "stepEndTime": "2021-03-15T16:10:09Z",
    "stepName": "myprocessingstep1",
    "stepType": "Processing",
    "previousStepStatus": "Executing",
    "currentStepStatus": "Succeeded",
    "pipelineArn": "arn:aws:sagemaker:us-east-1:123456789012:pipeline/myPipeline-123",
    "pipelineExecutionArn": "arn:aws:sagemaker:us-east-1:123456789012:pipeline/
myPipeline-123/execution/p4jn9xou8a8s"
  }
}
```

SageMaker 映像状态更改

指示 SageMaker 映像的状态更改。

```
{
  "version": "0",
  "id": "cee033a3-17d8-49f8-865f-b9ebf485d9ee",
  "detail-type": "SageMaker Image State Change",
  "source": "aws.sagemaker",
  "account": "123456789012",
  "time": "2021-04-29T01:29:59Z",
  "region": "us-east-1",
  "resources": ["arn:aws:sagemaker:us-west-2:123456789012:image/cee033a3-17d8-49f8-865f-
b9ebf485d9ee"],
  "detail": {
    "ImageName": "cee033a3-17d8-49f8-865f-b9ebf485d9ee",
    "ImageArn": "arn:aws:sagemaker:us-west-2:123456789012:image/cee033a3-17d8-49f8-865f-
b9ebf485d9ee",
    "ImageStatus": "Creating",
    "Version": 1.0,
    "Tags": {}
  }
}
```

SageMaker 映像版本状态更改

指示 SageMaker 映像版本的状态更改。

```
{
  "version": "0",
  "id": "07fc4615-ebd7-15fc-1746-243411f09f04",
  "detail-type": "SageMaker Image Version State Change",
  "source": "aws.sagemaker",
  "account": "123456789012",
  "time": "2021-04-29T01:29:59Z",
  "region": "us-east-1",
```

```
  "resources": ["arn:aws:sagemaker:us-west-2:123456789012:image-
version/07800032-2d29-48b7-8f82-5129225b2a85"],
  "detail": {
    "ImageArn": "arn:aws:sagemaker:us-west-2:123456789012:image/a70ff896-c832-4fe8-add6-
eba25a0f43e6",
    "ImageVersionArn": "arn:aws:sagemaker:us-west-2:123456789012:image-
version/07800032-2d29-48b7-8f82-5129225b2a85",
    "ImageVersionStatus": "Creating",
    "Version": 1.0,
    "Tags": {}
  }
}
```

有关 SageMaker 作业、终端节点和管道的状态值及其含义的更多信息，请参阅以下链接：

- AlgorithmStatus
- EndpointStatus
- FeatureGroupStatus
- HyperParameterTuningJobStatus
- LabelingJobStatus
- ModelPackageStatus
- NotebookInstanceStatus
- PipelineExecutionStatus
- StepStatus
- ProcessingJobStatus
- TrainingJobStatus
- TransformJobStatus

有关更多信息，请参阅 Amazon EventBridge 用户指南。

端节点部署状态更改

> **Important**
>
> 以下示例可能不适用于所有终端节点。有关可能排除终端节点的功能列表，请参阅Exclusions (p. 2251)页.

表示终端节点部署的状态更改。以下示例显示了使用蓝色/绿色 Canary 部署进行更新的端点。

```
{
    "version": "0",
    "id": "0bd4a141-0a02-9d8a-f977-3924c3fb259c",
    "detail-type": "SageMaker Endpoint Deployment State Change",
    "source": "aws.sagemaker",
    "account": "123456789012",
    "time": "2021-10-25T01:52:12Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:sagemaker:us-west-2:651393343886:endpoint/sample-endpoint"
    ],
    "detail": {
        "EndpointName": "sample-endpoint",
        "EndpointArn": "arn:aws:sagemaker:us-west-2:651393343886:endpoint/sample-endpoint",
        "EndpointConfigName": "sample-endpoint-config-1",
        "ProductionVariants": [
            {
                "VariantName": "AllTraffic",
```

```
                "CurrentWeight": 1,
                "DesiredWeight": 1,
                "CurrentInstanceCount": 3,
                "DesiredInstanceCount": 3
            }
        ],
        "EndpointStatus": "UPDATING",
        "CreationTime": 1635195148181,
        "LastModifiedTime": 1635195148181,
        "Tags": {},
        "PendingDeploymentSummary": {
            "EndpointConfigName": "sample-endpoint-config-2",
            "StartTime": Timestamp,
            "ProductionVariants": [
                {
                    "VariantName": "AllTraffic",
                    "CurrentWeight": 1,
                    "DesiredWeight": 1,
                    "CurrentInstanceCount": 1,
                    "DesiredInstanceCount": 3,
                    "VariantStatus": [
                        {
                            "Status": "Baking",
                            "StatusMessage": "Baking for 600 seconds
 (TerminationWaitInSeconds) with traffic enabled on canary capacity of 1 instance(s).",
                            "StartTime": 1635195269181,
                        }
                    ]
                }
            ]
        }
    }
}
```

以下示例表示终端节点部署的状态更改，该部署正在使用现有终端节点配置上的新容量进行更新。

```
{
    "version": "0",
    "id": "0bd4a141-0a02-9d8a-f977-3924c3fb259c",
    "detail-type": "SageMaker Endpoint Deployment State Change",
    "source": "aws.sagemaker",
    "account": "123456789012",
    "time": "2021-10-25T01:52:12Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:sagemaker:us-west-2:651393343886:endpoint/sample-endpoint"
    ],
    "detail": {
        "EndpointName": "sample-endpoint",
        "EndpointArn": "arn:aws:sagemaker:us-west-2:651393343886:endpoint/sample-endpoint",
        "EndpointConfigName": "sample-endpoint-config-1",
        "ProductionVariants": [
            {
                "VariantName": "AllTraffic",
                "CurrentWeight": 1,
                "DesiredWeight": 1,
                "CurrentInstanceCount": 3,
                "DesiredInstanceCount": 6,
                "VariantStatus": [
                    {
                        "Status": "Updating",
                        "StatusMessage": "Scaling out desired instance count to 6.",
                        "StartTime": 1635195269181,
                    }
                ]
```

```
            }
        ],
        "EndpointStatus": "UPDATING",
        "CreationTime": 1635195148181,
        "LastModifiedTime": 1635195148181,
        "Tags": {},
    }
```

以下辅助部署状态也可用于终端节点 (在VariantStatus对象。

- Creating：为生产变体创建实例。

  示例消息："Launching X instance(s)."
- Deleting：终止生产变体的实例。

  示例消息："Terminating X instance(s)."
- Updating：更新生产变体的容量。

  示例消息："Launching X instance(s)."、"Scaling out desired instance count to X."
- ActivatingTraffic：启用生产变体的流量。

  示例消息："Activating traffic on canary capacity of X instance(s)."
- Baking：在自动回滚配置中监控 CloudWatch 警报的等待时间。

  示例消息："Baking for X seconds (TerminationWaitInSeconds) with traffic enabled on full capacity of Y instance(s)."

Amazon SageMaker 的 API 参考指南

概览

Amazon SageMaker 提供可用于创建和管理笔记本实例以及训练和部署模型的 API、开发工具包和命令行界面。

- Amazon SageMaker Python 开发工具包(推荐)
- Amazon SageMaker API 参考
- Amazon Augmented AI API 参考
- Amazon Command Line Interface
- Amazon SDK for .NET
- Amazon SDK for C++
- Amazon SDK for Go
- Amazon SDK for Java
- Amazon SDK for JavaScript
- Amazon SDK for PHP
- Amazon SDK for Python (Boto)
- Amazon SDK for Ruby
- Amazon SageMaker Spark

您还可以从 Amazon SageMaker 示例笔记本 GitHub 存储库获取代码示例。

- 示例笔记本

Amazon SageMaker 的编程模型

直接从代码进行 API 调用会很麻烦，并且需要您编写代码来对请求进行身份验证。Amazon SageMaker 提供以下备选方案：

- 使用 SageMaker 控制台— 通过控制台，您无需编写任何代码。您使用控制台 UI 启动模型训练或部署模型。该控制台适用于简单作业，您在这些作业中使用内置的训练算法，并且无需对训练数据进行预处理。

- 修改示例 Jupyter 笔记本—SageMaker 提供几个使用特定算法和数据集来训练和部署模型的 Jupyter 笔记本。先从具有合适算法的笔记本开始，并对其进行修改，以满足您的数据源和特定需求。

- 从头编写模型训练和推理代码—SageMaker 提供了多个AmazonSDK 语言（在概述中列出）和Amazon SageMaker Python 开发工具包，高级 Python 库，您可以在代码中使用它们，以启动模型训练作业并部署生成的模型。

  - Amazon SageMaker Python 开发工具包— 此 Python 库简化了模型训练和部署。除了对请求进行身份验证之外，该库还通过提供简单的方法和默认参数来提取平台具体信息。例如：

- 要部署模型，只需调用 `deploy()` 方法即可。该方法会创建 SageMaker 模型构件和终端节点配置，然后在终端节点上部署模型。

- 如果您使用自定义框架脚本进行模型训练，则调用 `fit()` 方法。该方法会为您的脚本创建一个 .gzip 文件，将其上传到 Amazon S3 位置，然后针对模型训练和其他任务而运行它。有关更多信息，请参阅 将 Machine Learning 框架、Python 和 R 与 Amazon SageMaker 结合使用 (p. 11)。

- 这些区域有：Amazon软件开发工具包— 开发工具包提供对应于 SageMaker API 的方法（请参阅`Operations`）。使用开发工具包以编程方式启动模型训练任务，并在 SageMaker 中托管模型。开发工具包客户端通过使用您的访问密钥对请求进行身份验证，因此您无需编写身份验证代码。它们提供有多种语言和平台版本。有关更多信息，请参阅概览中前面的列表。

  In开始使用 Amazon SageMaker (p. 30)，您使用 SageMaker 提供的算法来训练和部署模型。该练习说明了如何使用这两个库。有关更多信息，请参阅 开始使用 Amazon SageMaker (p. 30)。

- 将 SageMaker 集成到你的 Apache Spark 工作流程—SageMaker 提供了一个用于从 Apache Spark 调用 API 的库。借助该库，您可以在 Apache Spark 管道中使用基于 SageMaker 的估算器。有关更多信息，请参阅将 Apache Spark 与 Amazon SageMaker 结合使用 (p. 12)。

Amazon SageMaker 的文档历史记录

update-history-change	update-history-description	update-history-date
re: Inent 2021 新特色 (p. 2837)	SageMaker 画布	2021 年 12 月 1 日
re: Inent 2021 新特色 (p. 2837)	SageMaker Ground Truth Plus	2021 年 12 月 1 日
re: Inent 2021 新特色 (p. 2837)	SageMaker 推理推荐器	2021 年 12 月 1 日
re: Inent 2021 新特色 (p. 2837)	SageMaker 无服务器端节点	2021 年 12 月 1 日
re: Inent 2021 新特色 (p. 2837)	SageMaker Studio 实验室	2021 年 12 月 1 日
re: Inent 2021 新特色 (p. 2837)	SageMaker Studio 笔记本和 Amazon EMR	2021 年 12 月 1 日
re: Inent 2021 新特色 (p. 2837)	SageMaker 培训编译器	2021 年 12 月 1 日
时间序列数据 (p. 2837)	Amazon SageMaker Autopilot 接受时间序列作为模型输入。有关更多信息，请参阅 。Amazon SageMaker 自动驾驶仪数据和问题类型。.	2021 年 10 月 25 日
Amazon托管策略 (p. 2837)	已开启跟踪 SageMaker 的更改托管策略.	2021 年 6 月 10 日
SageMaker 澄清更新 1.0.5	对于具有多个输出的模型，CSV 文件中的实例说明现在可以正确映射到 v1.0.5 中的右列。此外，改进了在内核 SHAP 中使用 Logit 时返回 0 和 1 作为类概率的模型的数值处理。	2021 年 4 月 22 日
re: Inent 2020 新特色 (p. 2837)	Amazon SageMaker 模型构建管道、使用 SageMaker 项目自动执行 mLOP、SageMaker 边缘管理器、SageMaker 澄清、SageMaker 数据牧马人、SageMaker Feature Store、JumpStart SageMaker Studio、使用模型注册表注册并部署模型、SageMaker 分布式、使用 SageMaker 调试器进行深度分	2020 年 12 月 1 日
Studio Notebooks (p. 2837)	SageMaker Studio 笔记本	2020 年 4 月 28 日
re:Invent 2019 新特色 (p. 2837)	SageMaker Studio、SageMaker Studio 笔记本（预览版），SageMaker 实验、SageMaker Autopilot、SageMaker 调试程序、SageMaker 模型监控器	2019 年 12 月 3 日

re:Invent 2018 新特色 (p. 2837)	Amazon SageMaker Ground Truth、在 Amazon SageMaker 中使用 Elastic Inference、SageMaker 资源Amazon Web Services Marketplace、SageMaker 推理管道、SageMaker Neo、通过搜索管理机器学习实验、在 Amazon SageMaker 中使用强化学习、将 Git 存储库与 Amazon SageMaker 笔记本实例关联、语义分割、在 TTrainingJobs 中使用增强清单文件	2018 年 11 月 28 日
配置笔记本实例 (p. 2837)	可以在创建或启动笔记本实例时使用 Shell 脚本配置该实例。有关更多信息，请参阅自定义笔记本实例。	2018 年 5 月 1 日
Application Auto Scaling 支持 (p. 2837)	Amazon SageMaker 现在支持面向生产变体的 Application Auto Scaling。想要了解有关信息，请参阅自动扩展 SageMaker 模型	2018 年 2 月 28 日
TensorFlow 1.5 和 MXNet 1.0 支持 (p. 2837)	Amazon SageMaker 深度学习容器现在支持 TensorFlow 1.5 和 Apache MXNet 1.0。	2018 年 2 月 27 日
BlazingText 算法 (p. 2837)	Amazon SageMaker 现在支持BlazingText算法。	2018 年 1 月 18 日
KMS 加密 (p. 2837)	Amazon SageMaker 现在支持用于托管实例和训练静态模型构件的 KMS 加密。	2018 年 1 月 17 日
CloudTrail 支持 (p. 2837)	Amazon SageMaker 现在支持使用登录Amazon CloudTrail.	2018 年 1 月 11 日
DeepAR 预测算法 (p. 2837)	Amazon SageMaker 现在支持DeepAR用于时间序列预测的算法。	2018 年 1 月 8 日

Amazon词汇表

有关最新Amazon术语，请参阅《Amazon一般参考》中的Amazon术语表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。