
Amazon CloudWatch

用户指南

亚马逊云科技



Amazon CloudWatch: 用户指南

Copyright © 2022 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

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

Table of Contents

什么是 Amazon CloudWatch ?	1
访问 CloudWatch	1
相关 Amazon 服务	1
CloudWatch 的工作原理	1
概念	2
命名空间	3
指标	3
维度	4
解析	5
统计数据	5
单位	5
时间段	5
聚合	6
百分位数	6
告警	7
账单和成本	7
使用 Cost Explorer 分析 CloudWatch 成本和使用数据	8
使用 Amazon Cost and Usage Report 和 Athena 分析 CloudWatch 成本和使用数据	11
优化和降低成本的最佳实践	13
资源	21
开始设置	22
注册 Amazon Web Services (Amazon)	22
登录 Amazon CloudWatch 控制台	22
设置 Amazon CLI	22
开始使用	23
查看所有 Amazon 服务中的关键指标	25
使服务不出现在跨服务控制面板中	26
专注于单个服务	26
专注于资源组	27
控制面板	29
创建控制面板	29
跨账户跨区域的控制面板	30
利用 Amazon Web Services Management Console 创建和使用跨账户跨区域的控制面板	30
以编程方式创建跨账户跨区域的控制面板	31
在 CloudWatch 控制面板上创建和使用小组件	33
添加或删除图表	33
在 CloudWatch 控制面板上手动绘制指标图表	35
编辑图表	36
将 Explorer 小组件添加到 CloudWatch 控制面板	40
添加或删除折线图小组件	42
添加或删除数字小组件	42
添加或删除量规图小组件	43
将自定义小组件添加到 CloudWatch 控制面板	44
添加或删除文本小组件	51
添加或删除警报小组件	52
链接和取消链接图表	53
共享控制面板	53
共享控制面板所需的权限	54
授予与您共享控制面板的人员的权限	55
与特定用户共享单个控制面板	55
公开共享单个控制面板	56
使用 SSO 共享账户中的所有 CloudWatch 控制面板	56
为 CloudWatch 控制面板共享设置 SSO	57
查看已共享的控制面板的数量	57

查看您的哪些控制面板已共享	58
停止共享一个或多个控制面板	58
查看共享控制面板权限并更改权限范围	58
允许与您共享的人员查看复合告警	59
允许与您共享的人员查看日志表小组件	60
允许与您共享的人员查看自定义小组件	61
使用实时数据	62
查看动画控制面板	62
将控制面板添加到收藏夹列表	63
更改时间段覆盖设置或刷新间隔	63
更改时间范围或时区格式	64
指标	66
基本监控和详细监控	66
使用 CloudWatch Metrics Insights 查询您的指标	67
构建查询	68
查询组件和语法	68
将 Metrics Insights 查询与指标数学配合使用	73
SQL 推理	74
示例查询	75
Metrics Insights 限制	81
Metrics Insights 术语表	81
Metrics Insights 问题排查	81
使用 Metrics Explorer 按标签和属性监控资源	82
适合 Metrics Explorer 的 CloudWatch 代理配置	83
使用指标流	83
设置指标流	84
可以流式传输的统计数据	87
指标流操作和维护	88
使用 CloudWatch 指标监控您的指标流	88
CloudWatch 和 Kinesis Data Firehose 之间的信任关系	89
指标流输出格式	90
问题排查	100
查看可用指标	100
搜索可用指标	103
绘制指标的图表	104
绘制指标的图表	104
使用动态标签	107
修改图表的时间范围或时区格式	109
放大图表	110
修改图表的 Y 轴	112
从图表上的指标创建告警	112
使用异常检测	113
异常检测的工作原理	115
指标数学异常检测	115
使用指标数学	116
向 CloudWatch 图表中添加数学表达式	116
指标数学语法和函数	117
使用 IF 表达式	128
指标数学异常检测	130
在图表中使用搜索表达式	131
搜索表达式语法	131
搜索表达式示例	135
使用搜索表达式创建图表	136
获取指标的统计数据	138
CloudWatch 统计数据定义	138
获取特定资源的统计数据	140
跨资源聚合统计数据	143

按 Auto Scaling 组聚合统计数据	145
按 AMI 聚合统计数据	146
发布自定义指标	147
高精度指标	147
使用维度	148
发布单一数据点	148
发布统计数据集	149
发布值零	149
停止发布指标	150
告警	151
指标告警状态	151
评估告警	152
告警操作	152
配置告警处理缺失数据的方式	153
在数据缺失时如何评估告警状态	153
高精度告警	155
针对数学表达式的告警	156
基于百分位数的告警和小数据样本	156
CloudWatch 告警和 Amazon EventBridge	156
CloudWatch 告警的常见功能	156
设置 SNS 主题	157
使用 Amazon Web Services Management Console 设置 Amazon SNS 主题	157
使用 Amazon CLI 设置 SNS 主题	158
根据静态阈值创建告警	159
根据异常检测创建告警	161
修改异常检测模型	163
删除异常检测模型	163
根据日志组指标筛选条件创建告警	163
根据指标数学表达式创建告警	164
创建复合告警	166
创建复合告警	167
复合告警操作抑制	168
编辑或删除 CloudWatch 告警	172
创建 CPU 使用率告警	174
使用 Amazon Web Services Management Console 设置 CPU 使用率告警	174
使用 Amazon CLI 设置 CPU 使用率告警	175
创建负载均衡器延迟告警	175
使用 Amazon Web Services Management Console 设置延迟告警	175
使用 Amazon CLI 设置延迟告警	176
创建存储吞吐量告警	176
使用 Amazon Web Services Management Console 设置存储吞吐量告警	177
使用 Amazon CLI 设置存储吞吐量告警	177
创建告警以停止、终止、重启或恢复 EC2 实例	178
在 Amazon CloudWatch 告警中添加停止操作	179
在 Amazon CloudWatch 告警中添加终止操作	179
在 Amazon CloudWatch 告警中添加重启操作	180
在 Amazon CloudWatch 告警中添加恢复操作	181
查看已触发的告警和操作的历史记录	182
创建账单告警	182
启用账单提醒	183
创建账单告警	183
删除账单告警	184
隐藏 Auto Scaling 警报	184
告警事件和 EventBridge	185
来自 CloudWatch 的示例事件	185
应用程序监控	194
使用 ServiceLens 监控应用程序的运行状况	194

部署 ServiceLens	196
使用服务地图	203
使用跟踪视图	205
使用资源运行状况视图	205
ServiceLens 问题排查	207
使用 Synthetics 监控	208
所需角色和权限	209
创建金丝雀	219
组	274
排查失败金丝雀的问题	275
金丝雀脚本示例代码	278
金丝雀和 X-Ray 跟踪	282
在 VPC 上运行金丝雀	283
加密金丝雀构件	283
查看金丝雀统计数据和详细信息	285
金丝雀发布的 CloudWatch 指标	286
编辑或删除金丝雀	287
使用 Amazon EventBridge 监控金丝雀事件	289
使用 CloudWatch Evidently 执行启动和 A/B 实验	292
使用 Evidently 的 IAM 策略	293
创建项目、功能、启动和实验	294
管理功能、启动和实验	306
将代码添加到应用程序	308
项目数据存储	310
Evidently 是如何计算结果的	312
在控制面板中查看启动结果	313
在控制面板中查看实验结果	313
CloudWatch Evidently 如何收集和存储数据	314
使用服务相关角色	315
CloudWatch Evidently 配额	316
教程：使用 Evidently 示例应用程序进行 A/B 测试	317
使用 CloudWatch RUM	323
使用 CloudWatch RUM 的 IAM 策略	325
设置应用程序以使用 CloudWatch RUM	325
配置 CloudWatch RUM Web 客户端	331
使用页面组	331
查看 CloudWatch RUM 控制面板	332
您可以使用 CloudWatch RUM 收集的 CloudWatch 指标	333
CloudWatch RUM 的数据保护和数据隐私	335
CloudWatch RUM Web 客户端收集的信息	335
管理使用 CloudWatch RUM 的应用程序	350
CloudWatch RUM 配额	351
问题排查	351
洞察力和操作可视性	353
使用 Container Insights	353
支持的平台	353
CloudWatch 代理容器镜像	354
支持的区域	354
设置 Container Insights	355
查看 Container Insights 指标	390
Container Insights 收集的指标	393
性能日志参考	398
Container Insights Prometheus 指标监控	418
与 Application Insights 集成	493
在 Container Insights 中查看 Amazon ECS 生命周期事件	493
Container Insights 问题排查	494
构建您自己的 CloudWatch 代理 Docker 镜像	496

在容器中部署其他 CloudWatch 代理功能	496
使用 Contributor Insights 分析高基数数据	497
创建 Contributor Insights 规则	497
Contributor Insights 规则语法	500
示例规则	502
查看 Contributor Insights 报告	505
绘制规则生成的指标的图表	505
使用 Contributor Insights 内置规则	507
使用 CloudWatch Application Insights 检测常见应用程序问题	508
Amazon CloudWatch Application Insights 是什么？	508
Application Insights 的工作方式	512
开始使用	518
使用组件配置	536
使用 CloudFormation 模板	572
教程：为 SAP HANA 设置监控	581
教程：为 .NET 和 SQL 设置监控	589
查看和排查 Application Insights	594
支持的日志和指标	595
跨账户跨区域的 CloudWatch 控制台	654
启用跨账户跨区域功能	654
（可选）与 Amazon Organizations 集成	656
故障排除	657
使用跨账户后禁用和清理	657
使用 CloudWatch 代理收集指标和日志	659
安装 CloudWatch 代理	660
使用命令行安装 CloudWatch 代理	660
使用 Systems Manager 安装 CloudWatch 代理	669
验证 CloudWatch 代理软件包的签名	681
创建 CloudWatch 代理配置文件	682
使用向导创建 CloudWatch 代理配置文件	682
手动创建或编辑 CloudWatch 代理配置文件	686
CloudWatch 代理收集的指标	729
Windows Server 实例上的 CloudWatch 代理收集的指标	729
Linux 和 macOS 实例上的 CloudWatch 代理收集的指标	729
CloudWatch 代理对 OpenTelemetry 的支持	737
IAM 权限	738
CloudWatch 代理 OpenTelemetry 配置	738
使用命令行管理具有 OpenTelemetry 支持的 CloudWatch 代理	738
使用 Systems Manager 通过嵌入式 OpenTelemetry Collector 管理 CloudWatch 代理	739
生成 OpenTelemetry 的指标和跟踪	741
CloudWatch 代理的常见场景	741
以不同用户身份运行 CloudWatch 代理	742
将自定义维度添加到 CloudWatch 代理收集的指标	743
多个 CloudWatch 代理配置文件	743
汇总或累积 CloudWatch 代理收集的指标	745
使用 CloudWatch 代理收集高精度指标	745
向不同账户发送指标和日志	746
统一的 CloudWatch 代理和更早的 CloudWatch Logs 代理之间的时间戳差异	747
CloudWatch 代理故障排除	748
CloudWatch 代理命令行参数	748
使用 Run Command 安装 CloudWatch 代理失败	748
CloudWatch 代理无法启动	749
验证 CloudWatch 代理是否正在运行	749
CloudWatch 代理未启动，并且错误中提及了 Amazon EC2 区域	750
无法在 Windows Server 上启动 CloudWatch 代理	750
在 Windows Server 上找不到凭证	750
指标存储在何处？	750

CloudWatch 代理需要很长时间才能在容器中运行或记录跃点数限制错误	751
我更新了代理配置，但在 CloudWatch 控制台中看不到新的指标或日志	751
CloudWatch 代理文件和位置	751
CloudWatch 代理生成的日志	752
停止和重新启动 CloudWatch 代理	753
采用 CloudWatch 嵌入式指标格式摄取高基数日志以生成指标	754
使用嵌入式指标格式生成日志	754
使用客户端库	754
通过 Amazon Distro for OpenTelemetry 使用嵌入式指标格式	755
手动生成嵌入式指标格式日志	755
在控制台中查看您的指标和日志	767
发布 CloudWatch 指标的服务	768
Amazon 使用情况指标	775
可视化 Service Quotas 并设置告警	775
Amazon API 使用情况指标	776
CloudWatch 使用情况指标	781
CloudWatch 教程	783
方案：监控估计费用	783
步骤 1：启用账单提醒	783
步骤 2：创建账单告警	784
步骤 3：检查告警状态	784
步骤 4：编辑账单告警	785
步骤 5：删除账单告警	785
方案：发布指标	785
步骤 1：定义数据配置	786
步骤 2：将指标添加到 CloudWatch	786
步骤 3：获取 CloudWatch 中的统计数据	787
步骤 4：使用控制台查看图表	787
使用 Amazon SDK	788
代码示例	789
操作	789
创建监控指标的告警	790
删除警报	795
描述告警历史记录	800
为指标描述告警	801
禁用警报操作	805
启用警报操作	810
获取控制面板详细信息	816
获取指标统计数据	817
列出控制面板	818
列出指标	819
将一组数据放入指标	825
将数据放入指标	826
场景	830
管理 Amazon CloudWatch 指标和告警	830
安全性	836
数据保护	836
传输中加密	837
Identity and Access Management	837
身份验证	837
访问控制	838
CloudWatch 控制面板权限更新	838
有关管理访问的概述	839
使用基于身份的策略 (IAM policy)	841
使用条件键限制对 CloudWatch 命名空间的访问	861
使用条件键限制 Contributor Insights 用户对日志组的访问	862
使用条件键限制告警操作	863

使用服务相关角色	863
对 CloudWatch RUM 使用服务相关角色	868
在 Application Insights 中使用服务相关角色	871
Amazon 适用于 Application Insights 的 托管式策略	878
Amazon CloudWatch 权限参考	882
合规性验证	890
故障恢复能力	890
基础设施安全性	890
网络隔离	891
接口 VPC 终端节点	891
CloudWatch	891
CloudWatch Synthetics	893
Synthetics 金丝雀的安全注意事项	894
使用安全连接	894
金丝雀命名注意事项	894
金丝雀代码中的密钥和敏感信息	894
权限注意事项	895
堆栈跟踪和异常消息	895
缩小 IAM 角色的范围	895
敏感数据编辑	895
使用 Amazon CloudTrail 记录 API 调用	897
CloudTrail 中的 CloudWatch 信息	897
示例：CloudWatch 日志文件条目	898
CloudTrail 中的 CloudWatch Synthetics 信息	900
示例：CloudWatch Synthetics 日志文件条目	900
标记 CloudWatch 资源	903
CloudWatch 中支持的资源	903
管理标签	903
标签命名和使用约定	903
Grafana 集成	905
服务限额	906
文档历史记录	909

什么是 Amazon CloudWatch ?

Amazon CloudWatch 可实时监控您的亚马逊云科技 (Amazon) 资源以及您在 Amazon 上运行的应用程序。您可以使用 CloudWatch 收集和跟踪指标，这些指标是您可衡量的相关资源和应用程序的变量。

CloudWatch 主页自动显示有关您使用的每项 Amazon 服务的指标。此外，您还可以创建自定义控制面板，以显示有关自定义应用程序的指标，并显示您选择的指标的自定义集合。

您可以创建警报，这些警报监视指标，当超出阈值时，它们会发送通知或者对您所监控的资源自动进行更改。例如，您可以监控您的 Amazon EC2 实例的 CPU 使用率以及磁盘读写情况，然后使用此数据确定您是否应启动其它实例来处理增加的负载。您还可以使用此数据停止未完全利用的实例以节省开支。

您可通过使用 CloudWatch 了解系统范围的资源使用率、应用程序性能和运行状况。

访问 CloudWatch

您可以使用以下任何方式访问 CloudWatch：

- Amazon CloudWatch 控制台 – <https://console.aws.amazon.com/cloudwatch/>
- Amazon CLI – 有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的[开始设置 Amazon Command Line Interface](#)。
- CloudWatch API – 有关更多信息，请参阅[Amazon CloudWatch API 参考](#)。
- Amazon SDK – 有关更多信息，请参阅[亚马逊云科技的工具](#)。

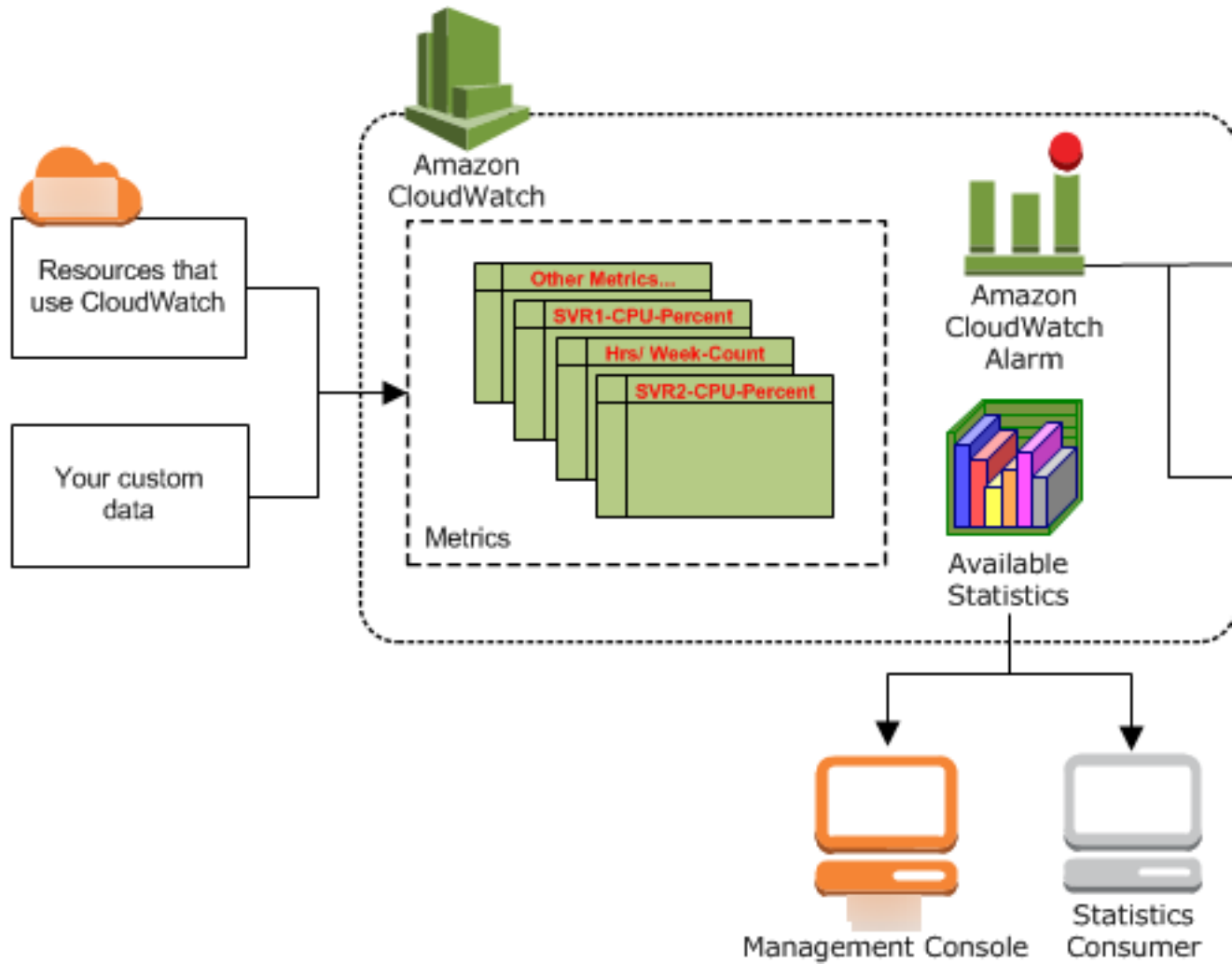
相关 Amazon 服务

以下服务可与 Amazon CloudWatch 一起使用：

- Amazon Simple Notification Service (Amazon SNS) 可协调和管理向订阅端点或客户端传送或发送消息的过程。您可结合使用 Amazon SNS 与 CloudWatch 以便在达到告警阈值时发送消息。有关更多信息，请参阅[设置 Amazon SNS 通知 \(p. 157\)](#)。
- Amazon EC2 Auto Scaling 支持根据用户定义的策略、运行状况检查和时间表自动启动或终止 Amazon EC2 实例。您可将 CloudWatch 警报与 Amazon EC2 Auto Scaling 一起使用以按需扩展 EC2 实例。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的[动态扩缩](#)。
- Amazon CloudTrail 支持监控针对您的账户向 Amazon CloudWatch API 做出的调用（包括由 Amazon Web Services Management Console、Amazon CLI 和其他服务进行的调用）。开启 CloudTrail 日志记录后，CloudWatch 将日志文件写入到您在配置 CloudTrail 时指定的 Amazon S3 存储桶。有关更多信息，请参阅[使用 Amazon CloudTrail 记录 Amazon CloudWatch API 调用 \(p. 897\)](#)。
- Amazon Identity and Access Management (IAM) 是一种 Web 服务，可帮助您安全地控制用户对 Amazon 资源的访问权限。使用 IAM 可以控制可使用您的 Amazon 资源（身份验证）的人员、他们可以使用的资源以及使用这些资源的方式（授权）。有关更多信息，请参阅[适用于 Amazon CloudWatch 的 Identity and Access Management \(p. 837\)](#)。

说明 Amazon CloudWatch 的工作原理

Amazon CloudWatch 本质上是一个指标存储库。Amazon 服务（例如 Amazon EC2）会将指标放在存储库中，而您可以根据这些指标来检索统计数据。如果将自己的自定义指标放在存储库中，则还可以检索有关这些指标的统计数据。



您可以在 CloudWatch 控制台中使用指标计算统计数据，然后以图形化的方式显示数据。有关生成指标并将其发送到 CloudWatch 的其他 Amazon 资源的更多信息，请参阅 [发布 CloudWatch 指标的 Amazon 服务 \(p. 768\)](#)。

在满足特定条件时，您可以配置告警操作以停止、启动或终止 Amazon EC2 实例。此外，您可创建代您启动 Amazon EC2 Auto Scaling 和 Amazon Simple Notification Service (Amazon SNS) 操作的告警。有关创建 CloudWatch 警报的更多信息，请参阅 [告警 \(p. 7\)](#)。

Amazon 云计算资源存储在具有高度可用性的数据中心设施中。要提供额外的可扩展性和可靠性，每个数据中心设施应位于称为区域的特定地理区域。从设计而言，每个区域都与其他区域完全隔离，以实现最大程度的故障隔离和稳定性。指标单独存储在区域中，但您可以使用 CloudWatch 跨区域功能来聚合来自不同区域的统计数据。有关更多信息，请参阅亚马逊云科技一般参考中的 [跨账户跨区域的 CloudWatch 控制台 \(p. 654\)](#) 以及 [区域和端点](#)。

Amazon CloudWatch 的概念

下面是便于您了解和使用 Amazon CloudWatch 的核心术语和概念：

- [命名空间 \(p. 3\)](#)

- [指标 \(p. 3\)](#)
- [维度 \(p. 4\)](#)
- [解析 \(p. 5\)](#)
- [统计数据 \(p. 5\)](#)
- [百分位数 \(p. 6\)](#)
- [告警 \(p. 7\)](#)

有关 CloudWatch 指标、警报、API 请求和警报电子邮件通知的服务限额的信息，请参阅 [CloudWatch 服务限额](#)。

命名空间

命名空间是用于 CloudWatch 指标的容器。不同命名空间中的指标彼此独立，因此来自不同应用程序的指标不会被错误地聚合到相同的统计信息中。

无默认命名空间。您必须为发布到 CloudWatch 的每个数据点指定命名空间。在创建指标时，您可以指定命名空间名称。这些名称必须包含有效的 XML 字符，且长度必须少于 256 个字符。可用字符包括：字母数字字符 (0-9A-Za-z)、句点 (.)、连字符 (-)、下划线 (_)、正斜杠 (/)、井号 (#) 和冒号 (:)。

Amazon 命名空间通常使用以下命名约定：AWS/*service*。例如，Amazon EC2 使用 AWS/EC2 命名空间。有关 Amazon 命名空间的列表，请参阅 [发布 CloudWatch 指标的 Amazon 服务 \(p. 768\)](#)。

指标

指标是 CloudWatch 中的基本概念。指标表示一个发布到 CloudWatch 并按时间排序的数据点集。可将指标视为要监控的变量，而数据点代表该变量随时间变化的值。例如，特定 EC2 实例的 CPU 使用率是 Amazon EC2 提供的一个指标。数据点本身可来自于您从中收集数据的任何应用程序或业务活动。

预设情况下，许多 Amazon 服务免费提供资源（例如 Amazon EC2 实例、Amazon EBS 卷和 Amazon RDS 数据库实例）的指标。如需收费，您也可以对某些资源（例如 Amazon EC2 实例）或您自己的应用程序指标启用详细监控。关于自定义指标，您可以您选择的任何顺序和任何速率添加数据点。您可以按一组有序的时间序列数据来检索关于这些数据点的统计数据。

指标仅存在于创建它们的区域中。指标无法删除，但如果在 15 个月没有向指标发布新数据，这些指标将自动过期。依据滚动机制，15 个月之前的数据点将过期；当新的数据点进入时，15 个月之前的数据将被丢弃。

指标是通过一个名称、一个命名空间以及零个或多个维度进行唯一定义的。指标中的每个数据点都有一个时间戳和一个度量单位（可选）。您可以从 CloudWatch 中检索任何指标的统计数据。

有关更多信息，请参阅 [查看可用指标 \(p. 100\)](#) 和 [发布自定义指标 \(p. 147\)](#)。

时间戳

每个指标数据点必须与一个时间戳关联。时间戳最长可以为过去的两周和将来的两小时。如果不提供时间戳，CloudWatch 会根据收到数据点的时间创建一个时间戳。

时间戳为 `dateTime` 对象，包含完整的日期以及小时、分钟和秒（例如，2016-10-31T23:59:59Z）。有关更多信息，请参阅 [dateTime](#)。尽管没有强制要求，但我们建议您使用协调世界时 (UTC)。从 CloudWatch 检索统计数据时，所有时间均采用 UTC。

CloudWatch 告警会基于当前时间 (UTC) 检查指标。发送到 CloudWatch 并且时间戳为非 UTC 时间的自定义指标会导致告警显示数据不足状态或产生延迟告警。

指标保留

CloudWatch 将保留指标数据，如下所示：

- 时间段短于 60 秒的数据点的可用时间为 3 小时。这些数据点是高精度自定义指标。
- 时间段为 60 秒 (1 分钟) 的数据点可用 15 天
- 时间段为 300 秒 (5 分钟) 的数据点可用 63 天
- 时间段为 3600 秒 (1 小时) 的数据点可用 455 天 (15 个月)

最初以较短时间段发布的数据点汇总在一起，可实现长期存储。例如，如果您使用 1 分钟的时间段收集数据，数据以 1 分钟的精度保持 15 天可用。15 天之后，此数据仍可用，但汇总在一起，只能以 5 分钟的精度检索。63 天之后，数据进一步汇总，以 1 小时的精度提供。

Note

控制台中不会显示在过去两周内没有任何新数据点的指标。当您在控制台的所有 metrics (全部指标) 选项卡的搜索框中键入指标名称或维度名称时，它们也不会显示，并且 `list-metrics` 命令的结果中不会返回它们。检索这些指标的最佳方法是使用 Amazon CLI 中的 `get-metric-data` 或者 `get-metric-statistics` 命令。

维度

维度是一个名称/值对，它是指标标识的一部分。您可以为一个指标分配最多 30 个维度。

每个指标包含用于描述该指标的特定特征，您可以将维度理解为这些特征的类别。维度可以帮助您设计统计数据计划的结构。因为维度是指标的唯一标识符的一部分，因此无论您在何时向一个指标添加唯一名称/值对，都会创建该指标的一个新变体。

向 CloudWatch 发送数据的 Amazon 服务会向每个指标附加维度。您可使用维度筛选 CloudWatch 返回的结果。例如，您可通过在搜索指标时指定 `InstanceId` 维度来获取特定 EC2 实例的统计数据。

对于由特定 Amazon 服务 (如 Amazon EC2) 生成的指标，CloudWatch 可以聚合多个维度的数据。例如，如果您在 `AWS/EC2` 命名空间中搜索指标但不指定任何维度，则 CloudWatch 会聚合指定指标的所有数据以创建您请求的统计数据。CloudWatch 不会跨多个维度聚合自定义指标的数据。

维度组合

CloudWatch 将维度的每种唯一组合视为一个单独的指标，即使指标具有相同的指标名称也是如此。您只能使用已发布的特定维度组合检索统计数据。当您检索统计数据时，为命名空间、指标名称和维度参数指定创建指标时使用的相同值。您还可指定 CloudWatch 要用于聚合的开始和结束时间。

例如，假设您在具有以下属性的 `DataCenterMetric` 命名空间中发布了 4 个名为 `ServerStats` 的不同指标：

```
Dimensions: Server=Prod, Domain=Frankfurt, Unit: Count, Timestamp: 2016-10-31T12:30:00Z, Value: 105
Dimensions: Server=Beta, Domain=Frankfurt, Unit: Count, Timestamp: 2016-10-31T12:31:00Z, Value: 115
Dimensions: Server=Prod, Domain=Rio, Unit: Count, Timestamp: 2016-10-31T12:32:00Z, Value: 95
Dimensions: Server=Beta, Domain=Rio, Unit: Count, Timestamp: 2016-10-31T12:33:00Z, Value: 97
```

如果您仅发布这 4 个指标，则可检索这些维度组合的统计数据：

- `Server=Prod,Domain=Frankfurt`
- `Server=Prod,Domain=Rio`

- `Server=Beta,Domain=Frankfurt`
- `Server=Beta,Domain=Rio`

如果您未指定任何维度，则无法检索以下维度的统计数据。（使用指标数学 SEARCH 函数时例外，该函数可以检索多个指标的统计数据。有关更多信息，请参阅 [在图表中使用搜索表达式 \(p. 131\)](#)。）

- `Server=Prod`
- `Server=Beta`
- `Domain=Frankfurt`
- `Domain=Rio`

解析

每个指标均为以下类型之一：

- 标准精度，数据粒度为一分钟
- 高精度，数据粒度为一秒

Amazon 服务生成的指标在默认情况下为标准精度。在发布自定义指标时，您可以将其定义为标准精度或高精度。发布高精度指标时，CloudWatch 使用 1 秒的精度来存储指标，您可以按照 1 秒、5 秒、10 秒、30 秒或 60 秒的任意倍数的时间段读取和检索。

高精度指标让您对应用程序的亚分钟级活动有着更详细的直观认识。请记住，每次对自定义指标的 `PutMetricData` 调用都会收取费用，因此对高精度指标频繁调用 `PutMetricData` 会导致较高的费用。有关 CloudWatch 定价的信息，请参阅 [Amazon CloudWatch 定价](#)。

如果对高精度指标设置告警，您可以指定 10 秒或 30 秒时间段的高精度告警，也可以设置 60 秒的任意倍数时间段的定期告警。10 秒或 30 秒时间段的高精度警报会产生较高的费用。

统计数据

统计数据是指定时间段内的指标数据汇总。CloudWatch 提供统计数据的依据是您的自定义数据所提供的指标数据点，或其他 Amazon 服务向 CloudWatch 提供的指标数据点。聚合通过使用命名空间、指标名称、维度以及数据点度量单位在您指定的时间段内完成。

有关 CloudWatch 支持的统计数据的详细定义，请参阅 [CloudWatch 统计数据定义 \(p. 138\)](#)。

单位

所有统计数据都有度量单位。示例单位包括 `Bytes`、`Seconds`、`Count` 和 `Percent`。有关 CloudWatch 支持的单位的完整列表，请参阅 Amazon CloudWatch API 参考中的 [MetricDatum](#) 数据类型。

您可在创建自定义指标时指定单位。如果未指定单位，CloudWatch 将使用 `None` 作为单位。单位帮助为数据提供概念意义。尽管在内部单位对于 CloudWatch 无重大意义，但其他应用程序还是可以根据单位得到语义信息。

指定度量单位的指标数据点是单独聚合的。当获得统计数据而不指定单位时，CloudWatch 会将相同单位的所有数据点聚合在一起。如果有两个完全相同的指标包含不同的单位，则每个单位会各返回一个数据流。

时间段

时间段是与特定 Amazon CloudWatch 统计数据关联的时间的长度。每项统计信息代表在指定时间段内对收集的指标数据的聚合。时间段以秒为单位定义，时间段的有效值为 1、5、10、30 或 60 的任意倍数。例如，

要指定六分钟的时间段，时间段的值应为 360。通过改变时间段的长度可以调整数据聚合的方式。时间段可短至一秒，也可长至一天 (86400 秒)。默认值为 60 秒。

只有您使用 1 秒的存储精度定义的自定义指标支持亚分钟级时间段。虽然设置为低于 60 的时间段的选项在控制台中始终可用，但您应选择与指标存储方式相符的时间段。有关支持亚分钟级时间段的指标的更多信息，请参阅[高精度指标](#) (p. 147)。

当您检索统计数据时，可指定时间段、开始时间和结束时间。这些参数决定了与统计数据关联的时间的总长度。开始时间和结束时间的默认值将让您获得上一小时的统计数据。您指定的开始时间值和结束时间值将确定 CloudWatch 返回的时间段的数量。例如，使用时间段、开始时间和结束时间的默认值检索统计数据将返回上一小时内每分钟的聚合统计数据集。如果您想要以 10 分钟为一块来聚合统计信息，请指定时间段 600。对于一个完整小时内聚合的统计数据，请指定时间段 3600。

当对某个时段的统计数据进行聚合时，将为聚合的数据标记上对应的时段起始时间。例如，从晚上 7:00 到晚上 8:00 聚合的数据将会加上晚上 7:00 的时间戳。此外，在晚上 7:00 和晚上 8:00 之间聚合的数据将在晚上 7:00 开始可见，然后该聚合数据的值可能会随着 CloudWatch 收集该时间段内的更多样本而发生更改。

时间段对 CloudWatch 告警也很重要。当您创建告警以监控特定指标时，您是在要求 CloudWatch 将该指标与指定的阈值进行比较。您对 CloudWatch 如何进行此比较拥有广泛的控制力。您不仅能够指定进行比较的时间段，还能够指定用于得出结论的评估期间的数目。例如，如果您指定三个评估期间，则 CloudWatch 将比较三个数据点的时间段。CloudWatch 仅告知您最旧的数据点是否超出阈值以及其他数据点是否超出阈值或丢失。

聚合

Amazon CloudWatch 将根据您在检索统计数据时指定的时间段长度聚合统计数据。您可以根据需要发布任意数量包含相同或类似时间戳的数据点。CloudWatch 会根据指定周期长度聚合这些数据点。CloudWatch 不会跨区域自动聚合数据，但您可以使用指标数学来聚合来自不同区域的指标。

您可以为具有相同时间戳、相同命名空间和维度的指标发布数据点。CloudWatch 会返回这些数据点的聚合统计数据。还可以为包含任意时间戳的相同或不同指标发布多个数据点。

对于大型数据集，您可插入称为统计数据集的预先汇总数据集。通过统计数据集，可以让 CloudWatch 为一定数量的数据点提供 Min、Max、Sum 和 SampleCount。当您需要在分钟内多次收集数据时，通常可以使用它。例如，假设您拥有一个关于网页请求延迟的指标。对命中的每一个网页都发布数据毫无意义。建议您收集该网页的所有点击延迟，每分钟汇总一次，然后将统计数据集发送到 CloudWatch。

Amazon CloudWatch 不会区分指标的来源。如果从不同的来源发布包含相同命名空间和维度的一个指标，则 CloudWatch 会将其视为一个单独的指标。这对分布式扩展型系统中的服务指标有其作用。例如，Web 服务器应用程序中的所有主机都可以发布表示其正在处理的请求延迟的相同指标。CloudWatch 会将这些指标视为单一指标，以便您可以获取应用程序中所有请求的最小值、最大值、平均值和总和的统计数据。

百分位数

百分位数指示某个值在数据集中的相对位置。例如，第 95 个百分位数表示 95% 的数据低于此值，5% 的数据高于此值。百分位数可帮助您更好地了解指标数据的分布情况。

百分位数通常用于隔离异常值。在正态分布中，95% 的数据在平均值的两个标准偏差范围内，99.7% 的数据在平均值的三个标准偏差范围内。落在三个标准偏差之外的任何数据通常被认为是异常值，因为它与平均值相差很多。例如，假设您正在监控 EC2 实例的 CPU 使用率，以确保客户有良好的体验。如果您监控平均值，这可以隐藏异常值。如果您监控最大值，单个异常值可能会使结果出现偏差。使用百分位数，您可以监控 CPU 使用率的第 95 个百分位数，以检查负载异常重的实例。

某些 CloudWatch 指标支持百分位数作为统计数据。对于这些指标，您可以使用百分位数监控您的系统和应用程序，就像使用其他 CloudWatch 统计数据（平均值、最小值、最大值和总和）一样。例如，在创建警报时，可以使用百分位数作为统计函数。您可以指定最多十个小数位的百分位数（例如 p95.0123456789）。

百分位数统计数据可用于自定义指标，只要您为自定义指标发布原始、未经汇总的数据点即可。当任何指标值为负数时，百分位数统计数据不可用于指标。

CloudWatch 需要原始数据点来计算百分位数。如果您改用统计数据发布数据，只有满足以下条件之一，才能检索此数据的百分位数统计数据：

- 统计数据集的 SampleCount 值为 1，且最小值、最大值和总和均相等。
- 最小值和最大值相等，总和等于最小值乘以 SampleCount。

以下 Amazon 服务包含支持百分位数统计的指标。

- API Gateway
- Application Load Balancer
- Amazon EC2
- 弹性负载均衡
- Kinesis
- Amazon RDS

CloudWatch 还支持切尾均值和其他性能统计数据，这些统计数据的使用与百分位数类似。有关更多信息，请参阅[CloudWatch 统计数据定义 \(p. 138\)](#)。

告警

您可使用警报代表您自动发起操作。警报在指定的时间段内监控单个指标，并根据指标值相对于阈值的变化情况执行一项或多项指定操作。操作是一个发送到 Amazon SNS 主题或 Auto Scaling 策略的通知。您还可以将警报添加到控制面板。

告警仅为持续状态更改调用操作。CloudWatch 告警将不会调用操作，因为这些操作处于特定状态。该状态必须已改变并在指定的若干个时间段内保持不变。

创建告警时，请选择一个大于或等于指标的精度的告警监控周期。例如，对 Amazon EC2 进行的基本监控每隔 5 分钟提供一次实例指标。为基本监控指标设置警报时，选择的时间段至少应为 300 秒（5 分钟）。对 Amazon EC2 的详细监控以 1 分钟的精度提供实例指标。当为详细监控指标设置警报时，选择的时间段至少为 60 秒（1 分钟）。

如果对高精度指标设置告警，您可以指定 10 秒或 30 秒时间段的高精度告警，也可以设置 60 秒的任意倍数时间段的定期告警。高精度告警的费用较高。有关高精度指标的更多信息，请参阅[发布自定义指标 \(p. 147\)](#)。

有关更多信息，请参阅[使用 Amazon CloudWatch 告警 \(p. 151\)](#)和[从图表上的指标创建告警 \(p. 112\)](#)。

CloudWatch 账单和成本

本部分介绍了 Amazon CloudWatch 功能如何产生成本。它还提供了可以帮助您分析、优化和降低 CloudWatch 成本的方法。在本部分中，我们在描述 CloudWatch 功能时有时会提及定价。有关定价的信息，请参阅[Amazon CloudWatch 定价](#)。

主题

- [使用 Cost Explorer 分析 CloudWatch 成本和使用数据 \(p. 8\)](#)
- [使用 Amazon Cost and Usage Report 和 Athena 分析 CloudWatch 成本和使用数据 \(p. 11\)](#)

- [优化和降低成本的最佳实践 \(p. 13\)](#)

使用 Cost Explorer 分析 CloudWatch 成本和使用数据

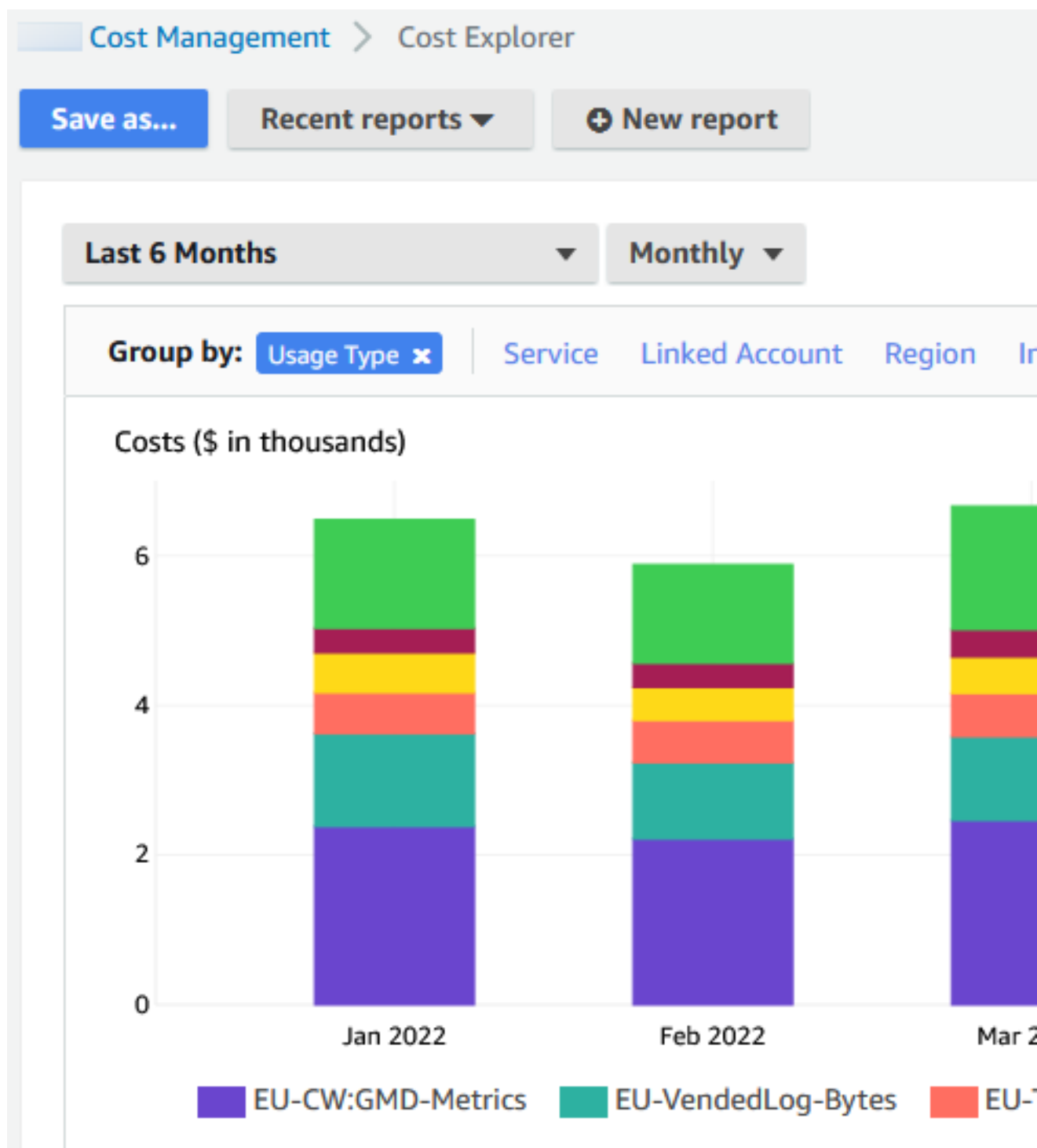
使用 Amazon Cost Explorer，您可以可视化和分析 Amazon Web Services 一段时间内的成本和使用数据，包括 CloudWatch。有关更多信息，请参阅 [Amazon Cost Explorer 入门](#)。

以下程序介绍如何使用 Cost Explorer 可视化和分析 CloudWatch 成本和使用数据。

可视化和分析 CloudWatch 成本和使用数据

1. 在 <https://console.aws.amazon.com/cost-management/home#/custom> 中登录 Cost Explorer 控制台。
2. 在 FILTERS (筛选条件) 下，对于 Service (服务)，选择 CloudWatch。
3. 对于 Group by (分组依据)，选择 Usage Type (使用类型)。您还可以按其他类别对结果进行分组，例如以下类别：
 - API Operation (API 操作) – 查看哪些 API 操作产生的成本最高。
 - Region (区域) – 查看哪些地区产生的成本最高。

下图显示了 CloudWatch 功能在六个月内产生的成本示例。



要了解哪些 CloudWatch 功能产生的费用最高，请查看 UsageType 的值。例如，EU-CW:GMD-Metrics 表示 CloudWatch 批量 API 请求产生的成本。

Note

UsageType 的字符串匹配特定的功能和区域。例如，EU-CW:GMD-Metrics (EU) 的第一部分匹配欧洲 (爱尔兰) 区域，EU-CW:GMD-Metrics (GMD-Metrics) 的第二部分匹配 CloudWatch 批量 API 请求。

UsageType 的整个字符串可以格式化如下：<Region>-CW:<Feature> 或 <Region>-<Feature>。

为了增强可读性，本文档的表格中的 UsageType 字符串已缩短为字符串后缀。例如，EU-CW:GMD-Metrics 缩短为 GMD-Metrics。

下表包含每个 CloudWatch 功能的名称，列出了每个子功能的名称，并列出了 UsageType 的字符串。

CloudWatch 功能	CloudWatch 子功能	UsageType
CloudWatch metrics (CloudWatch 指标)	自定义指标	MetricMonitorUsage
	详细监控	MetricMonitorUsage
	嵌入式指标	MetricMonitorUsage
CloudWatch API 请求	API 请求	Requests
	批量 (获取)	GMD-Metrics
	Contributor Insights	GIRR-Metrics
	位图图像快照	GMWI-Metrics
CloudWatch metric streams (CloudWatch 指标流)	指标流	MetricStreamUsage
CloudWatch dashboards (CloudWatch 控制面板)	控制面板	DashboardsUsageHour
CloudWatch alarms (CloudWatch 告警)	标准	AlarmMonitorUsage
	高分辨率	HighResAlarmMonitorUsage
	复合键	CompositeAlarmMonitorUsage
CloudWatch custom logs (CloudWatch 自定义日志)	收集 (摄取)	DataProcessing-Bytes
	存储 (存档)	TimedStorage-ByteHrs
	分析 (查询)	DataScanned-Bytes
CloudWatch 出售的日志	传输 (Amazon CloudWatch Logs)	VendedLog-Bytes
	传输 (Amazon Simple Storage Service)	S3-Egress-ComprBytes S3-Egress-Bytes
	传输 (Amazon Kinesis Data Firehose)	FH-Egress-Bytes
Contributor Insights	CloudWatch Logs (规则)	ContributorInsightRules

CloudWatch 功能	CloudWatch 子功能	UsageType
	CloudWatch Logs (事件)	ContributorInsightEvents
	Amazon DynamoDB (规则)	ContributorRulesManaged
	DynamoDB (事件)	ContributorEventsManaged
Canaries (Synthetics)	Run	Canary-runs
Evidently	事件	Evidently-event
	分析单位	Evidently-eau
RUM	事件	RUM-event

使用 Amazon Cost and Usage Report 和 Athena 分析 CloudWatch 成本和使用数据

分析 CloudWatch 成本和使用数据的另一种方法是使用 Amazon Cost and Usage Report 和 Amazon Athena。Amazon Cost and Usage Report 包含一组全面的成本和使用数据。您可以创建跟踪费用和使用量的报告，并将这些报告发布到您选择的 S3 存储桶。您还可以从 S3 存储桶中下载和删除报告。有关更多信息，请参阅《Amazon Cost and Usage Report 用户指南》中的[什么是 Amazon Cost and Usage Report ?](#)。

Note

使用 Amazon Cost and Usage Report 无任何费用。您只需在将报告发布到 Amazon Simple Storage Service (Amazon S3) 时支付存储费用。有关更多信息，请参阅《Amazon Cost and Usage Report 用户指南》中的[限额和限制](#)。

Athena 是一项查询服务，您可以将其与 Amazon Cost and Usage Report 结合使用来分析成本和使用数据。您可以在 S3 存储桶中查询报告，而无需先下载它们。有关更多信息，请参阅《Amazon Athena 用户指南》中的[什么是 Amazon Athena ?](#)。有关更多信息，请参阅《Amazon Athena 用户指南》中的[什么是 Amazon Athena ?](#)。有关定价的信息，请参阅[Amazon Athena 定价](#)。

以下程序描述了启用 Amazon Cost and Usage Report 并将该服务与 Athena 集成的过程。该程序包含两个可用于分析 CloudWatch 成本和使用数据的示例查询。

Note

您可以使用本文档中的任何示例查询。本文档中的所有示例查询都对应一个名为 costandusagereport 的数据库，并显示 2022 年 4 月和 2022 年的结果。您可以更改此信息。但是，在运行查询之前，请确保数据库的名称与查询中的数据库名称匹配。

使用 Amazon Cost and Usage Report 和 Athena 分析成本和使用数据

1. 启用 Amazon Cost and Usage Report。有关更多信息，请参阅《Amazon Cost and Usage Report 用户指南》中的[创建成本和使用报告](#)。

Tip

创建报告时，确保选择 Include resource IDs (包含资源 ID)。否则，您的报告将不会包含列 line_item_resource_id。此行可帮助您在分析成本和使用数据时进一步确定成本。

2. 将 Amazon Cost and Usage Report 与 Athena 集成。有关更多信息，请参阅《Amazon Cost and Usage Report 用户指南》中的[使用 Amazon CloudFormation 模板设置 Athena](#)。

3. 查询您的成本和使用情况报告。

Example: Athena query (示例 : Athena 查询)

您可以使用以下查询显示在给定月份中哪些 CloudWatch 功能产生的成本最高。

```
SELECT
CASE
-- Metrics
WHEN line_item_usage_type LIKE '%%MetricMonitorUsage%%' THEN 'Metrics (Custom, Detailed
  monitoring management portal EMF)'
WHEN line_item_usage_type LIKE '%%Requests%%' THEN 'Metrics (API Requests)'
WHEN line_item_usage_type LIKE '%%GMD-Metrics%%' THEN 'Metrics (Bulk API Requests)'
WHEN line_item_usage_type LIKE '%%MetricStreamUsage%%' THEN 'Metric Streams'
-- Dashboard
WHEN line_item_usage_type LIKE '%%DashboardsUsageHour%%' THEN 'Dashboards'
-- Alarms
WHEN line_item_usage_type LIKE '%%AlarmMonitorUsage%%' THEN 'Alarms (Standard)'
WHEN line_item_usage_type LIKE '%%HighResAlarmMonitorUsage%%' THEN 'Alarms (High
  Resolution)'
WHEN line_item_usage_type LIKE '%%CompositeAlarmMonitorUsage%%' THEN 'Alarms (Composite)'
-- Logs
WHEN line_item_usage_type LIKE '%%DataProcessing-Bytes%%' THEN 'Logs (Collect - Data
  Ingestion)'
WHEN line_item_usage_type LIKE '%%TimedStorage-ByteHrs%%' THEN 'Logs (Storage - Archival)'
WHEN line_item_usage_type LIKE '%%DataScanned-Bytes%%' THEN 'Logs (Analyze - Logs Insights
  queries)'
-- Vended Logs
WHEN line_item_usage_type LIKE '%%VendedLog-Bytes%%' THEN 'Vended Logs (Delivered to CW)'
WHEN line_item_usage_type LIKE '%%FH-Egress-Bytes%%' THEN 'Vended Logs (Delivered to
  Kinesis FH)'
WHEN (line_item_usage_type LIKE '%%S3-Egress-Bytes%%') OR (line_item_usage_type LIKE '%%S3-
  ComprBytes%%') THEN 'Vended Logs (Delivered to S3)'
-- Other
WHEN line_item_usage_type LIKE '%%Canary-runs%%' THEN 'Synthetics'
WHEN line_item_usage_type LIKE '%%Evidently%%' THEN 'Evidently'
WHEN line_item_usage_type LIKE '%%RUM-event%%' THEN 'RUM'
ELSE 'Others'
END AS UsageType,
-- REGEXP_EXTRACT(line_item_resource_id, '^(?:.+?:){5}(.)$', 1) as ResourceID,
-- SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_line_item_type NOT IN ('Tax', 'Credit', 'Refund', 'EdpDiscount', 'Fee', 'RIFee')
-- AND line_item_usage_account_id = '123456789012' - If you want to filter on a specific
  account, you can
remove this comment at the beginning of the line and specify an AWS account.
GROUP BY
1
ORDER BY
TotalSpend DESC,
UsageType;
```

Example: Athena query (示例 : Athena 查询)

您可以使用以下查询显示 UsageType 和 Operation 的结果。这向您展示了 CloudWatch 功能是如何产生成本的。结果还显示了 UsageQuantity 和 TotalSpend 的值，以便您可以查看总使用成本。

Tip

有关 UsageType 的更多信息，将以下一行添加到此查询：

line_item_line_line_item_description

此行创建一个名为 Description (描述) 的列。

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_usage_type AS UsageType,
line_item_operation AS Operation,
line_item_resource_id AS ResourceID,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_line_item_type NOT IN ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
line_item_usage_type,
line_item_resource_id,
line_item_operation
```

优化和降低成本的最佳实践

CloudWatch metrics (CloudWatch 指标)

许多 Amazon Web Services，例如 Amazon Elastic Compute Cloud (Amazon EC2)、Amazon S3 和 Amazon Kinesis Data Firehose 等会自动向 CloudWatch 免费发送指标。但是，按以下类别分组的指标可能会产生额外费用：

- Custom metrics, detailed monitoring, and embedded metrics (自定义指标、详细监控和嵌入式指标)
- API 请求
- 指标流

有关更多信息，请参阅[使用 Amazon CloudWatch 指标](#)。

自定义指标、详细监控和嵌入式指标

自定义指标

您可以创建自定义指标以按任何顺序和任何速率组织数据点。

所有自定义指标均按小时按比例分配。它们只有在发送到 CloudWatch 时才会计量。有关指标定价的信息，请参阅[Amazon CloudWatch 定价](#)。

下表列出了 CloudWatch 指标的相关子功能的名称。该表包含 UsageType 和 Operation 的字符串，它可以帮助您分析和确定与指标相关的成本。

Note

要在使用 Athena 查询成本和使用数据时获得下表中列出的指标的更多详细信息，请将 Operation 的字符串与为 line_item_operation 显示的结果匹配。

CloudWatch 子功能	UsageType	Operation	目的
自定义指标	MetricMonitorUsage	MetricStorage	自定义指标
详细监控	MetricMonitorUsage	MetricStorage:AWS/ {Service}	详细监控
嵌入式指标	MetricMonitorUsage	MetricStorage:Amazon Logs-EMF	日志嵌入式指标
日志筛选条件	MetricMonitorUsage	MetricStorage:Amazon CloudWatchLogs	日志组指标筛选条件

详细监控

CloudWatch 具有两种监控类型：

- 基本监控

基本监控是免费的，并自动为支持该功能的所有 Amazon Web Services 启用。

- 详细监控

详细监控会产生成本，并添加不同的增强功能，具体取决于 Amazon Web Service。对于支持详细监控的每个 Amazon Web Service，您可以选择是否为该服务启用它。有关更多信息，请参阅[基本和详细监控](#)。

Note

其他 Amazon Web Services 支持详细监控，可能使用其他名称引用此功能。例如，对 Amazon S3 的详细监控称为 request metrics（请求指标）。

与自定义指标类似，详细监控按小时按比例分配，并且仅在将数据发送到 CloudWatch 时计量。详细监控根据发送到 CloudWatch 的指标数量生成成本。为降低成本，请仅在必要时启用详细监控。有关详细监控定价方式的信息，请参阅[Amazon CloudWatch 定价](#)。

Example: Athena query（示例：Athena 查询）

您可以使用以下查询显示哪些 EC2 实例已启用详细监控。

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_usage_type AS UsageType,
line_item_operation AS Operation,
line_item_resource_id AS ResourceID,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_operation='MetricStorage:AWS/EC2'
AND line_item_line_item_type NOT IN ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
line_item_usage_type,
```

```
line_item_resource_id,
line_item_operation,
line_item_line_item_description
ORDER BY line_item_operation
```

嵌入式指标

借助 CloudWatch 嵌入式指标格式，您可以将应用程序数据作为日志数据提取，以便生成可操作的指标。有关更多信息，请参阅[采用 CloudWatch 嵌入式指标格式摄取高基数日志并生成指标](#)。

嵌入式指标会根据摄取的日志数量、存档的日志数和生成的自定义指标的数量产生成本。

下表列出了 CloudWatch 嵌入式指标格式的相关子功能的名称。该表包含 UsageType 和 Operation 的字符串，它可以帮助您分析和确定成本。

CloudWatch 子功能	UsageType	Operation	目的
自定义指标	MetricMonitorUsage	MetricStorage:AWS/Logs-EMF	日志嵌入式指标
日志摄取	DataProcessing-Bytes	PutLogEvents	将一批日志事件上传到指定的日志组或日志流
日志存档	TimedStorage-ByteHrs	HourlyStorageMetering	在 CloudWatch Logs 中存储每小时日志和每字节日志

要分析成本，请结合使用 Amazon Cost and Usage Report 和 Athena，以便您可以确定哪些指标正在产生成本，并确定成本是如何产生的。

要充分利用 CloudWatch 嵌入式指标格式产生的成本，请避免基于高基数维度创建指标。这样，CloudWatch 就不会为每个唯一的维度组合创建自定义指标。有关更多信息，请参阅[维度](#)。

如果您正在使用 CloudWatch Container Insights 来利用嵌入式指标格式，则可以使用 Amazon Distro for Open Telemetry 作为最大限度地利用指标相关成本的替代方案。使用 Container Insights，您可以从容器化应用程序和微服务中收集、聚合和汇总指标与日志。启用 Container Insights 后，CloudWatch 代理会将您的日志发送到 CloudWatch，这样它就可以使用这些日志生成嵌入式指标。但是，CloudWatch 代理仅向 CloudWatch 发送固定数量的指标，并且您需要为所有可用指标（包括未使用的指标）付费。使用 Amazon Distro for Open Telemetry，您可以配置和自定义将哪些指标和维度发送到 CloudWatch。这有助于您减少 Container Insights 生成的数据量和降低成本。有关更多信息，请参阅[以下资源](#)：

- [使用 Container Insights](#)
- [Amazon Distro for Open Telemetry](#)

API 请求

CloudWatch 具有以下类型的 API 请求：

- API requests (API 请求)
- Bulk (Get) (批量 (获取))
- Contributor Insights
- Bitmap image snapshot (位图图像快照)

API 请求会根据请求类型和请求的指标数量生成成本。

下表列出了 API 请求的类型并包含 UsageType 和 Operation 的字符串，它可以帮助您分析和确定与 API 相关的成本。

API 请求类型	UsageType	Operation	目的
API 请求	Requests	GetMetricStatistics	检索指定指标的统计数据
	Requests	ListMetrics	列出指定的指标
	Requests	PutMetricData	将指标数据点发布到 CloudWatch
	Requests	GetDashboard	显示指定控制面板的详细信息
	Requests	ListDashboards	列出您账户中的控制面板
	Requests	PutDashboard	创建或更新控制面板
	Requests	DeleteDashboards	删除所有指定的控制面板
批量（获取）	GMD-Metrics	GetMetricData	检索 CloudWatch 指标值
Contributor Insights	GIRR-Metrics	GetInsightRuleReport	返回由 Contributor Insights 规则收集的时间序列数据
位图图像快照	GMWI-Metrics	GetMetricWidgetImage	将一个或多个 CloudWatch 指标的图表快照作为位图图像检索

要分析成本，请使用 Cost Explorer，并按 API Operation（API 操作）对结果进行分组。

API 请求的成本各不相同，当您的 API 调用次数超过根据 Amazon 免费套餐限制提供给您的 API 调用次数时，会产生费用。

Note

GetMetricData 和 GetMetricWidgetImage 不包含在 Amazon 免费套餐限制下。有关更多信息，请参阅《Amazon Billing 用户指南》中的[使用 Amazon 免费套餐](#)。

通常会增加成本的 API 请求是 Put 和 Get 请求。

PutMetricData

PutMetricData 每次调用时都会产生成本，并且根据使用案例的不同，可能会产生巨额成本。有关更多信息，请参阅 Amazon CloudWatch API 参考中的[PutMetricData](#)。

要最大限度地利用 PutMetricData 产生的成本，将更多数据批量发送到您的 API 调用中。根据您的使用案例，考虑使用 CloudWatch Logs 或 CloudWatch 嵌入式指标格式来注入指标数据。有关更多信息，请参阅以下资源：

- Amazon CloudWatch Logs 用户指南中的[什么是 Amazon CloudWatch Logs？](#)
- [采用 CloudWatch 嵌入式指标格式摄取高基数日志并生成指标](#)
- [利用 Amazon CloudWatch 嵌入式自定义指标降低成本并关注我们的客户](#)

GetMetricData

GetMetricData 也可能产生巨大的成本。提高成本的常见使用案例涉及第三方监控工具，这些工具会提取数据以生成见解。有关更多信息，请参阅 Amazon CloudWatch API 参考中的 [GetMetricData](#)。

要降低 GetMetricData 产生的成本，请考虑只提取受监控和使用的数据，或者考虑减少提取数据的频率。根据您的使用案例，您可能会考虑使用指标流，而不是 GetMetricData，这样您就可以以更低的成本将数据近乎实时地推送给第三方。有关更多信息，请参阅以下资源：

- [使用指标流](#)
- [CloudWatch 指标流 - 将 Amazon 指标实时发送给合作伙伴和您的应用程序](#)

GetMetricStatistics

根据您的使用案例，您可能会考虑使用 GetMetricStatistics 而不是 GetMetricData。使用 GetMetricData，您可以快速大规模地检索数据。但是，GetMetricStatistics 包含在 Amazon 免费套餐限制下，最多一百万个 API 请求，如果您不需要在每次调用时检索那么多的指标和数据点，这可以帮助您降低成本。有关更多信息，请参阅以下资源：

- Amazon CloudWatch API 参考中的 [GetMetricStatistics](#)
- [我应该使用 GetMetricData 还是 GetMetricStatistics？](#)

Note

外部调用者进行 API 调用。目前，识别这些调用者的唯一方法是向 CloudWatch 团队提出技术支持请求并询问有关他们的信息。有关创建技术支持请求的信息，请参阅[如何从 Amazon 获得技术支持？](#)

CloudWatch metric streams (CloudWatch 指标流)

借助 CloudWatch 指标流，您可以将指标持续发送到 Amazon 目标和第三方服务提供商目标。

指标流根据指标更新的数量产生成本。指标更新始终包括以下统计数据点的值：

- Minimum
- Maximum
- Sample Count
- Sum

有关更多信息，请参阅[可以流式传输的统计数据](#)。

要分析 CloudWatch 指标流产生的成本，请结合使用 Amazon Cost and Usage Report 与 Athena。通过此方式，您可以确定哪些指标正在产生成本，并确定成本是如何产生的。

Example: Athena query (示例：Athena 查询)

您可以使用以下查询通过 Amazon 资源名称 (ARN) 跟踪哪些指标流产生成本。

```
SELECT
  SPLIT_PART(line_item_resource_id,'/',2) AS "Stream Name",
  line_item_resource_id as ARN,
  SUM(CAST(line_item_unblended_cost AS decimal(16,2))) AS TotalSpend
FROM
  costandusagereport
WHERE
```

```
product_product_name = 'AmazonCloudWatch'  
AND year='2022'  
AND month='4'  
AND line_item_line_item_type NOT IN ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')  
-- AND line_item_usage_account_id = '123456789012' - If you want to filter on a specific  
account, you can  
remove this comment at the beginning of the line and specify an AWS account.  
AND line_item_usage_type LIKE '%%MetricStreamUsage%%'  
GROUP BY line_item_resource_id  
ORDER BY TotalSpend DESC
```

要降低 CloudWatch 指标流产生的成本，请仅流式传输能为您带来业务价值的指标。您还可以停止或暂停任何未使用的指标流。

CloudWatch 告警

使用 CloudWatch 告警，您可以创建指标告警和复合告警。

Note

指标和复合告警的成本按小时按比例分配。只有当您的告警存在时，您才会产生告警成本。

Metric alarms (指标告警)

指标告警具有以下分辨率设置：

- Standard (标准) (每 60 秒评估一次)
- High resolution (高分辨率) (每 10 秒评估一次)

创建指标告警时，成本基于告警的分辨率设置和告警引用的指标数。例如，引用单项指标的指标告警，每小时产生一个告警指标成本。有关更多信息，请参阅[使用 Amazon CloudWatch 告警](#)。

如果创建的指标告警包含引用多个指标的指标数学表达式，则在指标数学表达式中引用的每个告警指标都会产生成本。有关如何创建包含指标数学表达式的指标告警的信息，请参阅[根据指标数学表达式创建 CloudWatch 告警](#)。

如果创建异常检测告警，其中告警会分析过去的指标数据以创建预期值模型，则告警中引用的每个警报指标加上两个额外的指标（分别用于异常检测模型创建的上下限指标）都会产生成本。有关如何创建异常检测告警的信息，请参阅[根据异常检测创建 CloudWatch 告警](#)。

Composite alarms (复合告警)

复合告警包含规则表达式，用于指定应如何评估其他告警的状态以确定其自身的状态。无论评估多少其他告警，复合告警每小时都会产生标准成本。在规则表达式中引用的复合告警会产生单独的成本。有关更多信息，请参阅[创建复合告警](#)。

告警使用类型

下表列出了 CloudWatch 告警的相关子功能的名称。该表包含 UsageType 的字符串，它可以帮助您分析和确定与告警相关的成本。

CloudWatch 子功能	UsageType
标准指标告警	AlarmMonitorUsage
高分辨率指标告警	HighResAlarmMonitorUsage
复合告警	CompositeAlarmMonitorUsage

降低告警成本

要优化聚合了四个或更多指标的指标数学告警所产生的成本，您可以在将数据发送到 CloudWatch 之前聚合数据。这样，您可以创建单个指标的告警，而不是为多个指标聚合数据的告警。有关更多信息，请参阅[发布自定义指标](#)。

降低成本的最佳方法是删除所有不必要或未使用的告警。例如，您可以删除评估不再存在的 Amazon 资源发出的指标的告警。

示例：使用 `DescribeAlarms` 检查处于 `INSUFFICIENT_DATA` 状态的告警

如果删除资源，但不删除该资源发出的指标告警，则告警仍然会存在，通常会进入 `INSUFFICIENT_DATA` 状态。要检查是否有处于 `INSUFFICIENT_DATA` 状态的告警，使用以下 Amazon Command Line Interface (Amazon CLI) 命令。

```
$ aws cloudwatch describe-alarms --state-value INSUFFICIENT_DATA
```

其他降低成本的方法如下：

- 确保为正确的指标创建告警。
- 确保在您不工作的区域中未启用任何告警。
- 请记住，尽管复合告警降低了噪音，但也会产生额外的成本。
- 在决定是创建标准告警还是高分辨率告警时，请考虑您的使用案例以及每种告警类型带来的价值。

CloudWatch Logs

Amazon CloudWatch Logs 具有以下日志类型：

- Custom logs (自定义日志) (您为应用程序创建的日志)
- Vended logs (已出售日志) (其他 Amazon Web Services，例如 Amazon Virtual Private Cloud (Amazon VPC) 和 Amazon Route 53，代表您创建的日志)

有关已出售日志的更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[启用特定 Amazon 服务中的日志记录](#)。

自定义日志和已出售日志会根据收集、存储和分析的日志数量产生成本。另外，已出售日志会针对到 Amazon S3 和 Kinesis Data Firehose 的传输产生成本。

下表列出了 CloudWatch Logs 功能的名称和相关子功能的名称。该表包含 UsageType 和 Operation 的字符串，它可以帮助您分析和确定与日志相关的成本。

CloudWatch Logs 功能	CloudWatch Logs 子功能	UsageType	Operation	目的
Custom logs (自定义日志)	收集 (摄取)	DataProcessing-Bytes	PutLogEvents	将一批日志上传到特定的日志流
	存储 (存档)	TimedStorage-ByteHrs	HourlyStorageMetrics	在 CloudWatch Logs 中存储每小时日志和每字节日志
	分析 (Logs Insights 查询)	DataScanned-Bytes	StartQuery	CloudWatch Logs Insights 查询所扫描的日志数据

CloudWatch Logs 功能	CloudWatch Logs 子功能	UsageType	Operation	目的
Vended logs (已出售日志)	传输 (CloudWatch Logs)	VendedLog-Bytes	PutLogEvents	将一批日志上传到特定的日志流
	传输 (Amazon S3)	S3-Egress-ComprBytes S3-Egress-Bytes	LogDelivery	发送出售日志 (CloudWatch、Amazon S3 或 Kinesis Data Firehose)
	传输 (Kinesis Data Firehose)	FH-Egress-Bytes	LogDelivery	发送出售日志 (CloudWatch、Amazon S3 或 Kinesis Data Firehose)

要分析成本，请结合使用 Amazon Cost and Usage Report 和 Athena，以便您可以确定哪些日志正在产生成本，并确定成本是如何产生的。

Example: Athena query (示例：Athena 查询)

您可以使用以下查询按资源 ID 跟踪哪些日志会产生成本。

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_resource_id AS ResourceID,
line_item_usage_type AS UsageType,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_operation IN
('PutLogEvents','HourlyStorageMetering','StartQuery','LogDelivery')
AND line_item_line_item_type NOT IN ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
line_item_usage_type,
line_item_resource_id,
line_item_operation
ORDER BY
TotalSpend DESC
```

要充分利用 CloudWatch Logs 产生的成本，请考虑以下几点：

- 仅记录能为您带来业务价值的事件。这有助于您减少摄取成本。
- 更改日志保留设置，以减少存储成本。有关更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[更改 CloudWatch Logs 中的日志数据留存](#)。
- 运行 CloudWatch Logs Insights 自动保存在历史记录中的查询。这样，您产生的分析成本就会降低。有关更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[查看正在运行的查询或查询历史记录](#)。
- 使用 CloudWatch 代理收集系统日志和应用程序日志并将其发送到 CloudWatch。这样，您就可以仅收集符合条件的日志事件。有关更多信息，请参阅[Amazon CloudWatch 代理增加了对日志筛选表达式的支持](#)。

要降低已出售日志的成本，请考虑您的使用案例，然后确定应将日志发送到 CloudWatch 还是 Amazon S3。有关的更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[发送到 Amazon S3 的日志](#)。

Tip

如果您想使用指标筛选器、订阅筛选器、Logs Insights 和 Contributor Insights，请将已出售日志发送到 CloudWatch。

或者，如果您正在使用 VPC 流日志并将其用于审计和合规性目的，请将已出售日志发送到 Amazon S3。

有关如何跟踪将 VPC 流日志发布到 S3 存储桶时产生的费用的信息，请参阅[使用 Amazon Cost and Usage Report 和成本分配标签了解 Amazon S3 中的 VPC 流日志数据摄取](#)。

有关如何充分利用 CloudWatch Logs 产生的成本的更多信息，请参阅[哪个日志组导致我的 CloudWatch Logs 账单突然增加？](#)。

Amazon CloudWatch 资源

下列相关资源在您使用此服务的过程中会有所帮助。

资源	描述
Amazon CloudWatch 常见问题	“常见问题解答”涵盖了开发人员对此产品提出的一些最热门的问题。
Amazon 开发人员中心	这是一个帮助您入门的资源整合点，您可以在这里找到相关的文档、代码示例、发布说明和其他信息，帮助您通过 Amazon 构建创新的应用程序。
Amazon Web Services Management Console	此控制台让您无需进行编程即可执行 Amazon CloudWatch 和其他各种 Amazon 产品的大部分功能。
Amazon CloudWatch 论坛	由开发人员组成的社群形式的论坛，开发人员可在这里讨论与 Amazon CloudWatch 有关的技术问题。
Amazon Web Services Support	用于创建和管理 Amazon Web Services Support 案例的中心。还包括指向其他有用资源的链接，如论坛、技术常见问题、服务运行状况和 Amazon Trusted Advisor。
Amazon CloudWatch 产品信息	提供了 Amazon CloudWatch 相关信息的主要 Web 页面。
联系我们	用来查询 Amazon 账单、账户、事件、滥用等信息的中心联络点。

开始设置

要使用 Amazon CloudWatch，您需要一个 Amazon 账户。您的 Amazon 账户允许您使用 Amazon EC2 等服务，来生成可在 CloudWatch 控制台（一种基于 Web 的点击式界面）中查看的指标。此外，您可以安装并配置 Amazon 命令行界面 (CLI)。

注册 Amazon Web Services (Amazon)

创建 Amazon 账户时，我们会自动为所有 Amazon 服务注册您的账户。您只需为使用的服务付费。

如果您已有账户，请跳到下一步。如果您还没有账户，请使用以下步骤创建。

如何注册 Amazon 账户

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

在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

登录 Amazon CloudWatch 控制台

登录 Amazon CloudWatch 控制台

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 如果需要，可使用导航栏将该区域更改为 Amazon 资源所在的区域。
3. 即使这是您首次使用 CloudWatch 控制台，Your Metrics（您的指标）也可能已报告指标，因为您使用的 Amazon 产品会免费自动将指标推送到 Amazon CloudWatch。其他服务要求您启用指标。

如果您没有任何警报，则“Your Alarms”区段将有一个“Create Alarm”按钮。

设置 Amazon CLI

您可以使用 Amazon CLI 或 Amazon CloudWatch CLI 来执行 CloudWatch 命令。请注意，Amazon CLI 会替换 CloudWatch CLI，我们仅将新的 CloudWatch 功能包含在此 Amazon CLI 中。

有关如何安装和配置 Amazon CLI 的信息，请参阅 Amazon Command Line Interface 用户指南中的 [使用 Amazon Command Line Interface 进行设置](#)。

有关如何安装和配置 Amazon CloudWatch CLI 的信息，请参阅 Amazon CloudWatch CLI 参考中的 [设置 Command Line Interface](#)。

Amazon CloudWatch 入门

访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。

将显示 CloudWatch 概览主页。

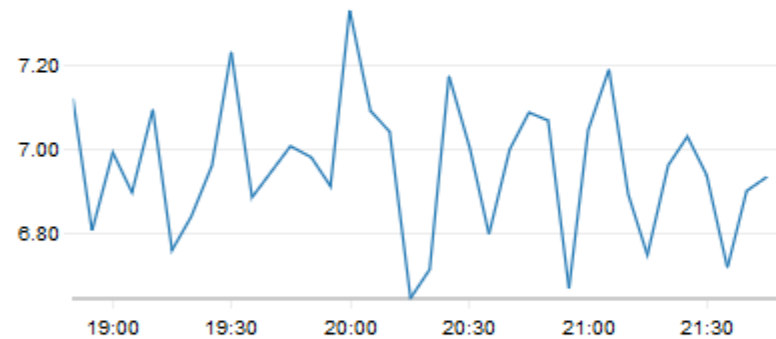
Services

Status	Alarm
 EC2	1
 Lambda	2
 RDS	1
 Kinesis	-
 DynamoDB	-
 API Gateway	-
 Billing	-
 Classic ELB	-
 CloudFront	-
 CloudWatch Events	-

Default dashboard [Edit dashboard](#)

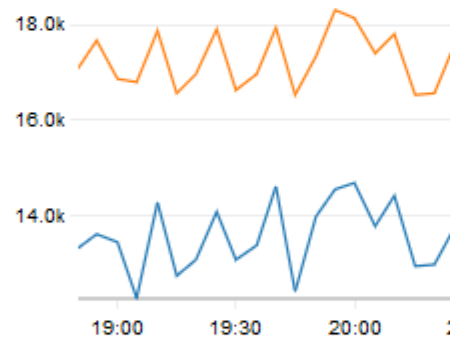
Custom metric 1

Percent



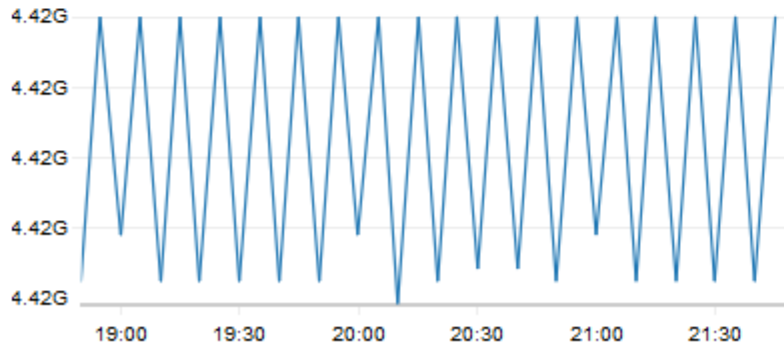
Custom metric 2

Bytes



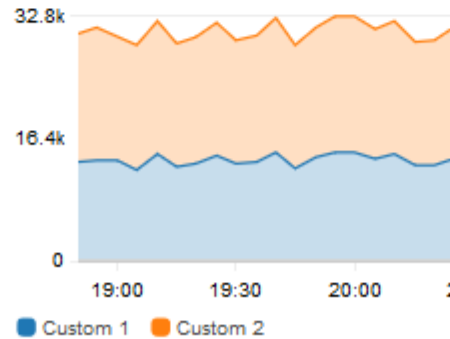
Custom metrics 5

Bytes



Custom metrics 2

Bytes

Cross service dashboard

The cross service dashboard aggregates key metrics from each of the services in your account. [View cross service dashboard](#)

概览显示以下项目且自动刷新。

- 左上角显示在您的账户中使用的 Amazon 服务的列表，以及这些服务中的警报状态。右上角显示您账户中的两个或四个警报，具体取决于您使用的 Amazon 服务的数量。显示的警报是处于 ALARM 状态或最近更改状态的警报。

通过这些上部区域，您可以通过查看每项服务中的警报状态和最新更改了状态的警报，以评估 Amazon 服务的运行状况。这有助于您监控并快速诊断问题。

- 这些区域下方是默认控制面板（如果有）。默认控制面板是您已创建且命名为 CloudWatch-Default 的自定义控制面板。这是一种方便的方式，供您将有关自己的自定义服务或应用程序的指标添加到概览页面，或从您最想监控的 Amazon 服务引入其他关键指标。
- 如果您使用六个或更多 Amazon 服务，默认控制面板下会提供一个链接指向自动跨服务控制面板。跨服务控制面板自动显示您使用的每项 Amazon 服务的关键指标，而不要求您选择要监控哪些指标或创建自定义控制面板。您还可以使用它向下钻取到任何 Amazon 服务并查看该服务的更多关键指标。

如果您使用的 Amazon 服务少于六个，跨服务控制面板会自动显示在此页面上。

在此概览中，您可以将视图专注于特定的资源组或特定的 Amazon 服务。这样，您就可以将视图缩小到您感兴趣的资源子集。使用资源组，您可以使用标签来组织项目，专注于您架构的子集，或只是区分您的生产环境与开发环境。有关更多信息，请参阅[什么是 Amazon Resource Groups？](#)。

主题

- [查看所有 Amazon 服务中的关键指标 \(p. 25\)](#)
- [专注于单个 Amazon 服务中的指标和告警 \(p. 26\)](#)
- [专注于资源组中的指标和告警 \(p. 27\)](#)

查看所有 Amazon 服务中的关键指标

您可以切换到跨服务控制面板屏幕，然后与您正在使用的所有 Amazon 服务的控制面板互动。CloudWatch 控制台按字母顺序显示您的控制面板，并在每个控制面板上显示一两个关键指标。

Note

如果您使用五个或更多 Amazon 服务，CloudWatch 控制台不会在概览屏幕上显示跨服务控制面板。

打开跨服务控制面板

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。

会将您定向到概览屏幕。

2. 在概览屏幕中，选择显示 Overview（概览）的下拉菜单，然后选择 Cross service dashboard（跨服务控制面板）。

会将您定向到跨服务控制面板屏幕。

3. （可选）如果您使用原始界面，请滚动到 Cross-service dashboard（跨服务控制面板）部分，然后选择 View Cross-service dashboard（查看跨服务控制面板）。

会将您定向到跨服务控制面板屏幕。

4. 您可以按两种方式专注于特定服务：

- a. 要查看服务的更多关键指标，请从当前显示 Cross service dashboard（跨服务控制面板）的屏幕顶部的列表中选择其名称。或者，您可以选择服务名称旁边的 View Service dashboard（查看服务控制面板）复选框。

将显示针对该服务的自动控制面板，其中显示该服务的更多指标。此外，对于某些服务，服务控制面板的底部显示与该服务相关的资源。您可以选择该服务控制台中的其中一个资源，并进一步专注于该资源。

- b. 要查看与服务相关的所有警报，请选择屏幕右侧该服务名称旁边的按钮。此按钮上的文本指示您在此服务中已经创建的警报数量，以及任意警报是否处于 ALARM 状态。

显示多个警报时，具有相似设置（如维度、阈值或时间段）的多个警报可能显示在一个图中。

然后，您可以查看有关警报的详细信息和查看警报历史记录。为此，请将鼠标悬停在警报图表上，选择操作图标，然后选择在警报中查看。

警报视图显示在新的浏览器标签中，其中显示您的警报列表以及有关所选警报的详细信息。要查看此警报的历史记录，请选择 History (历史记录) 选项卡。

5. 您可以专注于特定资源组中的资源。为此，请从其中显示所有资源的页面顶部的列表中选择资源组。

有关更多信息，请参阅 [专注于资源组中的指标和告警 \(p. 27\)](#)。

6. 要更改所有图表和当前显示的警报中显示的时间范围，请在屏幕顶部的时间范围旁边选择需要的范围。选择自定义可从默认显示的选项之外的多个时间范围选项进行选择。
7. 警报始终一分钟刷新一次。要刷新该视图，请选择屏幕右上角的刷新图标（双曲线箭头）。要更改屏幕上警报之外的项的自动刷新频率，请选择刷新图标旁边的向下箭头，然后选择需要的刷新率。还可以选择关闭自动刷新。

使服务不出现在跨服务控制面板中

您可以防止服务的指标出现在跨服务控制面板中。这有助于您专注于跨服务控制面板以了解您最需要监控的服务。

如果您从跨服务控制面板上删除某个服务，针对该服务的告警仍然会出现在告警的视图中。

从跨服务控制面板中删除服务的指标

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。

此时将显示主页。

2. 在页面顶部，在概览下选择您要删除的服务。

视图将改为仅显示该服务的指标。

3. 选择 Actions (操作)，然后清除 Show on cross service dashboard (在跨服务控制面板上显示) 旁边的复选框。

专注于单个 Amazon 服务中的指标和告警

在 CloudWatch 主页上，您可以让视图专注于单个 Amazon 服务。您可以通过同时专注于单个 Amazon 服务和资源组来进一步向下钻取。以下过程显示如何仅专注于一个 Amazon 服务。

专注于单个服务

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。

此时将显示主页。

2. 从当前显示概览的屏幕顶部的列表中选择服务名称。

视图将改为显示所选服务的关键指标的图表。

3. 要切换到查看此服务的告警，请选择当前显示的 Service dashboard (服务控制面板) 的屏幕顶部的 Alarms dashboard (告警控制面板) 。
4. 当查看指标时，您可以通过多种方式专注于特定指标：

- a. 要查看任何图表中有关指标的更多详细信息，请将鼠标悬停在图表上，选择操作图标根据指标查看。

该图表显示在新标签中，图表下方列出相关的指标。您可以自定义此图表的视图，同时更改显示的指标和资源、统计数据、时间段以及其他因素，以更好地了解当前情况。

- b. 您可以查看图表中显示的时间范围内的日志事件。这可以帮助您发现您的基础设施中发生的导致意外更改您的指标的事件。

要查看日志事件，请将鼠标悬停在图表上，选择操作图标，然后选择 View in logs (在日志中查看)。

该 CloudWatch Logs 视图会在新选项卡中出现，并显示您的日志组列表。要查看在原始图表中显示的时间范围内发生的其中一个日志组中的日志事件，请选择该日志组。

5. 当查看警报时，您可以通过多种方式专注于特定警报：

- 要查看有关警报的更多详细信息，请将鼠标悬停在警报上，选择操作图标，然后选择在警报中查看。

警报视图显示在新的标签中，其中显示您的警报列表以及有关所选警报的详细信息。要查看此警报的历史记录，请选择 History (历史记录) 选项卡。

6. 警报始终一分钟刷新一次。要刷新该视图，请选择屏幕右上角的刷新图标 (双曲线箭头)。要更改屏幕上除警报之外的项的自动刷新率，请选择刷新图标旁边的向下箭头，然后选择刷新率。还可以选择关闭自动刷新。
7. 要更改所有图表和当前显示的警报中显示的时间范围，请在屏幕顶部的时间范围旁边选择范围。要从默认显示的选项之外的多个时间范围选项进行选择，请选择自定义。
8. 要返回到跨服务控制面板，请在当前显示您专注的服务的屏幕顶部的列表中选择 Overview (概览)。

或者，从任何视图中，您可以在屏幕顶部选择 CloudWatch，以清除所有筛选器并返回到概览页面。

专注于资源组中的指标和告警

您可以将您的视图专注于显示单个资源组中的指标和警报。使用资源组，您可以使用标签来组织项目，专注于您架构的子集，或区分您的生产环境与开发环境。它们还使您能够专注于 CloudWatch 概览上的每个资源组。有关更多信息，请参阅[什么是 Amazon Resource Groups](#)？

当您专注于一个资源组时，显示的内容将更改，而只显示您在其中将资源标记为此资源组一部分的服务。最近的告警区域仅显示与资源组的资源相关联的告警。此外，如果您创建了一个名为 CloudWatch-Default-ResourceGroupName 的控制面板，它将显示在 Default dashboard (默认控制面板) 区域中。

您可以通过同时专注于单个 Amazon 服务和资源组来进一步向下钻取。以下过程显示如何仅专注于一个资源组。

专注于单个资源组

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 从其中显示所有资源的页面顶部，选择一个资源组。
3. 要查看与此资源组相关的更多指标，请在屏幕底部附近选择 View cross service dashboard (查看跨服务控制面板)。

此时将显示跨服务控制面板，同时只显示与此资源组相关的服务。对于每项服务，将显示一个或两个关键指标。

4. 要更改所有图表和当前显示的警报中显示的时间范围，对于屏幕顶部的时间范围，选择一个范围。要从默认显示的选项之外的多个时间范围选项进行选择，请选择自定义。
5. 警报始终一分钟刷新一次。要刷新该视图，请选择屏幕右上角的刷新图标（双曲线箭头）。要更改屏幕上除警报之外的项的自动刷新率，请选择刷新图标旁边的向下箭头，然后选择刷新率。还可以选择关闭自动刷新。
6. 要返回以显示有关您账户中所有资源的信息，请在当前显示资源组名称的屏幕顶部附近，选择所有资源。

使用 Amazon CloudWatch 控制面板

Amazon CloudWatch 控制面板是 CloudWatch 控制台中的可自定义主页，可用于在单个视图中监控资源，即便是分布到不同区域的资源，也能对其进行监控。您可以使用 CloudWatch 控制面板创建 Amazon 资源的指标和告警的自定义视图。

利用控制面板，您可以创建以下各项：

- 所选指标和告警的单一视图，用于帮助您跨一个或多个区域评估资源和应用程序的运行状况。您可以在每个图表上选择用于每个指标的颜色，以便轻松地跨多个图表跟踪同一指标。

您可以创建控制面板以显示来自多个 Amazon 账户和多个区域的图表和其他小组件。有关更多信息，请参阅[跨账户跨区域的 CloudWatch 控制台](#) (p. 654)。

- 一个操作手册，为团队成员提供有关如何对操作事件期间发生的特定事故做出响应的指南。
- 关键资源与应用程序测量的公共视图，团队成员可以共享该视图，以便在操作事件期间加快通信流。

您可以从控制台或使用 Amazon CLI 或 PutDashboard API 操作来创建控制面板。您可以将控制面板添加到收藏夹列表。在收藏夹列表中，您不仅可以访问您收藏的控制面板，还可以访问您最近访问过的控制面板。有关更多信息，请参阅[将控制面板添加到收藏夹列表](#)。

要访问 CloudWatch 控制面板，您需要以下项目之一：

- AdministratorAccess 策略
- CloudWatchFullAccess 策略
- 包含以下一个或多个特定权限的自定义策略：
 - cloudwatch:GetDashboard 和 cloudwatch:ListDashboards 能够查看控制面板
 - cloudwatch:PutDashboard 能够创建或修改控制面板
 - cloudwatch:DeleteDashboards 能够删除控制面板

目录

- [创建 CloudWatch 控制面板](#) (p. 29)
- [跨账户跨区域的控制面板](#) (p. 30)
- [在 CloudWatch 控制面板上创建和使用小组件](#) (p. 33)
- [共享 CloudWatch 控制面板](#) (p. 53)
- [使用实时数据](#) (p. 62)
- [查看动画控制面板](#) (p. 62)
- [将 CloudWatch 控制面板添加到收藏夹列表](#) (p. 63)
- [更改 CloudWatch 控制面板的时间段覆盖设置或刷新闻隔](#) (p. 63)
- [更改 CloudWatch 控制面板的时间范围或时区格式](#) (p. 64)

创建 CloudWatch 控制面板

要开始使用，请创建 CloudWatch 控制面板。您可以创建多个控制面板，而且您可以将它们添加到一个收藏夹列表。您不会受 Amazon Web Services 账户 内的控制面板数量的限制。所有控制面板都适用于全局。它们不是某个区域所特有的。

以下程序说明如何从 CloudWatch 控制台创建控制面板。您可以在命令行界面中使用 PutDashboard API 操作来创建控制面板。API 操作包含对您的控制面板内容进行定义的 JSON 字符串。有关使用

PutDashboard API 操作创建控制面板的更多信息，请参阅 Amazon CloudWatch API 参考中的 [PutDashboard](#)。

Tip

如果使用 PutDashboard API 操作创建新的控制面板，您可以使用来自现有控制面板的 JSON 字符串。

从控制台创建控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择 Create dashboard（创建控制面板）。
3. 在 Create new dashboard（创建新的控制面板）对话框中，输入控制面板的名称，然后选择 Create dashboard（创建控制面板）。

如果使用的名称是 CloudWatch-Default 或 CloudWatch-Default-**ResourceGroupName**，控制面板将在 CloudWatch 概述主页的 Default Dashboard（默认控制面板）下方显示。有关更多信息，请参阅 [Amazon CloudWatch 入门 \(p. 23\)](#)。

4. 在 Add to this dashboard（添加到此控制面板）对话框中执行下列操作之一：
 - 要将图表添加到控制面板，请选择 Line（折线图）或 Stacked area（堆叠面积图），然后选择 Configure（配置）。在 Add metric graph（添加指标图表）对话框中，选择要生成图表的指标，然后选择 Create widget（创建小组件）。如果某指标由于超过 14 天未发布数据而未显示在对话框中，您可以手动添加该指标。有关更多信息，请参阅 [在 CloudWatch 控制面板上手动绘制指标图表 \(p. 35\)](#)。
 - 要将显示指标的数字添加到控制面板中，请选择 Number（数字），然后选择 Configure（配置）。在 Add metric graph（添加指标图表）对话框中，选择要生成图表的指标，然后选择 Create widget（创建小组件）。
 - 要将文本块添加到控制面板中，请选择 Text（文本），然后选择 Configure（配置）。在 New text widget（新的文本小组件）对话框中，对于 Markdown，使用 [Markdown](#) 设置文本格式，然后选择 Create widget（创建小组件）。
5. （可选）选择 Add widget（添加小组件）并重复步骤 4，将另一个小组件添加到控制面板。您可以多次重复此步骤。
6. 选择 Save dashboard。

跨账户跨区域的控制面板

您可以创建跨账户跨区域的控制面板，这些控制面板可将来自多个 Amazon 账户和多个区域的 CloudWatch 数据汇总到一个控制面板中。在此高级控制面板中，您可以了解整个应用程序，并深入了解更多特定的控制面板，而无需登录和注销账户或切换区域。

您可以在 Amazon Web Services Management Console 中和以编程方式创建跨账户跨区域的控制面板。

先决条件

您必须先启用至少一个共享账户和至少一个监控账户，然后才能创建跨账户跨区域的控制面板。此外，要能够使用 CloudWatch 控制台创建跨账户的控制面板，您必须启用控制台以实现跨账户功能。有关更多信息，请参阅 [跨账户跨区域的 CloudWatch 控制台 \(p. 654\)](#)。

利用 Amazon Web Services Management Console 创建和使用跨账户跨区域的控制面板

您可以使用 Amazon Web Services Management Console 创建跨账户跨区域的控制面板。

创建跨账户跨区域的控制面板

1. 登录到监控账户。
2. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
3. 在导航窗格中，选择 Dashboards (控制面板)。
4. 选择一个控制面板，或创建一个新的控制面板。
5. 在屏幕顶部，您可以在账户和区域之间切换。在创建控制面板时，您可以包含来自多个账户和区域的小组件。小组件包括图表、告警和 CloudWatch Logs Insights 小组件。

使用来自不同的账户和区域的指标创建图表

1. 登录到监控账户。
2. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
3. 在导航窗格中，选择 Metrics (指标)，然后选择 All metrics (所有指标)。
4. 选择要从中添加指标的账户和区域。您可以从屏幕右上角附近的账户和区域下拉菜单中选择您的账户和区域。
5. 将所需指标添加到图表中。有关更多信息，请参阅[绘制指标的图表 \(p. 104\)](#)。
6. 重复步骤 4-5 以添加来自其他账户和区域的指标。
7. (可选) 选择 Graphed metrics (绘制的指标) 选项卡，然后添加使用了所选指标的指标数学函数。有关更多信息，请参阅[使用指标数学 \(p. 116\)](#)。

您还可以设置单个图表以包含多个 SEARCH 函数。每项搜索均可引用不同的账户或区域。

8. 在完成图表后，依次选择 Actions (操作) 和 Add to dashboard (添加到控制面板)。

选择您的跨账户控制面板，然后选择 Add to dashboard (添加到控制面板)。

将来自不同账户的警报添加到跨账户的控制面板

1. 登录到监控账户。
2. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
3. 在页面的顶部，选择告警所在的账户。
4. 在导航窗格中，选择 Alarms (告警)。
5. 选中要添加的警报旁边的复选框，然后选择 Add to dashboard (添加到控制面板)。
6. 选择要将警报添加到的跨账户的控制面板，然后选择 Add to dashboard (添加到控制面板)。

以编程方式创建跨账户跨区域的控制面板

您可以使用 Amazon API 和 SDK 以编程方式创建控制面板。有关更多信息，请参阅 [PutDashboard](#)。

为了启用跨账户跨区域的控制面板，我们已向控制面板正文结构添加新参数，如下表和示例中所示。有关总体控制面板正文结构的更多信息，请参阅[控制面板正文结构和语法](#)。

参数	使用	范围	默认值
accountId	指定小组件或指标所在的账户的 ID。	小组件或指标	当前登录的账户
region	指定指标的区域。	小组件或指标	控制台中选定的当前区域

以下示例显示了跨账户跨区域的控制面板中的小组件的 JSON 源。

此示例将 `accountId` 字段设置为小组件级别的共享账户的 ID。这指定了此小组件中的所有指标将来自该共享账户和区域。

```
{
  "widgets": [
    {
      ...
      "properties": {
        "metrics": [
          ...
        ],
        "accountId": "111122223333",
        "region": "us-east-1"
      }
    }
  ]
}
```

此示例在每个指标级别以不同的方式设置 `accountId` 字段。在此示例中，此指标数学表达式中的各个指标来自不同的共享账户和不同的区域。

```
{
  "widgets": [
    {
      ...
      "properties": {
        "metrics": [
          [ { "expression": "SUM(METRICS())", "label": "[avg: ${AVG}] Expression1", "id": "e1", "stat": "Sum" } ],
          [ "AWS/EC2", "CPUUtilization", { "id": "m2", "accountId": "5555666677778888", "region": "us-east-1", "label": "[avg: ${AVG}] ApplicationALabel" } ],
          [ ".", ".", { "id": "m1", "accountId": "9999000011112222", "region": "eu-west-1", "label": "[avg: ${AVG}] ApplicationBLabel" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "stat": "Sum",
        "period": 300,
        "title": "Cross account example"
      }
    }
  ]
}
```

此示例显示了一个警报小组件。

```
{
  "type": "metric",
  "x": 6,
  "y": 0,
  "width": 6,
  "height": 6,
  "properties": {
    "accountId": "111122223333",
    "title": "over50",
    "annotations": {
      "alarms": [
        "arn:aws:cloudwatch:us-east-1:379642911888:alarm:over50"
      ]
    }
  }
}
```

```
    },  
    "view": "timeSeries",  
    "stacked": false  
  }  
}
```

此示例针对的是 CloudWatch Logs Insights 小组件。

```
{  
  "type": "log",  
  "x": 0,  
  "y": 6,  
  "width": 24,  
  "height": 6,  
  "properties": {  
    "query": "SOURCE 'route53test' | fields @timestamp, @message\n| sort @timestamp  
desc\n| limit 20",  
    "accountId": "111122223333",  
    "region": "us-east-1",  
    "stacked": false,  
    "view": "table"  
  }  
}
```

以编程方式创建控制面板的另一种方法是，先在 Amazon Web Services Management Console 中创建一个控制面板，然后复制此控制面板的 JSON 源。为此，请加载该控制面板，然后依次选择 Actions (操作) 和 View/edit source (查看/编辑源)。然后，您可以复制此控制面板 JSON 以将其用作模板，从而创建类似的控制面板。

在 CloudWatch 控制面板上创建和使用小组件

使用本节中的主题，在控制面板中创建和处理图表、告警和文本小组件。

目录

- [在 CloudWatch 控制面板中添加或删除图表 \(p. 33\)](#)
- [在 CloudWatch 控制面板上手动绘制指标图表 \(p. 35\)](#)
- [使用现有图表 \(p. 36\)](#)
- [将 Metrics Explorer 小组件添加到 CloudWatch 控制面板 \(p. 40\)](#)
- [在 CloudWatch 控制面板上添加或删除折线图小组件 \(p. 42\)](#)
- [在 CloudWatch 控制面板上添加或删除数字小组件 \(p. 42\)](#)
- [在 CloudWatch 控制面板上添加或删除量规图小组件 \(p. 43\)](#)
- [将自定义小组件添加到 CloudWatch 控制面板 \(p. 44\)](#)
- [在 CloudWatch 控制面板上添加或删除文本小组件 \(p. 51\)](#)
- [在 CloudWatch 控制面板上添加或删除警报小组件 \(p. 52\)](#)
- [在 CloudWatch 控制面板上链接和取消链接图表 \(p. 53\)](#)

在 CloudWatch 控制面板中添加或删除图表

您可以将包含一个或多个指标的图表添加到您的 CloudWatch 控制面板。可添加到控制面板的图表类型包括：折线图、堆叠面积图、数字、量规图、条形图和饼图。当您不再需要这些图表时，可以将它们从控制面

板中删除。本节内容中的步骤描述如何在您的控制面板中添加和删除图表。如需了解有关如何编辑控制面板上的图表，请参阅[在 CloudWatch 控制面板上编辑图表](#)。

将图表添加到控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 选择 + 符号，然后选择您想要添加到控制面板的图表。
 - 如果您选择 Line（折线图）、Stacked area（堆叠面积图）、Bar（条形图）或 Pie（饼图），请选择 Metrics（指标）。
4. 选择 Browse（浏览），然后选择您想要在控制面板上生成图表的指标。
5. （可选）要更改图表的时间范围，请在图表右上角选择一个预定义的时间范围。这些时间范围从 1 小时到 1 周（1h（1 小时）、3h（3 小时）、12h（12 小时）、1d（1 天）、3d（3 天）或者 1w（1 周））。
 - 要设置自己的时间范围，请选择 Custom（自定义）。
6. （可选）要更改图表的小组件类型，请选择 Options（选项），然后选择新的小组件类型。

Tip

您可以通过选择预定义时间范围旁边的下拉菜单来更改图表的小组件类型。

7. （可选）在 Graphed metrics（绘制的指标）中，您可以为指标添加动态标签，然后更改您的指标的标签、标签颜色、统计数据和时间段。您还可以从左到右决定 Y 轴上的标签的位置。
 - a. 要添加动态标签，请选择 Graphed metrics（绘制的指标），然后选择 Add dynamic labels（添加动态标签）。动态标签显示关于图表图例中的指标的统计数据。每当刷新控制面板或图表时，动态标签都会自动更新。默认情况下，添加到标签的动态值显示在标签开头。有关更多信息，请参阅[使用动态标签 \(p. 107\)](#)。
 - b. 要更改指标的颜色，请选择指标旁边的色块。
 - c. 要更改统计数据，请选择 Statistic（统计数据）下方的下拉菜单，然后选择一个新值。有关更多信息，请参阅[统计数据](#)。
 - d. 要更改时间段，请选择 Period（时间段）列下方的下拉菜单，然后选择一个新值。
8. （可选）要自定义 Y 轴，请选择 Options（选项）。您可以在标签字段的 Left Y-axis（左 Y 轴）下方添加自定义标签。如果图表在 Y 轴的右侧显示值，您也可以自定义该标签。您还可以在 Y 轴上设置最小值和最大值限制，以便图表仅显示您指定的值范围。
9. （可选）要添加或编辑横向注释，请选择 Options（选项）：
 - a. 要添加横向注释，请选择 Add horizontal annotation：
 - b. 对于 Label（标签），选择当前注释旁边的笔和纸图标，然后输入新的注释。
 - c. 对于 Value（值），选择当前值旁边的笔和纸图标，然后输入新值。
 - d. 对于 Fill（填充），选择下拉菜单，然后指定注释将如何使用阴影。您可以选择 None（无）、Above（上方）、Between（中间）或者 Below（下方）。要更改填充颜色，请选择注释旁边的色块。
 - e. 对于 Axis（轴），指定注释显示在 Y 轴的左侧还是右侧。
 - f. 要隐藏注释，请清除要隐藏的注释旁边的复选框。
 - g. 要删除注释，请选择 Actions（操作）下方的 X。

Note

您可以重复这些步骤以向同一个图表添加多个横向注释。

10. （可选）要添加或编辑垂直注释，请选择 Options（选项）：

- a. 要添加垂直注释，请选择 Add vertical annotation (添加垂直注释)。
- b. 对于 Label (标签)，选择当前注释旁边的笔和纸图标，然后输入新的注释。如果您想要仅显示日期和时间，请将标签字段留空。
- c. 对于 Date (日期)，选择当前的日期和时间，然后输入新的日期和时间。
- d. 对于 Fill (填充)，选择下拉菜单，然后指定注释将如何使用阴影。您可以选择 None (无)、Above (上方)、Between (中间) 或者 Below (下方)。要更改填充颜色，请选择注释旁边的色块。
- e. 要隐藏注释，请清除要隐藏的注释旁边的复选框。
- f. 要删除注释，请选择 Actions (操作) 下方的 X。

Note

您可以重复这些步骤以向同一个图表添加多个垂直注释。

11. 选择 Create widget。
12. 选择 Save dashboard。

从控制面板中删除图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 在要删除的图表的右上角，选择 Widget actions (小组件操作)，然后选择 Delete (删除)。
4. 选择 Save dashboard。

在 CloudWatch 控制面板上手动绘制指标图表

如果某个指标在过去 14 天内未发布数据，在搜索要添加到 CloudWatch 控制面板上的图表的指标时，将找不到该指标。执行以下操作步骤手动将指标添加到现有图表：

将在搜索时找不到的指标添加到图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 控制面板必须已包含要添加该指标的图表。如果没有，请创建图表并在其中添加任何指标。有关更多信息，请参阅 [在 CloudWatch 控制面板中添加或删除图表 \(p. 33\)](#)。
4. 依次选择操作、编辑控制面板。

出现一个 JSON 数据块。该数据块指定控制面板中包含的小部件及其内容。下面的示例是一个定义图表的数据块的一部分。

```
{
    "type": "metric",
    "x": 0,
    "y": 0,
    "width": 6,
    "height": 3,
    "properties": {
        "view": "singleValue",
        "metrics": [
            [ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ]
        ],
        "region": "us-west-1"
    }
}
```



```
    },  
  },  
}
```

此示例中，下面的部分定义了显示在此图表中的指标。

```
[ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ]
```

5. 在右方括号后面添加一个逗号（如果没有），然后在逗号后面添加一个带有方括号的类似部分。在该新部分中，指定要添加到图表的指标的命名空间、指标名称以及所需的任何维度。以下是示例。

```
[ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ],  
[ "MyNamespace", "MyMetricName", "DimensionName", "DimensionValue" ]
```

有关指标的 JSON 格式的详细信息，请参阅[指标小部件对象的属性](#)。

6. 选择 Update（更新）。

使用现有图表

按照这些部分中的步骤编辑和修改现有的控制面板图表小部件。

主题

- 在 [CloudWatch 控制面板上编辑图表](#) (p. 36)
- 在 [CloudWatch 控制面板上移动图表或调整图表的大小](#) (p. 40)
- 在 [CloudWatch 控制面板上重命名图表](#) (p. 40)

在 CloudWatch 控制面板上编辑图表

您可以编辑添加到 CloudWatch 控制面板的图表。您可以更改图表的标题、统计数据或时间段。您可以在您的图表中添加、更新和删除指标。如果图表包含不止一个指标，您可以通过隐藏不使用的指标，减少视觉混乱。本节内容中的程序描述如何编辑控制面板上的图表。有关创建图表的信息，请参阅[在 CloudWatch 控制面板中添加或删除图表](#)。

New interface

在控制面板上编辑图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 在要编辑的图表的右上角，选择 Widget actions（小部件操作），然后选择 Edit（编辑）。
4. 要更改图表的标题，请选择当前标题旁边的笔和纸图标。输入新标题，然后选择 Apply（应用）。
5. 要更改图表的时间范围，请在图表右上角选择一个预定义的时间范围。这些时间从 1 小时到 1 周（1h（1 小时）、3h（3 小时）、12h（12 小时）、1d（1 天）、3d（3 天）或者 1w（1 周））。
 - 要设置自己的时间范围，请选择 Custom（自定义）。
6. 要更改图表的小部件类型，请选择 Options（选项）选项卡。您可以选择 Line（折线图）、Stacked area（堆叠面积图）、Number（数字）、Gauge（量规图）、Bar（条形图）、Pie（饼图）。

Tip

您可以通过选择预定义时间范围旁边的下拉菜单来更改图表的小部件类型。

7. 在 Graphed metrics（绘制的指标）中，您可以为指标添加动态标签，然后更改您的指标的标签、标签颜色、统计数据和时间段。您还可以从左到右决定 Y 轴上的标签的位置。

- a. 要为某指标添加动态标签，请选择 Dynamic labels (动态标签)。动态标签显示关于图表图例中的指标的统计数据。每当刷新控制面板或图表时，动态标签都会自动更新。默认情况下，添加到标签的动态值显示在标签开头。有关更多信息，请参阅[使用动态标签 \(p. 107\)](#)。
 - b. 要更改指标的颜色，请选择指标旁边的色块。
 - c. 要更改统计数据，请选择 Statistic (统计数据) 列下的统计值，然后选择一个新值。有关更多信息，请参阅[统计数据 \(p. 5\)](#)。
 - d. 要更改时间段，请选择 Period (时间段) 列下的时间段值，然后选择一个新值。
8. 要添加或编辑横向注释，请选择 Options (选项)：
- a. 要添加横向注释，请选择 Add horizontal annotation：
 - b. 对于 Label (标签)，选择当前注释旁边的笔和纸图标。然后输入新注释。在输入注释后，选择对勾图标。
 - c. 对于 Value (值)，选择当前指标值旁边的笔和纸图标。然后输入新指标值。在输入值后，选择对勾图标。
 - d. 对于 Fill (填充)，选择列下的下拉菜单，然后指定注释将如何使用阴影。您可以选择 None (无)、Above (上方)、Between (中间) 或者 Below (下方)。如果选择 Between (中间)，系统将显示另一个新的标签和值字段。

Tip

您可以通过选择注释旁边的彩色方块来更改填充颜色。

- e. 对于 Axis (轴)，指定注释显示在 Y 轴的左侧还是右侧。
- f. 要隐藏注释，请取消选择要在图表上隐藏的注释旁边的复选框。
- g. 要删除注释，请选择 Actions (操作) 列下的 X。

Note

您可以重复这些步骤以向同一个图表添加多个横向注释。

9. 要添加或编辑垂直注释，请选择 Options (选项)：
- a. 要添加垂直注释，请选择 Add vertical annotation (添加垂直注释)。
 - b. 对于 Label (标签)，选择当前注释旁边的笔和纸图标。然后输入新注释。在输入注释后，选择对勾图标。

Tip

要仅显示日期和时间，请将标签字段留空。

- c. 对于 Date (日期)，选择当前日期和时间。然后，输入新的日期和时间。
- d. 对于 Fill (填充)，选择列下的下拉菜单，然后指定注释将如何使用阴影。您可以选择 None (无)、Above (上方)、Between (中间) 或者 Below (下方)。如果选择 Between (中间)，系统将显示一个新的标签和值字段。

Tip

您可以通过选择注释旁边的彩色方块来更改填充颜色。

Note

您可以重复这些步骤以向同一个图表添加多个垂直注释。

- e. 要隐藏注释，请取消选择要在图表上隐藏的注释旁边的复选框。
 - f. 要删除注释，请选择 Actions (操作) 列下的 X。
10. 要自定义 Y 轴，请选择 Options (选项)。在 Left Y-axis (左 Y 轴) 的下方，您可以为 Label (标签) 中输入自定义标签。如果图表在右 Y 轴上显示值，您也可以自定义该标签。您还可以在 Y 轴上设置最小值和最大值，以便图表仅显示您指定的值范围。
11. 完成更改后，选择 Update widget (更新小组件)。

隐藏或更改图表图例的位置

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 在要编辑的图表的右上角，选择 Widget actions（小组件操作）。选择 Legend（图例），然后选择 Hidden（隐藏）、Bottom（底部）或 Right（右侧）。

为控制面板上的图表暂时隐藏指标

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 为您想要在图表脚注中隐藏的指标选择色块。当您将鼠标悬停在色块上时，色块中会显示 X；而当您选中它时，方块会变成灰色。
4. 要恢复隐藏的指标，请清除灰色方块中的 X。

Original interface

在控制面板上编辑图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 将鼠标悬停在要编辑的图表的右上角。选择 Widget actions（小组件操作），然后选择 Edit（编辑）。
4. 要更改图表的标题，请选择当前标题旁边的铅笔图标，然后输入新的标题。
5. 要更改图表的时间范围，请在图表右上角选择一个预定义的时间范围。这些时间从 1 小时到 1 周（1h（1 小时）、3h（3 小时）、12h（12 小时）、1d（1 天）、3d（3 天）或者 1w（1 周））。
 - 要设置自己的时间范围，请选择 Custom（自定义）。
6. 要更改图表的小组件类型，请选择 Graph options（图表选项）选项卡。您可以选择 Line（折线图）、Stacked area（堆叠面积图）、Number（数字）、Bar（条形图）或 Pie（饼图）。

Tip

您可以通过选择预定义时间范围旁边的下拉菜单来更改图表的小组件类型。

7. 在 Graphed metrics（绘制的指标）中，您可以为指标添加动态标签，然后更改您的指标的标签、标签颜色、统计数据和时间段。您还可以从左到右决定 Y 轴上的标签的位置。
 - a. 要为某指标添加动态标签，请选择 Dynamic labels（动态标签）。动态标签显示关于图表图例中的指标的统计数据。每当刷新控制面板或图表时，动态标签都会自动更新。默认情况下，添加到标签的动态值显示在标签开头。有关更多信息，请参阅[使用动态标签 \(p. 107\)](#)。
 - b. 要更改指标的颜色，请选择指标旁边的色块。
 - c. 要更改统计数据，请选择 Statistic（统计数据）列下的统计值，然后选择一个新值。有关更多信息，请参阅[统计数据 \(p. 5\)](#)。
 - d. 要更改时间段，请选择 Period（时间段）列下的时间段值，然后选择一个新值。
8. 要添加或编辑横向注释，请选择 Graph options：
 - a. 要添加横向注释，请选择 Add horizontal annotation：
 - b. 对于 Label（标签），选择当前注释旁边的铅笔图标。然后输入新注释。在输入注释后，选择对勾图标。
 - c. 对于 Value（值），选择当前指标值旁边的铅笔图标。然后输入新指标值。在输入值后，选择对勾图标。

- d. 对于 Fill (填充) , 选择列下的下拉菜单, 然后指定注释将如何使用阴影。您可以选择 None (无) 、 Above (上方) 、 Between (中间) 或者 Below (下方) 。如果选择 Between (中间) , 系统将显示一个新的标签和值字段。

Tip

您可以通过选择注释旁边的彩色方块来更改填充颜色。

- e. 对于 Axis (轴) , 指定注释显示在 Y 轴的左侧还是右侧。
- f. 要隐藏注释, 请取消选择要在图表上隐藏的注释旁边的复选框。
- g. 要删除注释, 请选择 Actions (操作) 列下的 X。

Note

您可以重复这些步骤以向同一个图表添加多个横向注释。

9. 要添加或编辑垂直注释, 请选择 Graph options (图表选项) :

- a. 要添加垂直注释, 请选择 Add vertical annotation (添加垂直注释) 。
- b. 对于 Label (标签) , 选择当前注释旁边的铅笔图标。然后输入新注释。在输入注释后, 选择对勾图标。

Tip

要仅显示日期和时间, 请将标签字段留空。

- c. 对于 Date (日期) , 选择当前日期和时间旁边的铅笔图标。然后, 输入新的日期和时间。
- d. 对于 Fill (填充) , 选择列下的下拉菜单, 然后指定注释将如何使用阴影。您可以选择 None (无) 、 Above (上方) 、 Between (中间) 或者 Below (下方) 。如果选择 Between (中间) , 系统将显示一个新的标签和值字段。

Tip

您可以通过选择注释旁边的彩色方块来更改填充颜色。

Note

您可以重复这些步骤以向同一个图表添加多个垂直注释。

- e. 要隐藏注释, 请取消选择要在图表上隐藏的注释旁边的复选框。
 - f. 要删除注释, 请选择 Actions (操作) 列下的 X。
10. 要自定义 Y 轴, 请选择 Graph options (图表选项) 。在 Left Y-axis (左 Y 轴) 的下方, 您可以为 Label (标签) 中输入自定义标签。如果图表在右 Y 轴上显示值, 您也可以自定义该标签。您还可以在 Y 轴上设置最小值和最大值, 以便图表仅显示您指定的值范围。
 11. 完成更改后, 选择 Update widget (更新小组件) 。

隐藏或更改图表图例的位置

1. 通过以下网址打开 CloudWatch 控制台: <https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中, 选择 Dashboards (控制面板) , 然后选择一个控制面板。
3. 将鼠标悬停在要编辑的图表的右上角, 然后选择 Widget actions (小组件操作) 。选择 Legend (图例) , 然后选择 Hidden (隐藏) 、 Bottom (底部) 或 Right (右侧) 。

为控制面板上的图表暂时隐藏指标

1. 通过以下网址打开 CloudWatch 控制台: <https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中, 选择 Dashboards (控制面板) , 然后选择一个控制面板。
3. 为您想要在图表脚注中隐藏的指标选择色块。当您鼠标悬停在色块上时, 色块中会显示 X; 而当您选中它时, 方块会变成灰色。

4. 要恢复隐藏的指标，请清除灰色方块中的 X。

在 CloudWatch 控制面板上移动图表或调整图表的大小

您可以在 CloudWatch 控制面板上安排图表和调整图表的大小。

在控制面板上移动图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 将鼠标指针悬停在图表标题的上方，直至选择图标出现。选择图表并将其拖动到控制面板上的新位置。
4. 选择 Save dashboard。

调整图表大小

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 要增大或减小尺寸，请将鼠标指针悬停在图表的上方，然后拖动图表的右下角。到达希望的大小时，停止拖放。
4. 选择 Save dashboard。

暂时放大图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 选择图表。或者，将鼠标指针悬停在图表标题的上方，然后依次选择 Widget actions 和 Enlarge。

在 CloudWatch 控制面板上重命名图表

您可以在控制面板上更改 CloudWatch 分配给图表的默认名称。

在控制面板上重命名图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 将鼠标指针悬停在图表标题上，选择小部件操作，然后选择编辑。
4. 在 Edit graph 屏幕上靠近顶部的部分，选择图表的标题。
5. 对于标题，输入新的名称，然后选择确定（对勾）。在 Edit graph（编辑图表）屏幕的右下角，选择 Update widget（更新小部件）。

将 Metrics Explorer 小组件添加到 CloudWatch 控制面板

Metrics Explorer 小组件包括具有相同标签或共享相同资源属性（例如实例类型）的多个资源的图表。创建或删除匹配的资源时，这些小组件会保持最新状态。将 Metrics Explorer 小组件添加到控制面板可帮助您更有效地对环境进行故障排除。

例如，您可以通过分配代表其环境（例如生产或测试）的标签来监控 EC2 实例队列。然后，您可以使用这些标签来筛选和聚合操作指标，例如 CPUUtilization，以便了解与每个标签关联的 EC2 实例的运行状况和性能。

以下步骤说明了如何使用控制台将 Metrics Explorer 小组件添加到控制面板。您还可以以编程方式使用 Amazon CloudFormation 将其添加。有关更多信息，请参阅 [Metrics Explorer 小组件对象定义和 AWS::CloudWatch::Dashboard](#)。

将 Metrics Explorer 小组件添加到控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)。
3. 选择要向其添加 Metrics Explorer 小组件的控制面板的名称。
4. 选择 + 符号。
5. 选择 Explorer，然后选择 Next (下一步)。

Note

您必须选择加入新的控制面板视图才能添加 Metrics Explorer 小组件。要加入，请在导航窗格中选择 Dashboards (控制面板)，然后在页面顶部的广告条中选择 try out the new interface (试用新界面)。

6. 请执行下列操作之一：
 - 要使用模板，请选择 Pre-filled Explorer widget (预填充的 Explorer 小组件)，然后选择要使用的模板。
 - 要创建自定义可视化，请选择 Empty Explorer widget (空 Explorer 小组件)。
7. 选择 Create (创建)。

如果您使用了模板，则该小组件会显示在控制面板上 (包含选定的指标)。如果您对 Explorer 小组件和控制面板感到满意，请选择 Save dashboard (保存控制面板)。

如果您没有使用模板，请继续执行以下步骤。

8. 在 Explorer 下的新小组件的 Metrics (指标) 框中，从服务中选择一个指标或所有可用指标。

选择指标之后，您可以选择性地重复此步骤添加更多指标。

9. 对于选定的每个指标，CloudWatch 会在指标名称后面显示其将随即使用的统计数据。若要更改此统计数据，请选择统计数据名称，然后选择您想要的统计数据。
10. 在 From (从) 下，选择标签或资源属性来筛选结果。

执行此操作后，您可以选择性地重复此步骤以选择更多标签或资源属性。

如果您选择同一属性的多个值 (如两个 EC2 实例类型)，则 Explorer 将显示与任一选定属性匹配的所有资源。Explorer 将此选择视为 OR 操作。

如果您选择了不同的属性或标签，例如 **Production** 标签和 M5 实例类型，则仅显示与所有这些选择匹配的资源。Explorer 将此选择视为 AND 操作。

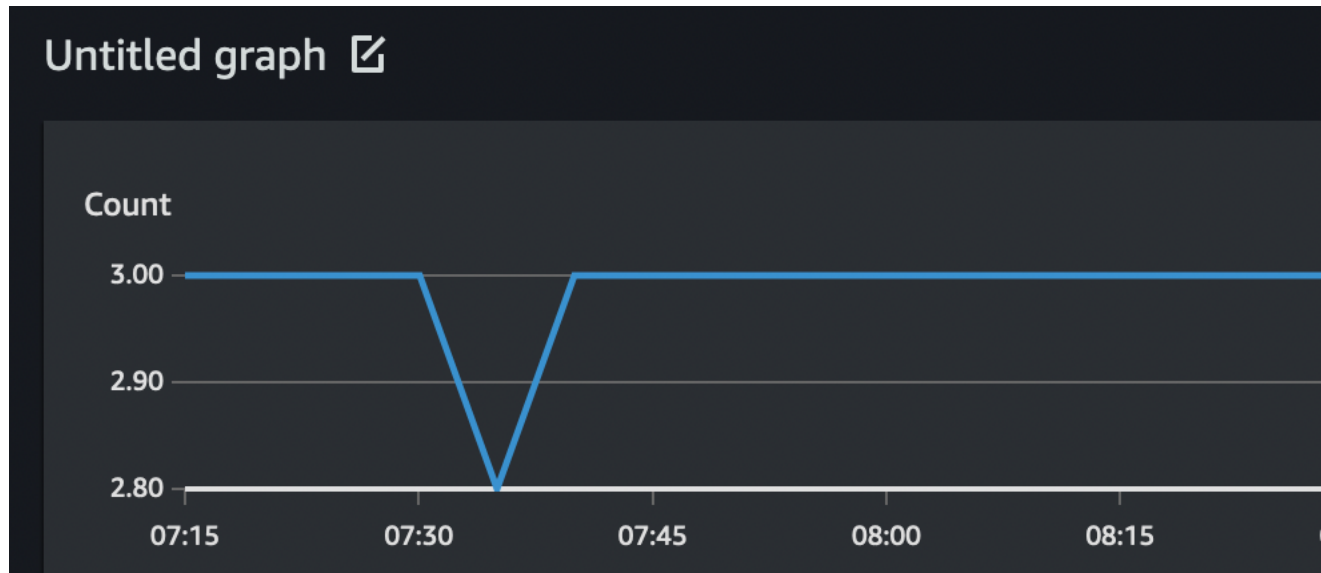
11. (可选) 对于 Aggregate by (聚合依据)，选择要用于聚合指标的统计数据。然后，在 for (聚合方式) 旁边，选择如何聚合列表中的指标。您可以将当前显示的所有资源聚合在一起，也可以按单个标签或资源属性聚合。

根据您选择的聚合方式，结果可能是单个时间序列或多个时间序列。

12. 在 Split by (拆分方式) 下，您可以选择将具有多个时间序列的单个图表拆分为多个图表。可以按您在 Split by (拆分方式) 下选择的多种标准进行拆分。
13. 在 Graph options (绘制选项图表) 下，您可以通过更改周期、图表类型、图例放置和布局来优化图表。
14. 如果您对 Explorer 小组件和控制面板感到满意，请选择 Save dashboard (保存控制面板)。

在 CloudWatch 控制面板上添加或删除折线图小组件

使用折线图小组件，您可以比较一段时间内的指标。您还可以使用小组件的缩微贴图缩放功能来检查折线图的各个部分，而无需在放大和缩小视图之间进行切换。本节内容中的程序描述如何在 CloudWatch 控制面板上添加和删除折线图小组件。有关使用小组件的缩微贴图缩放功能和折线图的信息，请参阅[放大折线图或堆叠面积图](#)。



将折线图小组件添加到控制面板

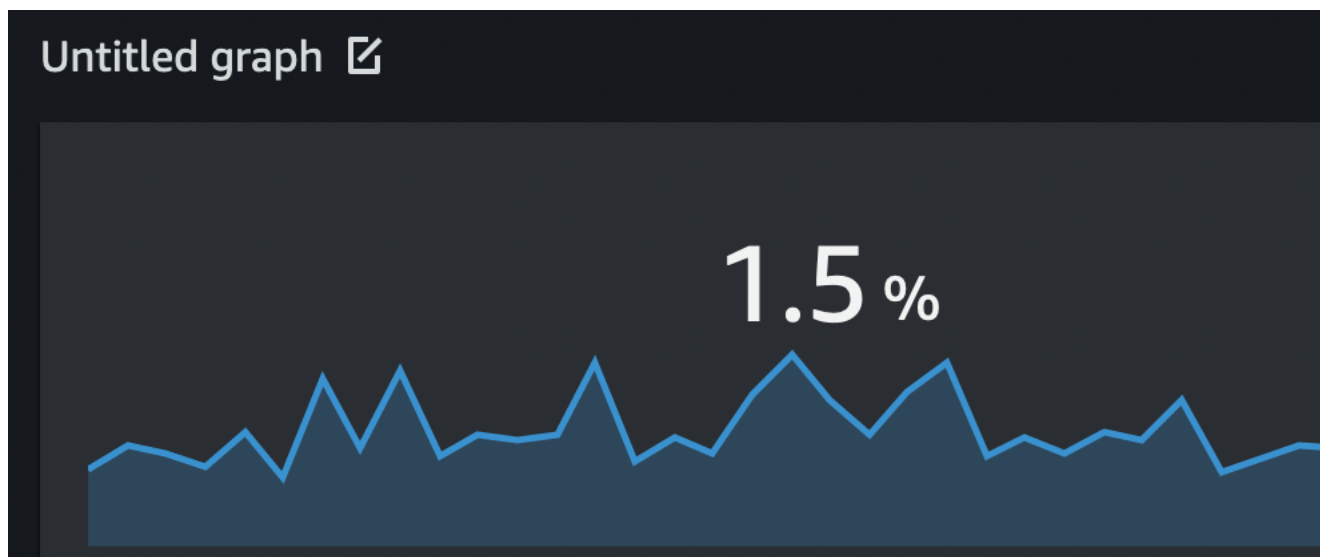
1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 选择 + 符号，然后选择 Line（线条）。
4. 选择 Metrics（指标）。
5. 选择 Browse（浏览），然后选择您想要生成图表的指标。
6. 选择 Create widget（创建小组件），然后选择 Save dashboard（保存控制面板）。

从控制面板中删除折线图小组件

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 在要删除的折线图小组件的右上角，选择 Widget actions（小组件操作），然后选择 Delete（删除）。
4. 选择 Save dashboard。

在 CloudWatch 控制面板上添加或删除数字小组件

使用数字小组件，您可以在最新的指标值和趋势出现后立即查看。由于数字小组件包含走势图功能，因此您可以在单个图表中对指标趋势的上下两部分进行可视化。本节内容中的程序描述如何在 CloudWatch 控制面板中添加和删除数字小组件。



Note

只有新接口支持走势图功能。当您创建数字小组件时，走势图功能会自动包含在内。

将数字小组件添加到控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 选择 + 符号，然后选择 Number（数字）。
4. 选择 Browse（浏览），然后选择您想要生成图表的指标。
5. （可选）要更改走势图功能的颜色，请选择 Graphed metrics（绘制的指标），然后选择指标标签旁边的颜色框。将显示一个菜单，供您选择不同的颜色或输入六位十六进制颜色代码，以指定颜色。
6. （可选）要关闭走势图功能，请选择 Options（选项）。在 Sparkline（走势图）的下方，勾选复选框。
7. 选择 Create widget（创建小组件），然后选择 Save dashboard（保存控制面板）。

Tip

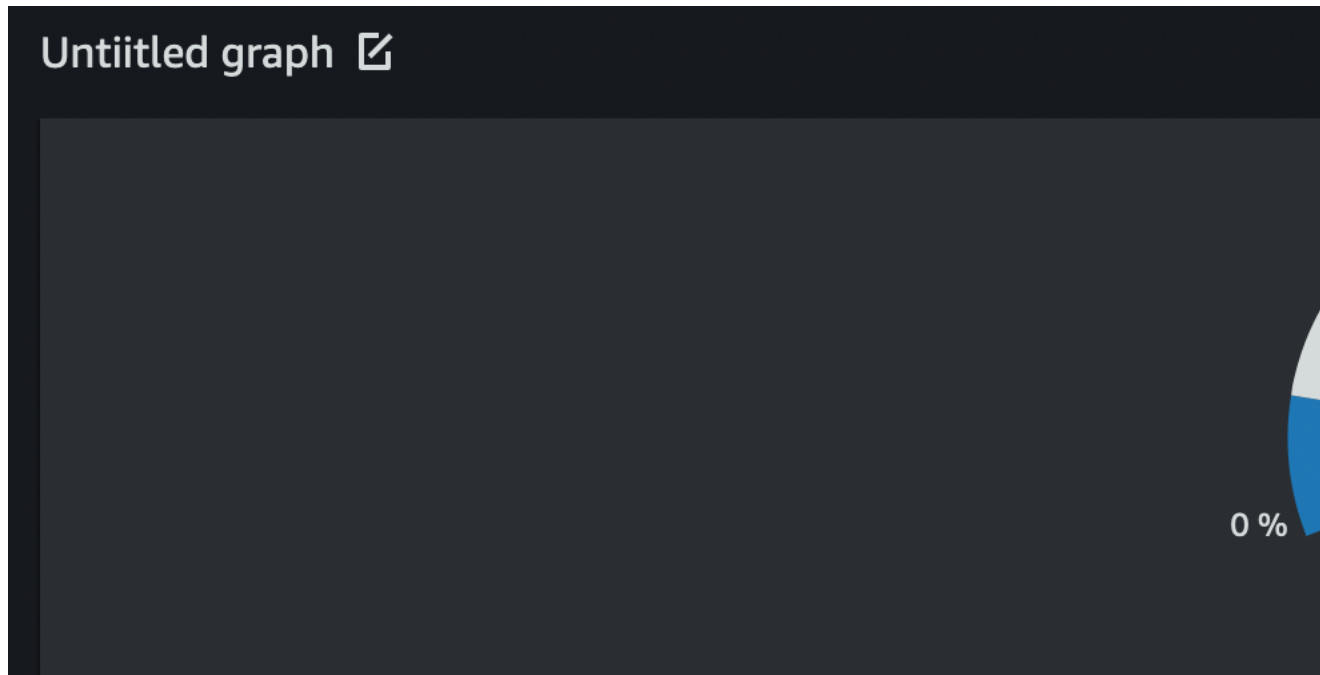
您可以在控制面板屏幕的数字小组件中关闭走势图功能。在要修改的数字小组件的右上角，选择 Widget actions（小组件操作）。选择 Sparkline（走势图），然后选择 Hide sparkline（隐藏走势图）。

从控制面板中删除数字小组件

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择包含您想要删除的数字小组件的控制面板。
3. 在要删除的数字小组件的右上角，选择 Widget actions（小组件操作），然后选择 Delete（删除）。
4. 选择 Save dashboard。

在 CloudWatch 控制面板上添加或删除量规图小组件

使用量规图小组件，您可以对介于范围之间的指标值进行可视化。例如，您可以使用量规图小组件来绘制关于百分比和 CPU 使用率的图表，以便在发生任何性能问题时进行观察和诊断。本节内容中的程序描述如何在 CloudWatch 控制面板中添加和删除量规图小组件。



Note

只有 CloudWatch 控制台中的新接口支持创建量规图小组件。在创建此小组件时，您必须设置一个量规范围。

将量规图小组件添加到控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 在控制面板屏幕中，选择 + 符号，然后选择 Gauge（量规图）。
4. 选择 Browse（浏览），然后选择您想要生成图表的指标。
5. 选择 Options。在 Gauge range（量规范围）下方，设置 Min（最小）和 Max（最大）值。对于百分比，如 CPU 使用率等，我们建议您将值设置为 Min 到 0 和 Max 到 100。
6. （可选）要更改量规图小组件的颜色，请选择 Graphed metrics（绘制的指标），然后选择指标标签旁边的颜色框。将显示一个菜单，供您选择不同的颜色或输入六位十六进制颜色代码，以指定颜色。
7. 选择 Create widget（创建小组件），然后选择 Save dashboard（保存控制面板）。

从控制面板中删除量规图小组件

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择包含您想要删除的量规图小组件的控制面板。
3. 在要删除的量规图小组件的右上角，选择 Widget actions（小组件操作），然后选择 Delete（删除）。
4. 选择 Save dashboard。

将自定义小组件添加到 CloudWatch 控制面板

custom widget（自定义小组件）是一个 CloudWatch 控制面板小组件，它可以使用自定义参数调用任何 Amazon Lambda 函数。然后显示返回的 HTML 或 JSON。自定义小组件是一种在控制面板上构建自定义

数据视图的简单方法。如果您可以编写 Lambda 代码并创建 HTML，则可以创建有用的自定义小组件。此外，Amazon 提供了几个预构建的自定义小组件，您无需任何代码即可创建这些小组件。

当您创建 Lambda 函数以用作自定义小组件时，我们强烈建议您在函数名称中包含前缀 customWidget。这有助于您在将自定义小组件添加到控制面板时，了解哪些 Lambda 函数可以安全使用。

自定义小组件的操作类似于控制面板上的其他小组件。它们可以刷新和自动刷新、调整大小和移动。它们会对控制面板的时间范围做出反应。

如果您已设置 CloudWatch 控制台跨账户功能，则可以将在一个账户中创建的自定义小组件添加到其他账户中的控制面板。有关更多信息，请参阅[跨账户跨区域的 CloudWatch 控制台 \(p. 654\)](#)。

您还可以使用 CloudWatch 控制面板共享功能在自己的网站上使用自定义小组件。有关更多信息，请参阅[共享 CloudWatch 控制面板 \(p. 53\)](#)。

主题

- [有关自定义小组件的详细信息 \(p. 45\)](#)
- [安全与 JavaScript \(p. 46\)](#)
- [自定义小组件中的交互性 \(p. 47\)](#)
- [创建自定义小组件 \(p. 48\)](#)
- [示例自定义小组件 \(p. 49\)](#)

有关自定义小组件的详细信息

自定义小组件的工作方式如下：

1. CloudWatch 控制面板调用包含小组件代码的 Lambda 函数。并传递在小组件中定义的任何自定义参数。
2. Lambda 函数返回 HTML、JSON 或 Markdown 字符串。Markdown 以 JSON 格式返回，格式如下：

```
{"markdown": "markdown content"}
```

3. 控制面板显示返回的 HTML 或 JSON。

如果函数返回 HTML，则支持大多数 HTML 标签。您可以使用 Cascading 样式表 (CSS) 样式和可扩展矢量图形 (SVG) 来构建复杂的视图。

HTML 元素（例如链接和表格）的默认样式遵循 CloudWatch 控制面板的样式。您可以通过使用内联样式和 `<style>` 标签来自定义此样式。您还可以通过包含具有 `cwdb-no-default-styles` 类的单个 HTML 元素，停用默认样式。以下示例会停用默认样式：`<div class="cwdb-no-default-styles"></div>`。

自定义小组件对 Lambda 的每次调用都包含一个具有以下内容的 `widgetContext` 元素，以便为 Lambda 函数开发人员提供有用的上下文信息。

```
{
  "widgetContext": {
    "dashboardName": "Name-of-current-dashboard",
    "widgetId": "widget-16",
    "accountId": "012345678901",
    "locale": "en",
    "timezone": {
      "label": "UTC",
      "offsetISO": "+00:00",
      "offsetInMinutes": 0
    },
    "period": 300,
  }
}
```

```
{
  "isAutoPeriod": true,
  "timeRange": {
    "mode": "relative",
    "start": 1627236199729,
    "end": 1627322599729,
    "relativeStart": 86400012,
    "zoom": {
      "start": 1627276030434,
      "end": 1627282956521
    }
  },
  "theme": "light",
  "linkCharts": true,
  "title": "Tweets for Amazon website problem",
  "forms": {
    "all": {}
  },
  "params": {
    "original": "param-to-widget"
  },
  "width": 588,
  "height": 369
}
```

默认 CSS 样式

自定义小组件提供以下默认 CSS 样式元素：

- 您可以使用 CSS 类 btn 添加按钮。它将锚点 (<a>) 转换为按钮，如以下示例所示：

```
<a class="btn" href=https://amazon.com">Open Amazon</a>
```

- 您可以使用 CSS 类 btn btn-primary 添加主按钮。
- 默认情况下，以下元素采用样式：table (表)、select (选择)、headers (h1, h2, and h3) (标头 (标头 1、标头 2 和标头 3))、preformatted text (pre) (预先格式化的文本 (预先))、input (输入) 和 text area (文本区域)。

使用描述参数

我们强烈建议您在函数中支持 describe (描述) 参数，即使它只返回一个空字符串。如果您不支持它，并在自定义小组件中将其调用，它会像显示文档一样显示小组件内容。

如果您包含 describe (描述) 参数，Lambda 函数以 Markdown 格式返回文档，而不执行任何其他操作。

当您在控制台中创建自定义小组件时，在选择 Lambda 函数后，将出现一个 Get documentation (获取文档) 按钮。如果选择此按钮，则会使用 describe (描述) 参数调用该函数并返回该函数的文档。如果文档在 Markdown 中格式正确，CloudWatch 会解析文档中由 YAML 中的三个反引号字符 (```) 引起的第一个条目。然后，它会自动填充参数中的文档。以下是此类格式正确的文档示例。

```
``` yaml
echo: <h1>Hello world</h1>
```
```

安全与 JavaScript

出于安全原因，返回的 HTML 中不允许使用 JavaScript。删除 JavaScript 可防止出现权限升级问题，即 Lambda 函数的编写者注入的代码可能以比在控制面板上查看小组件的用户更高的权限运行。

如果返回的 HTML 包含任何 JavaScript 代码或其他已知的安全漏洞，则会在将其呈现在控制面板上之前，将其从 HTML 中清除。例如，<iframe> 和 <use> 标签是不允许的，将被删除。

默认情况下，自定义小组件不会在控制面板中运行。相反，如果您信任自定义小组件调用的 Lambda 函数，则必须明确允许其运行。对于单个小组件和整个控制面板，您可以选择允许一次或始终允许。您还可以拒绝对单个小组件和整个控制面板的权限。

自定义小组件中的交互性

尽管不允许使用 JavaScript，但还有其它方法允许与返回的 HTML 进行交互。

- 返回的 HTML 中的任何元素都可以在 <cwdb-action> 标签中使用特殊配置进行标记，该配置可以在弹出窗口中显示信息，在点击时要求确认，并在选择该元素时调用任何 Lambda 函数。例如，您可以定义使用 Lambda 函数调用任何 Amazon API 的按钮。可以将返回的 HTML 设置为替换现有 Lambda 小组件的内容，或在模式中显示。
- 返回的 HTML 可以包含打开新控制台、打开其他客户页面或加载其他控制面板的链接。
- HTML 可以包含元素的 title 属性，如果用户在该元素上悬停，它会提供附加信息。
- 元素可以包含 CSS 选择器，如 :hover，它可以调用动画或其他 CSS 效果。该元素可以包含 CSS 选择器，例如，可以调用动画或其他 CSS 效果。您还可以在页面中显示或隐藏元素。您还可以显示或隐藏页面中的元素。

<cwdb-action> 定义和用法

<cwdb-action> 元素定义了前一个元素的行为。<cwdb-action> 的内容为要显示的 HTML 或要传递给 Lambda 函数的参数的 JSON 数据块。

以下是 <cwdb-action> 元素的示例。

```
<cwdb-action
  action="call|html"
  confirmation="message"
  display="popup|widget"
  endpoint="<lambda ARN>"
  event="click|dblclick|mouseenter">

  html | params in JSON
</cwdb-action>
```

- 操作 – 有效值为 call，会调用 Lambda 函数并且显示包含在 <cwdb-action> 中的任何 HTML 的 html。默认为 html。
- 确认 – 显示在执行操作之前必须确认的确认消息，允许客户取消。
- 显示 – 有效值为 popup 和 widget，会替换小组件本身的内容。默认为 widget。
- 端点 – 要调用的 Lambda 函数的 Amazon Resource Name (ARN)。如果 action 是 call，则这是必需的。
- 事件 – 定义上一个调用操作的元素上的事件。有效值包括 click、dblclick 和 mouseenter。mouseenter 事件仅能与 html 操作结合使用。默认为 click。

示例

以下示例说明了如何使用 <cwdb-action> 标签创建按钮，该按钮可使用 Lambda 函数调用重启 Amazon EC2 实例。此选项会在弹出窗口中显示呼叫成功或失败。

```
<a class="btn">Reboot Instance</a>
<cwdb-action action="call" endpoint="arn:aws:lambda:us-
east-1:123456:function:rebootInstance" display="popup">
```

```
{ "instanceId": "i-342389adbfe" }  
</cwdb-action>
```

下一个示例会在弹出窗口中显示更多信息。

```
<a>Click me for more info in popup</a>  
<cwdb-action display="popup">  
  <h1>Big title</h1>  
  More info about <b>something important</b>.  
</cwdb-action>
```

此示例为 Next (下一步) 按钮, 该按钮使用对 Lambda 函数的调用替换小组件的内容。

```
<a class="btn btn-primary">Next</a>  
<cwdb-action action="call" endpoint="arn:aws:lambda:us-east-1:123456:function:nextPage">  
  { "pageNum": 2 }  
</cwdb-action>
```

创建自定义小组件

要创建自定义小组件, 您可以使用 Amazon 提供的任一示例, 也可创建您自己的示例。Amazon 示例包括 JavaScript 和 Python 示例, 并由 Amazon CloudFormation 堆栈创建。有关示例列表, 请参阅 [示例自定义小组件 \(p. 49\)](#)。

在 CloudWatch 控制面板中创建自定义小组件

1. 通过以下网址打开 CloudWatch 控制台: <https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中, 选择 Dashboards (控制面板), 然后选择一个控制面板。
3. 选择 + 符号。
4. 选择 Custom widget (自定义小组件) 。
5. 使用以下方法之一:
 - 要使用 Amazon 提供的示例自定义小组件, 请执行以下操作:
 - a. 在下拉框中选择示例。

Amazon CloudFormation 控制台在新浏览器中启动。在 Amazon CloudFormation 控制台中, 执行以下操作:
 - b. (可选) 自定义 Amazon CloudFormation 堆栈名称。
 - c. 选择所示例使用的任何参数。
 - d. 选择 I acknowledge that Amazon CloudFormation might create IAM resources (我确认, Amazon CloudFormation 可能创建 IAM 资源), 然后选择 Create stack (创建堆栈) 。
 - 要创建 Amazon 提供的自定义小组件, 请执行以下操作:
 - a. 选择 Next (下一步) 。
 - b. 选择从列表中选择 Lambda 函数, 或输入其 Amazon Resource Name (ARN)。如果从列表中选择它, 还要指定函数所在的区域以及要使用的版本。
 - c. 对于 Parameters (参数), 选择函数所使用的任何参数。
 - d. 输入小组件的标题。
 - e. 对于 Update on (更新), 配置应该更新小组件的时间 (应该再次调用 Lambda 函数的时间)。此选项可以是以下一个或多个项: Refresh (刷新) 以便在控制面板自动刷新时更新它, Resize (调整大小) 以便在调整小组件大小时更新它, 或者 Time Range (时间范围) 以便在调整控制面板的时间范围时更新它 (包括缩放图表时) 。
 - f. 如果对预览满意, 请选择 Create widget (创建小组件) 。

示例自定义小组件

Amazon 在 JavaScript 和 Python 中提供了示例自定义小组件。您可以使用此列表中各个小组件的链接来创建这些示例小组件。您也可以使用 CloudWatch 控制台创建和自定义小组件。此列表中的链接会打开 Amazon CloudFormation 控制台并使用 Amazon CloudFormation 快速创建链接来创建自定义小组件。

您还可以在 [GitHub](#) 上访问自定义小组件示例。

在此列表之后，显示了每种语言的 Echo 小组件的完整示例。

JavaScript

JavaScript 中的示例自定义小组件

- [Echo](#) – 一个基本的回显器，可用于测试 HTML 在自定义小组件中的显示方式，而无需编写新的小组件。
- [Hello world](#) – 一个非常基本的入门小组件。
- [自定义小组件调试程序](#) – 一个调试程序小组件，会显示有关 Lambda 运行时环境的有用信息。
- [查询 CloudWatch Logs Insights](#) – 运行和编辑 CloudWatch Logs Insights 查询。
- [运行 Amazon Athena 查询](#) – 运行和编辑 Athena 查询。
- [调用 Amazon API](#) – 调用任何只读 Amazon API 并以 JSON 格式显示结果。
- [快速 CloudWatch 位图图表](#) – 在服务器端使用渲染 CloudWatch 图表，以便快速显示。
- [CloudWatch 控制面板中的文本小组件](#) – 显示指定 CloudWatch 控制面板中的第一个文本小组件。
- [CloudWatch 指标数据作为表](#) – 在表中显示原始 CloudWatch 指标数据。
- [Amazon EC2 表](#) – 按 CPU 使用率显示排名靠前的 EC2 实例。此小组件还包括一个重启按钮，该按钮默认处于禁用状态。
- [Amazon CodeDeploy 部署](#) – 显示 CodeDeploy 部署。
- [Amazon Cost Explorer 报告](#) – 显示选定时间范围内每项 Amazon 服务的成本报告。
- [显示外部 URL 的内容](#) – 显示可从外部访问的 URL 的内容。
- [显示 Amazon S3 对象](#) – 显示您账户中 Amazon S3 存储桶中的对象。
- [简单的 SVG 饼图](#) – 基于 SVG 的图表小组件示例。

Python

Python 中的示例自定义小组件

- [Echo](#) – 一个基本的回显器，可用于测试 HTML 在自定义小组件中的显示方式，而无需编写新的小组件。
- [Hello world](#) – 一个非常基本的入门小组件。
- [自定义小组件调试程序](#) – 一个调试程序小组件，会显示有关 Lambda 运行时环境的有用信息。
- [调用 Amazon API](#) – 调用任何只读 Amazon API 并以 JSON 格式显示结果。
- [快速 CloudWatch 位图图表](#) – 在服务器端使用渲染 CloudWatch 图表，以便快速显示。
- [通过电子邮件发送控制面板快照](#) – 拍摄当前控制面板的快照，并将其发送给电子邮件收件人。
- [向 Amazon S3 发送控制面板快照](#) – 拍摄当前控制面板的快照，并将其存储在 Amazon S3 中。
- [CloudWatch 控制面板中的文本小组件](#) – 显示指定 CloudWatch 控制面板中的第一个文本小组件。
- [显示外部 URL 的内容](#) – 显示可从外部访问的 URL 的内容。
- [RSS 读取器](#) – 显示 RSS 馈送。
- [显示 Amazon S3 对象](#) – 显示您账户中 Amazon S3 存储桶中的对象。

- [简单的 SVG 饼图](#) – 基于 SVG 的图表小组件示例。

JavaScript 中的 Echo 小组件

以下是 JavaScript 中的 Echo 示例小组件。

```
const DOCS = `
## Echo
A basic echo script. Anything passed in the ````echo```` parameter is returned as the
  content of the custom widget.
### Widget parameters
Param | Description
---|---
**echo** | The content to echo back

### Example parameters
````yaml
echo: <h1>Hello world</h1>
````
`;

exports.handler = async (event) => {
  if (event.describe) {
    return DOCS;
  }

  let widgetContext = JSON.stringify(event.widgetContext, null, 4);
  widgetContext = widgetContext.replace(/</g, '&lt;');
  widgetContext = widgetContext.replace(/>/g, '&gt;');

  return `${event.echo || ''}<pre>${widgetContext}</pre>`;
};
```

Python 中的 Echo 小组件

以下是 Python 中的 Echo 示例小组件。

```
import json

DOCS = """
## Echo
A basic echo script. Anything passed in the ``echo`` parameter is returned as the content
  of the custom widget.
### Widget parameters
Param | Description
---|---
**echo** | The content to echo back

### Example parameters
``yaml
echo: <h1>Hello world</h1>
``"""

def lambda_handler(event, context):
    if 'describe' in event:
        return DOCS

    echo = event.get('echo', '')
    widgetContext = event.get('widgetContext')
    widgetContext = json.dumps(widgetContext, indent=4)
    widgetContext = widgetContext.replace('<', '&lt;')
    widgetContext = widgetContext.replace('>', '&gt;')
```



```
return f'{echo}<pre>{widgetContext}</pre>'
```

Java 中的 Echo 小组件

以下是 Java 中的 Echo 示例小组件。

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class Handler implements RequestHandler<Event, String>{

    static String DOCS = ""
        + "## Echo\n"
        + "A basic echo script. Anything passed in the ``echo`` parameter is returned as the\n"
        + "content of the custom widget.\n"
        + "### Widget parameters\n"
        + "Param | Description\n"
        + "----|---\n"
        + "**echo** | The content to echo back\n\n"
        + "### Example parameters\n"
        + "``yaml\n"
        + "echo: <h1>Hello world</h1>\n"
        + "```\n";

    Gson gson = new GsonBuilder().setPrettyPrinting().create();

    @Override
    public String handleRequest(Event event, Context context) {

        if (event.describe) {
            return DOCS;
        }

        return (event.echo != null ? event.echo : "") + "<pre>" +
            gson.toJson(event.widgetContext) + "</pre>";
    }
}

class Event {

    public boolean describe;
    public String echo;
    public Object widgetContext;

    public Event() {}

    public Event(String echo, boolean describe, Object widgetContext) {
        this.describe = describe;
        this.echo = echo;
        this.widgetContext = widgetContext;
    }
}
```

在 CloudWatch 控制面板上添加或删除文本小组件

文本小部件包含 [Markdown](#) 格式的文本块。您可以在 CloudWatch 控制面板中添加、编辑或删除文本小组件。

将文本小部件添加到控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 选择 + 符号。
4. 选择 Text（文本），然后选择 Configure（配置）。
5. 对于 Markdown，使用 [Markdown](#) 添加文本并设置格式，然后选择 Create widget（创建小部件）。
6. 选择 Save dashboard。

在控制面板上编辑文本小部件

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 将鼠标指针悬停在文本块的右上角，然后选择 Widget actions（小部件操作）。然后选择 Edit（编辑）。
4. 根据需要更新文本，然后选择 Update widget（更新小部件）。
5. 选择 Save dashboard。

从控制面板中删除文本小部件

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 将鼠标指针悬停在文本块的右上角，然后选择 Widget actions（小部件操作）。然后选择 Delete（删除）。
4. 选择 Save dashboard。

在 CloudWatch 控制面板上添加或删除警报小部件

要将警报小部件添加到控制面板，请选择下列选项之一：

- 在小部件中添加单个警报，该小部件会显示警报指标的图表和警报状态。
- 添加一个警报状态小部件，它可以在一个网格中显示多个警报的状态。仅显示警报名称和当前状态，不显示图表。一个警报状态小部件最多可包含 100 个警报。

将单个告警（包括其图表）添加到控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Alarms，选择要添加的警报，然后选择 Add to Dashboard。
3. 选择控制面板，选择小部件类型（Line、Stacked area 或 Number），然后选择 Add to dashboard。
4. 要在控制面板上查看警报，请在导航窗格中选择 Dashboards，然后选择控制面板。
5. （可选）要暂时放大警报图表，请选择该图表。
6. （可选）要更改小部件类型，请将鼠标指针悬停在图表标题上，选择 Widget actions（小部件操作），然后选择 Widget type（小部件类型）。

将告警状态小部件添加到控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。

3. 选择 + 符号。
4. 选择 Alarm status (告警状态)。
5. 选择要向小组件添加的告警旁边的复选框，然后选择 Create widget (创建小组件)。
6. 选择 Add to dashboard (添加到控制面板)。

从控制面板中删除告警小组件

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 在小组件上悬停，选择 Widget actions (小组件操作)，然后选择 Delete (删除)。
4. 选择 Save dashboard。如果在保存更改之前尝试离开控制面板，将提示您保存或放弃更改。

在 CloudWatch 控制面板上链接和取消链接图表

可以在控制面板上将图表链接在一起，这样当您放大或缩小一个图表时，其他图表将同时放大或缩小。可以取消链接图表以仅允许缩放一个图表。

在控制面板上链接图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 选择操作，然后选择链接图表。

在控制面板上取消链接图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 清除操作，然后清除链接图表。

共享 CloudWatch 控制面板

您可以与无法直接访问您 Amazon 账户的人员共享 CloudWatch 控制面板。这使您能够跨团队、与利益攸关方以及企业外部人员共享控制面板。您甚至可以在团队区域的大屏幕上显示控制面板，或将控制面板嵌入到 Wiki 和其他网页中。

Warning

与您共享控制面板的所有人员授予该账户中 [授予与您共享控制面板的人员的权限 \(p. 55\)](#) 列出的权限。如果您公开共享控制面板，则拥有控制面板链接的所有人均具有这些权限。
`cloudwatch:GetMetricData` 和 `ec2:DescribeTags` 权限范围不能缩小到特定指标或 EC2 实例，因此具有控制面板访问权限的人员可以查询所有 CloudWatch 指标以及账户中所有 EC2 实例的名称和标签。

共享控制面板时，您可以通过三种方式指定谁可以查看控制面板：

- 共享单个控制面板并指定可查看控制面板的人员的特定电子邮件地址。这些用户中的每个用户都会创建自己的密码，他们必须输入该密码才能查看控制面板。
- 公开共享单个控制面板，以便拥有该链接的任何人都可以查看控制面板。
- 共享您账户中的所有 CloudWatch 控制面板，并指定第三方单点登录 (SSO) 提供商进行控制面板访问。属于此 SSO 提供商列表成员的所有用户均可访问该账户中的所有控制面板。要启用此功能，请将 SSO 提供

商与 Amazon Cognito 集成。SSO 提供商必须支持安全断言标记语言 (SAML)。有关 Amazon Cognito 的更多信息，请参阅[什么是 Amazon Cognito？](#)

Note

如果您共享的控制面板中有带警报注释的指标小组件，共享该控制面板的人员将不会看到这些小组件。相反，他们会看到一个空白的小组件，以及说明该小组件不可用的文字。当您自行查看控制面板时，依然可以看到带有警报注释的指标小组件。

共享控制面板所需的权限

要能够使用以下任一方法共享控制面板并查看已共享哪些控制面板，您必须登录具有特定权限的 IAM 用户或 IAM 角色。

为了能够共享控制面板，您的 IAM 用户或 IAM 角色必须包含以下策略语句中包含的权限：

```
{
  "Effect": "Allow",
  "Action": [
    "iam:CreateRole",
    "iam:CreatePolicy",
    "iam:AttachRolePolicy",
    "iam:PassRole"
  ],
  "Resource": [
    "arn:aws:iam::*:role/service-role/CWDBSharing*",
    "arn:aws:iam::*:policy/*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "cognito-idp:*",
    "cognito-identity:*"
  ],
  "Resource": [
    "*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "cloudwatch:GetDashboard",
  ],
  "Resource": [
    "*",
    // or the ARNs of dashboards that you want to share
  ]
}
```

为了能够查看已共享的控制面板但无法共享控制面板，您的 IAM 用户或 IAM 角色可以包含类似于以下内容的策略语句：

```
{
  "Effect": "Allow",
  "Action": [
    "cognito-idp:*",
    "cognito-identity:*"
  ],
  "Resource": [
```

```
        "*"
      ],
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:ListDashboards",
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

授予与您共享控制面板的人员的权限

当您共享控制面板时，CloudWatch 会在账户中创建一个 IAM 角色，该角色会向与您共享控制面板的人员授予以下权限：

- `cloudwatch:GetInsightRuleReport`
- `cloudwatch:GetMetricData`
- `cloudwatch:DescribeAlarms`
- `ec2:DescribeTags`

Warning

向您共享控制面板的所有人员授予该账户的这些权限。如果您公开共享控制面板，则拥有控制面板链接的所有人均具有这些权限。

`cloudwatch:GetMetricData` 和 `ec2:DescribeTags` 权限范围不能缩小到特定指标或 EC2 实例，因此具有控制面板访问权限的人员可以查询所有 CloudWatch 指标以及账户中所有 EC2 实例的名称和标签。

共享控制面板时，默认情况下，CloudWatch 创建的权限仅限制对共享控制面板上的告警和 Contributor Insights 规则的访问。如果您向控制面板添加新告警或 Contributor Insights 规则，并希望与您共享控制面板的人员也能查看，则必须更新策略以允许这些资源。

与特定用户共享单个控制面板

使用本节中的步骤，以便与您选择的特定电子邮件地址共享控制面板。

Note

默认情况下，控制面板上的任何 CloudWatch Logs 小组件对您共享控制面板的人员均不可见。有关更多信息，请参阅[允许与您共享的人员查看日志表小组件 \(p. 60\)](#)。

默认情况下，控制面板上的任何复合告警小组件对您共享控制面板的人员均不可见。有关更多信息，请参阅[允许与您共享的人员查看复合告警 \(p. 59\)](#)。

与特定用户共享控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)。
3. 选择控制面板的名称。
4. 依次选择 Actions (操作)、Share dashboard (共享控制面板)。
5. 在 Share your dashboard and require a username and password (共享您的控制面板且需要用户名和密码) 旁边，选择 Start sharing (开始共享)。
6. 在 Add email addresses (添加电子邮件地址) 下，输入要与其共享控制面板的电子邮件地址。

7. 输入所有电子邮件地址后，阅读协议并选择确认框。然后，选择 Preview policy (预览策略)。
8. 确认将要共享的资源是您想要的，然后选择 Confirm and generate shareable link (确认并生成可共享的链接)。
9. 在下一页上，选择 Copy link to clipboard (将链接复制到剪贴板)。然后，您可以将此链接粘贴到电子邮件中并将其发送给受邀用户。他们会自动收到一封带有用户名和临时密码的单独电子邮件，用于连接到控制面板。

公开共享单个控制面板

按照本节中的步骤公开共享控制面板。这十分有利于在团队会议室的大屏幕上显示控制面板或嵌入 Wiki 页面。

Important

公开共享控制面板使得拥有该链接的任何人都可以访问该控制面板，而无需身份验证。仅对不包含敏感信息的面板执行此操作。

Note

默认情况下，控制面板上的任何 CloudWatch Logs 小组件对与您共享控制面板的人员均不可见。有关更多信息，请参阅[允许与您共享的人员查看日志表小组件 \(p. 60\)](#)。
默认情况下，控制面板上的任何复合告警小组件对与您共享控制面板的人员均不可见。有关更多信息，请参阅[允许与您共享的人员查看复合告警 \(p. 59\)](#)。

公开共享控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)。
3. 选择控制面板的名称。
4. 依次选择 Actions (操作)、Share dashboard (共享控制面板)。
5. 在 Share your dashboard publicly (公开共享您的控制面板) 旁，选择 Start sharing (开始共享)。
6. 在文本框中输入 **Confirm**。
7. 阅读协议并选择确认框。然后，选择 Preview policy (预览策略)。
8. 确认将要共享的资源是您想要的，然后选择 Confirm and generate shareable link (确认并生成可共享的链接)。
9. 在下一页上，选择 Copy link to clipboard (将链接复制到剪贴板)。然后，您可以共享此链接。与您共享链接的任何人都可以访问控制面板，而无需提供凭证。

使用 SSO 共享账户中的所有 CloudWatch 控制面板

使用本节中的步骤，通过单点登录 (SSO) 与用户共享您账户中的所有控制面板。

Note

默认情况下，控制面板上的任何 CloudWatch Logs 小组件对与您共享控制面板的人员均不可见。有关更多信息，请参阅[允许与您共享的人员查看日志表小组件 \(p. 60\)](#)。
默认情况下，控制面板上的任何复合告警小组件对与您共享控制面板的人员均不可见。有关更多信息，请参阅[允许与您共享的人员查看复合告警 \(p. 59\)](#)。

与 SSO 提供商列表中的用户共享您的 CloudWatch 控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)。
3. 选择控制面板的名称。

- 依次选择 Actions (操作)、Share dashboard (共享控制面板)。
- 选择 Go to CloudWatch Settings (转到 CloudWatch 设置)。
- 如果您需要的 SSO 提供商未在 Available SSO providers (可用 SSO 提供商) 中列出, 请选择 Manage SSO providers (管理 SSO 提供商) 并按照 为 CloudWatch 控制面板共享设置 SSO (p. 57) 中的说明进行操作。

然后返回到 CloudWatch 控制台并刷新浏览器。您启用的 SSO 提供商现在应显示在列表中。

- 在 Available SSO providers (可用 SSO 提供商) 列表中选择所需的 SSO 提供商。
- 选择保存更改。

为 CloudWatch 控制面板共享设置 SSO

若要通过支持 SAML 的第三方单点登录提供商设置控制面板共享, 请按照下列步骤操作。

Important

我们强烈建议您不要使用非 SAML SSO 提供商共享控制面板。这样操作会导致无意中允许第三方访问您账户的控制面板的风险。

设置 SSO 提供商以启用控制面板共享

- 将 SSO 提供商与 Amazon Cognito 集成。有关更多信息, 请参阅[将第三方 SAML 身份提供商与 Amazon Cognito 用户池集成](#)。
- 从 SSO 提供商下载元数据 XML 文件。
- 通过以下网址打开 CloudWatch 控制台: <https://console.aws.amazon.com/cloudwatch/>。
- 在导航窗格中, 选择 Settings (设置)。
- 在 Dashboard sharing (控制面板共享) 部分中, 选择 Configure (配置)。
- 选择 Manage SSO providers (管理 SSO 提供商)。

此操作会在美国东部 (弗吉尼亚北部) 区域 (us-east-1) 中打开 Amazon Cognito 控制台。如果您没有看到任何 User Pools (用户池), 则 Amazon Cognito 控制台可能已在其他区域中打开。如果是, 请将区域更改为 US East (N. Virginia) us-east-1 (美国东部 (弗吉尼亚北部) us-east-1), 然后继续执行后续步骤。

- 选择 CloudWatchDashboardSharing 池。
- 在导航窗格中, 选择 Identity providers (身份提供程序)。
- 选择 SAML。
- 在 Provider name (提供商名称) 中输入您的 SSO 提供商的名称。
- 选择 Select file (选择文件), 然后选择您在步骤 1 中下载的元数据 XML 文件。
- 选择 Create provider (创建提供商)。

查看已共享的控制面板的数量

您可以使用 CloudWatch 控制台查看当前正在与其他人共享的 CloudWatch 控制面板的数量。

查看正在共享的控制面板的数量

- 通过以下网址打开 CloudWatch 控制台: <https://console.aws.amazon.com/cloudwatch/>。
- 在导航窗格中, 选择 Settings (设置)。
- Dashboard sharing (共享控制面板) 部分会显示共享的控制面板的数量。
- 要查看共享的控制面板, 请选择 Username and password (用户名和密码) 和 Public dashboards (公有控制面板) 下的 **number** dashboards shared (共享控制面板数量)。

查看您的哪些控制面板已共享

您可以使用 CloudWatch 控制台查看您的哪些控制面板当前正在与其他人共享。

查看正在共享哪些控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)。
3. 在控制面板列表中，请参阅 Share (共享) 列。包含此列中图标的控制面板当前正在共享。
4. 要查看与哪些用户共享控制面板，请选择控制面板名称，然后依次选择 Actions (操作)、Share dashboard (共享控制面板)。

Share dashboard (共享控制面板) **dashboard name#####** 页面会显示控制面板的共享方式。如果需要，可以通过选择 Stop sharing (停止共享)，停止共享控制面板。

停止共享一个或多个控制面板

您可以停止共享单个共享控制面板，或一次性停止共享所有共享控制面板。

停止共享单个控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)。
3. 选择共享控制面板的名称。
4. 依次选择 Actions (操作)、Share dashboard (共享控制面板)。
5. 选择 Stop sharing (停止共享)。
6. 在确认框中，选择 Stop sharing (停止共享)。

停止共享所有的共享控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Settings (设置)。
3. 在 Dashboard sharing (控制面板共享) 部分中，选择 Stop sharing all dashboards (停止共享所有控制面板)。
4. 在确认框中，选择 Stop sharing all dashboards (停止共享所有控制面板)。

查看共享控制面板权限并更改权限范围

如果要查看共享控制面板用户的权限，或更改共享控制面板权限的范围，请按照本节中的步骤进行操作。

查看共享控制面板的权限

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)。
3. 选择共享控制面板的名称。
4. 依次选择 Actions (操作)、Share dashboard (共享控制面板)。
5. 在 Resources (资源) 下，选择 IAM Role (IAM 角色)。
6. 在 IAM 控制台中，选择所显示的策略。

7. (可选) 要限制共享控制面板用户可以查看哪些告警, 请选择 Edit policy (编辑策略) 并将 `cloudwatch:DescribeAlarms` 权限从其当前位置移动到一个新 Allow 语句, 该语句仅列出您希望共享控制面板用户查看的告警的 ARN。请参阅以下示例。

```
{
  "Effect": "Allow",
  "Action": "cloudwatch:DescribeAlarms",
  "Resource": [
    "AlarmARN1",
    "AlarmARN2"
  ]
}
```

如果您执行此操作, 请务必从当前策略的如下部分中删除 `cloudwatch:DescribeAlarms` 权限:

```
{
  "Effect": "Allow",
  "Action": [
    "cloudwatch:GetInsightRuleReport",
    "cloudwatch:GetMetricData",
    "cloudwatch:DescribeAlarms",
    "ec2:DescribeTags"
  ],
  "Resource": "*"
}
```

8. (可选) 要限制共享控制面板用户可以查看的 Contributor Insights 规则的范围, 请选择 Edit policy (编辑策略) 将 `cloudwatch:GetInsightRuleReport` 从当前位置移动到一个新 Allow 语句, 该语句仅列出您希望共享控制面板用户查看的 Contributor Insights 规则的 ARN。请参阅以下示例。

```
{
  "Effect": "Allow",
  "Action": "cloudwatch:GetInsightRuleReport",
  "Resource": [
    "PublicContributorInsightsRuleARN1",
    "PublicContributorInsightsRuleARN2"
  ]
}
```

如果您执行此操作, 请务必从当前策略的如下部分中删除 `cloudwatch:GetInsightRuleReport`:

```
{
  "Effect": "Allow",
  "Action": [
    "cloudwatch:GetInsightRuleReport",
    "cloudwatch:GetMetricData",
    "cloudwatch:DescribeAlarms",
    "ec2:DescribeTags"
  ],
  "Resource": "*"
}
```

允许与您共享的人员查看复合告警

在共享控制面板时, 默认情况下, 控制面板上的复合告警小组件对与您共享控制面板的人员不可见。要使复合告警小组件可见, 您需要向控制面板共享策略添加 `DescribeAlarms: *` 权限。该权限如下所示:

```
{
```

```
"Effect": "Allow",
"Action": "cloudwatch:DescribeAlarms",
"Resource": "*"
}
```

Warning

上述策略语句允许访问账户中的所有告警。要缩小 `cloudwatch:DescribeAlarms` 的范围，您必须使用 `Deny` 语句。您可以向策略添加一个 `Deny` 语句，并指定要锁定的告警的 ARN。该拒绝语句应类似于以下内容：

```
{
  "Effect": "Allow",
  "Action": "cloudwatch:DescribeAlarms",
  "Resource": "*"
},
{
  "Effect": "Deny",
  "Action": "cloudwatch:DescribeAlarms",
  "Resource": [
    "SensitiveAlarm1ARN",
    "SensitiveAlarm1ARN"
  ]
}
```

允许与您共享的人员查看日志表小组件

在共享控制面板时，默认情况下，控制面板上的 CloudWatch Logs Insights 小组件对与您共享控制面板的人员不可见。这会影响现存的 CloudWatch Logs Insights 小组件以及在共享该小组件后添加到控制面板的任何小组件。

如果您希望这些人员能够查看 CloudWatch Logs 小组件，则必须向 IAM 角色添加权限以进行控制面板共享。

允许与您共享控制面板的人员查看 CloudWatch Logs 小组件

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板）。
3. 选择共享控制面板的名称。
4. 依次选择 Actions（操作）、Share dashboard（共享控制面板）。
5. 在 Resources（资源）下，选择 IAM Role（IAM 角色）。
6. 在 IAM 控制台中，选择所显示的策略。
7. 选择 Edit policy（编辑策略）并添加以下语句。在新语句中，我们建议您仅指定要共享的日志组的 ARN。请参阅以下示例。

```
{
  "Effect": "Allow",
  "Action": [
    "logs:FilterLogEvents",
    "logs:StartQuery",
    "logs:StopQuery",
    "logs:GetLogRecord"
  ],
  "Resource": [
    "SharedLogGroup1ARN",
    "SharedLogGroup2ARN"
  ]
},
```

8. 选择 Save Changes。

如果您的控制面板共享 IAM policy 已包含带 * 的四个权限作为资源，我们强烈建议您更改策略并仅指定您要共享的日志组的 ARN。例如，如果这些权限的 Resource 部分如下所示：

```
"Resource": "*"
```

更改策略以仅指定要共享的日志组的 ARN，如下例所示：

```
"Resource": [  
  "SharedLogGroup1ARN",  
  "SharedLogGroup2ARN"  
]
```

允许与您共享的人员查看自定义小组件

当您共享控制面板时，默认情况下控制面板上的自定义小组件对您共享控制面板的人员不可见。这既会影响现存的自定义小组件，也会影响共享后添加到控制面板的任何自定义小组件。

如果您希望这些人员能够查看自定义小组件，您必须向 IAM 角色添加权限以进行控制面板共享。

允许与您共享控制面板的人员查看自定义小组件

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板）。
3. 选择共享控制面板的名称。
4. 依次选择 Actions（操作）、Share dashboard（共享控制面板）。
5. 在 Resources（资源）下，选择 IAM Role（IAM 角色）。
6. 在 IAM 控制台中，选择所显示的策略。
7. 选择 Edit policy（编辑策略）并添加以下语句。在新语句中，我们建议您仅指定要共享的 Lambda 函数的 ARN。请参阅以下示例。

```
{  
  "Sid": "Invoke",  
  "Effect": "Allow",  
  "Action": [  
    "lambda:InvokeFunction"  
  ],  
  "Resource": [  
    "LambdaFunction1ARN",  
    "LambdaFunction2ARN"  
  ]  
}
```

8. 选择 Save Changes。

如果您的控制面板共享 IAM policy 已包含带 * 的权限作为资源，我们强烈建议您更改策略并仅指定要共享的 Lambda 函数的 ARN。例如，如果这些权限的 Resource 部分如下所示：

```
"Resource": "*"
```

更改策略以仅指定要共享的自定义小组件的 ARN，如下示例所示：

```
"Resource": [  
  "CustomWidget1ARN",  
  "CustomWidget2ARN"  
]
```

```
"LambdaFunction1ARN",  
"LambdaFunction2ARN"  
]
```

使用实时数据

您可以选择指标小部件是否显示实时数据。实时数据是将在最后一分钟内发布的，尚未完全聚合的数据。

- 如果关闭实时数据，则仅显示过去至少一分钟的聚合期的数据点。例如，如果使用 5 分钟聚合期，12:35 的数据点将从 12:35 聚合到 12:40，并在 12:41 时显示。
- 如果启用实时数据，则在相应聚合间隔内发布任何数据后，立即显示最新数据点。每次刷新显示时，最新数据点都可能会随着该聚合期内的新数据发布而发生变化。如果您使用累积统计数据（例如总和或样本数），则使用实时数据可能会导致图形末尾出现下滑。

您可以选择为整个控制面板启用实时数据，也可以为控制面板上的单个小部件启用实时数据。

选择是否在整个控制面板上使用实时数据

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 要永久打开或关闭控制面板上所有小组件的实时数据，请执行以下操作：
 - a. 选择 Actions（操作）、Settings（设置）、Bulk update live data（批量更新实时数据）。
 - b. 选择 Live Data on（实时数据打开）或 Live Data off（实时数据关闭），然后选择 Set（设置）。
4. 要临时覆盖每个小组件的实时数据设置，请选择 Actions（操作）。然后，在 Overrides（覆盖）下的 Live data（实时数据）旁边，请执行以下任一操作：
 - 选择 On（打开）可临时为所有小部件启用实时数据。
 - 选择 OFF（关闭）可临时关闭所有小组件的实时数据。
 - 选择 Do not override（不覆盖）可保留每个小部件的实时数据设置。

选择是否在单个小部件上使用实时数据

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards（控制面板），然后选择一个控制面板。
3. 选择小部件，然后选择 Actions（操作）、Edit（编辑）。
4. 选择 Graph options（图形选项）选项卡。
5. 选中或清除 Live Data（实时数据）下的复选框。

查看动画控制面板

您可以查看动画控制面板，该控制面板可回放随时间变化而捕获的 CloudWatch 指标数据。这可以帮助您在问题发生后查看趋势、进行演示或分析问题。

控制面板中的动画小部件包括折线小部件、堆叠面积小部件、数字小部件和 Metrics Explorer 小部件。饼图、条形图、文本小部件和日志小部件会显示在控制面板中，而不会以动画形式显示。

查看动画控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。

2. 在导航窗格中，选择 Dashboards (控制面板)。
3. 选择控制面板的名称。
4. 选择 Actions (操作)、Replay dashboard (回放控制面板)。
5. (可选) 默认情况下，启动动画时，动画会显示为滑动窗口。如果您希望动画显示为逐点动画，请在动画暂停时选择放大图标并重置缩放。
6. 要开启动画，请选择“Play (播放)”按钮。您还可以选择后退和前进按钮以移动到其他时间点。
7. (可选) 要更改动画的时间窗口，请选择日历然后选择时间段。
8. 要更改动画的速度，请选择 Auto speed (自动速度)，然后选择新速度。
9. 在完成后，选择 Exit animate (退出动画)。

将 CloudWatch 控制面板添加到收藏夹列表

在 CloudWatch 控制台中，您可以将控制面板、警报和日志组添加到收藏夹列表。您可以从导航窗格访问收藏夹列表。以下程序描述如何将控制面板添加到收藏夹列表。

将控制面板添加到收藏夹列表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)。
3. 在控制面板列表中，选择您想要收藏的控制面板名称旁边的星号。
 - (可选) 您还可以通过在列表中选择控制面板，然后选择控制面板名称旁边的星号，收藏某个控制面板。
4. 要访问收藏夹列表，请在导航窗格中选择 Favorites and recents (收藏夹和最近记录)。菜单中有两个列。一列包含您收藏的控制面板、警报和日志组，另一列包含您最近访问过的控制面板、警报和日志组。

Tip

在 CloudWatch 控制台导航窗格中，您可以通过 Favorites and recents (收藏夹和最近记录) 菜单收藏控制面板，以及警报和日志组。在 Recently visited (最近访问) 列中，将鼠标悬停在您想要收藏的控制面板上，然后选择旁边的星号。

更改 CloudWatch 控制面板的时间段覆盖设置或刷新间隔

您可以指定如何保留或修改添加到此控制面板的图表的时间段设置。

更改时间段覆盖选项

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 选择 Actions。
3. 在 Period override (时间段覆盖) 下，选择以下选项之一：
 - 选择 Auto (自动)，让每个图表上的指标时间段自动适应控制面板的时间范围。
 - 选择 Do not override (不要覆盖) 以确保始终遵循每个图表的时间段设置。
 - 选择其他选项之一将导致添加到控制面板的图表始终适应该所选时间作为其时间段设置。

当关闭控制面板或刷新浏览器后，Period override (时间段覆盖) 始终会还原为 Auto (自动)。无法保存不同的时间段覆盖设置。

您可以更改 CloudWatch 控制面板上的数据刷新频率或者将它设置为自动刷新。

更改控制面板刷新闻隔

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 在刷新选项菜单 (右上角) 上，选择 10 秒、1 分钟、2 分钟、5 分钟或 15 分钟。

自动刷新控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 选择刷新选项，然后选择自动刷新。

更改 CloudWatch 控制面板的时间范围或时区格式

可以更改时间范围以显示几分钟、几小时、几天或几周的控制面板数据。您还可以更改时区格式以按 UTC 或本地时间格式显示控制面板数据。本地时间按计算机操作系统中指定的时区显示。

Note

如果您创建的控制面板中的图表包含了 100 个或更多的高精度指标，我们建议您不要设置超过 1 小时的时间范围。有关更多信息，请参阅[高精度指标 \(p. 147\)](#)。

New console

更改控制面板时间范围

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 在控制面板屏幕中，执行以下任一操作：
 - 在右上角，选择一个预定义的时间范围。这些时间从 1 小时到 1 周 (1h (1 小时)、3h (3 小时)、12h (12 小时)、1d (1 天) 或者 1w (1 周))。
 - 或者，您可以选择以下自定义时间范围选项之一：
 - 选择 Custom (自定义)，然后选择 Relative (相对) 选项卡。选择 1 分钟到 15 个月的时间范围。
 - 选择 Custom (自定义)，然后选择 Absolute (绝对) 选项卡。使用日历或文本字段指定时间范围。

Tip

如果在聚合期设置为 Auto (自动) 时更改图表的时间范围，则 CloudWatch 可能会更改时间段。要手动设置时间段，请选择 Actions (操作) 下拉菜单，然后选择 Period override (时间段覆盖)。

更改控制面板时区格式

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。

2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 在控制面板屏幕的右上角，选择 Custom (自定义)。
4. 在显示的框的右上角，从下拉菜单中选择 UTC (协调世界时) 或 Local time (本地时间)。
5. 选择 Apply (应用)。

Old console

更改控制面板时间范围

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 在控制面板屏幕中，执行以下任一操作：
 - 在右上角，选择一个预定义的时间范围。这些时间从 1 小时到 1 周 (1h (1 小时)、3h (3 小时)、12h (12 小时)、1d (1 天)、3d (3 天) 或者 1w (1 周))。
 - 或者，您可以选择以下自定义时间范围选项之一：
 - 选择 custom (自定义) 下拉菜单，然后选择 Relative (相对) 选项卡。选择某个预定义的范围，其跨度从 1 分钟到 15 个月。
 - 选择 custom (自定义) 下拉菜单，然后选择 Absolute (绝对) 选项卡。使用日历或文本字段指定时间范围。

Tip

如果在聚合期设置为 Auto (自动) 时更改图表的时间范围，则 CloudWatch 可能会更改时间段。要手动设置时间段，请选择 Actions (操作) 下拉菜单，然后选择 Period override (时间段覆盖)。

更改控制面板时区格式

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards (控制面板)，然后选择一个控制面板。
3. 在控制面板屏幕的右上角，选择 Custom (自定义) 下拉菜单。
4. 在显示的框的右上角，从下拉菜单中选择 UTC (协调世界时) 或 Local timezone (本地时区)。

使用 Amazon CloudWatch 指标

指标是关于您的系统性能的数据。预设情况下，许多服务都提供资源（例如，Amazon EC2 实例、Amazon EBS 卷和 Amazon RDS 数据库实例）的免费指标。您也可以对某些资源（例如 Amazon EC2 实例）启用详细监控，或发布您自己的应用程序指标。Amazon CloudWatch 可以加载您账户中的所有指标（您提供的 Amazon 资源指标和应用程序指标）以进行搜索、绘制图表及设置告警。

指标数据的保留期限为 15 个月，这使您能够查看最新数据和历史数据。

要在控制台中绘制指标图表，您可以使用 CloudWatch Metrics Insights，这是一个高性能的 SQL 查询引擎，可用于实时识别所有指标中的趋势和模式。

目录

- [基本监控和详细监控 \(p. 66\)](#)
- [使用 CloudWatch Metrics Insights 查询您的指标 \(p. 67\)](#)
- [使用 Metrics Explorer 按标签和属性监控资源 \(p. 82\)](#)
- [使用指标流 \(p. 83\)](#)
- [查看可用指标 \(p. 100\)](#)
- [绘制指标的图表 \(p. 104\)](#)
- [使用 CloudWatch 异常检测 \(p. 113\)](#)
- [使用指标数学 \(p. 116\)](#)
- [在图表中使用搜索表达式 \(p. 131\)](#)
- [获取指标的统计数据 \(p. 138\)](#)
- [发布自定义指标 \(p. 147\)](#)

基本监控和详细监控

CloudWatch 提供两类监控：基本监控和详细监控。

很多 Amazon 服务通过向 CloudWatch 发布一组默认指标来提供基本监控，而不会为客户产生费用。默认情况下，当您开始使用其中某一 Amazon Web Services 时，将自动启用基本监控。有关提供基本监控的服务的列表，请参阅 [发布 CloudWatch 指标的 Amazon 服务 \(p. 768\)](#)。

仅有一些服务提供详细监控。它还会产生费用。要将其用于 Amazon 服务，您必须选择激活它。有关定价的更多信息，请参见 [Amazon CloudWatch 定价](#)。

详细监控选项因提供这种监控的服务而异。例如，Amazon EC2 详细监控提供了更频繁的指标，每隔一分钟发布一次，而不是 Amazon EC2 基本监控中使用的五分钟间隔。针对 Amazon S3 和 Amazon Managed Streaming for Apache Kafka 进行详细监控，意味着更精细的指标。金丝雀金丝雀

在不同 Amazon 服务中，详细监控也有不同名称。例如，在 Amazon EC2 中，它被称为详细监控；在 Amazon Elastic Beanstalk 中，它被称为增强监控；而在 Amazon S3 中，它被称为请求指标。

针对 Amazon EC2 使用详细监控，可帮助您更好地管理 Amazon EC2 资源，以便您可以找到趋势并更快地采取措施。对于 Amazon S3，请求指标每隔一分钟可以使用一次，可帮助您快速识别运行问题，并针对其采取措施。在 Amazon MSK 上，当您启用 PER_BROKER、PER_TOPIC_PER_BROKER 或 PER_TOPIC_PER_PARTITION 级别监控时，您将获得可以提供更多可见性的额外指标。

下表列出了提供详细监控的服务。它还包括指向这些服务的文档的链接，这些文档说明了关于详细监控的更多信息，并提供了如何激活该功能的说明。

服务	文档	
Amazon API Gateway	API Gateway 指标的维度	
Amazon CloudFront	查看其他 CloudFront 分配指标	
Amazon EC2	对实例启用或禁用详细监控	
Elastic Beanstalk	增强型运行状况报告和监控	
Amazon Kinesis Data Streams	增强的分片级指标	
Amazon MSK	用于使用 CloudWatch 进行监控的 Amazon MSK 指标	
Amazon S3	CloudWatch 中的 Amazon S3 请求指标	

使用 CloudWatch Metrics Insights 查询您的指标

CloudWatch Metrics Insights 是一个功能强大的高性能 SQL 查询引擎，您可以使用它来大规模查询指标。您可以实时识别所有 CloudWatch 指标中的趋势和模式。

您可以使用 CloudWatch 控制台执行 Metrics Insights 查询，也可通过 Amazon CLI 或 Amazon SDK 来使用 `GetMetricData` 或 `PutDashboard` 进行查询。在控制台中运行的查询免费。有关 CloudWatch 定价的信息，请参阅 [Amazon CloudWatch 定价](#)。

使用 CloudWatch 控制台时，您可以从各种预构建的示例查询中进行选择，也可以创建自己的查询。创建查询时，您可以使用构建器视图，其能以交互方式提示您并支持浏览现有指标和维度以轻松构建查询，还可以使用编辑器视图从头开始编写查询、编辑在构建器视图中构建的查询以及编辑示例查询以进行自定义。

借助 Metric Insights，您可以大规模运行查询。通过使用 GROUP BY 子句，您可以根据自己的使用案例，灵活地将指标实时分组为每个特定维度值的不同时间序列。因为 Metrics Insights 查询包括 ORDER BY 能力，您可以使用 Metrics Insights 进行“前 N 个”类型的查询，这些查询可以扫描您账户中的数百万个指标，并返回例如前 10 个 CPU 消耗最多的实例，以帮助您查明和纠正应用程序中的延迟问题。

主题

- [构建查询 \(p. 68\)](#)
- [Metrics Insights 查询组件和语法 \(p. 68\)](#)
- [将 Metrics Insights 查询与指标数学配合使用 \(p. 73\)](#)
- [SQL 推理 \(p. 74\)](#)
- [Metrics Insights 示例查询 \(p. 75\)](#)
- [Metrics Insights 限制 \(p. 81\)](#)

- [Metrics Insights 术语表 \(p. 81\)](#)
- [Metrics Insights 问题排查 \(p. 81\)](#)

构建查询

您可以使用 CloudWatch 控制台、Amazon CLI 或者 Amazon SDK 来运行 CloudWatch Metrics Insights 查询。在控制台中运行的查询免费。有关 CloudWatch 定价的信息，请参阅 [Amazon CloudWatch 定价](#)。

有关使用 Amazon SDK 执行 Metrics Insights 查询的更多信息，请参阅 [GetMetricData](#)。

要使用 CloudWatch 控制台运行查询，请执行以下步骤：

使用 Metrics Insights 查询指标

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Metrics (指标)、All metrics (所有指标)。
3. 选择 Query (查询) 选项卡。
4. (可选) 要运行预构建的示例查询，请选择 Add query (添加查询)，然后选择要运行的查询。如果您对此查询满意，您可以跳过此过程的其余部分。或者，您可以选择 Editor (编辑器) 来编辑示例查询，然后选择 Run (运行) 以运行修改后的查询。
5. 要创建查询，您可以使用 Builder (构建器) 视图、Editor (编辑器) 视图，还可以使用两者的组合。您可以随时在两个视图之间切换，并在两个视图中查看正在进行的工作。

在 Builder (构建器) 视图中，您可以浏览并选择指标命名空间、指标名称、筛选条件、分组和排序选项。对于其中每个选项，查询构建器都会为您提供您环境中的可选项列表，供您选择。

在 Editor (编辑器) 视图中，您可以开始编写查询。输入时，编辑器会根据您当前已输入的字符提供建议。

6. 如果对查询感到满意，请选择 Run (运行)。
7. (可选) 编辑已绘制图表查询的另一种方法是选择 Graphed metrics (已绘制图表指标) 选项卡，然后选择 Details (详细信息) 列中查询公式旁边的编辑图标。
8. (可选) 要从图表中删除查询，请选择 Graphed metrics (已绘制图表指标)，然后选择显示查询一行右侧的 X 图标。

Metrics Insights 查询组件和语法

CloudWatch Metrics Insights 语法如下。

```
SELECT FUNCTION(metricName)
FROM namespace | SCHEMA(...)
[ WHERE labelKey OPERATOR labelValue [AND ... ] ]
[ GROUP BY labelKey [ , ... ] ]
[ ORDER BY FUNCTION() [ DESC | ASC ] ]
[ LIMIT number ]
```

Metrics Insights 查询中可能的子句如下。关键字不区分大小写，但是指标名称、命名空间和维度等标识符需区分大小写。

SELECT

必需。指定用于聚合每个时间段的观测值的函数 (由提供的时段确定)。还可以指定要查询的指标的名称。

FUNCTION 的有效值为 AVG、COUNT、MAX、MIN 和 SUM。

- AVG 计算查询匹配的观测值的平均值。
- COUNT 返回查询匹配的观测值的计数。
- MAX 返回查询匹配的观测值的最大值。
- MIN 返回查询匹配的观测值的最小值。
- SUM 计算查询匹配的观测值的总和。

FROM

必需。指定指标源。您可以指定包含要查询的指标的指标命名空间，也可以指定 SCHEMA 表函数。指标命名空间的示例包括："AWS/EC2"、"AWS/Lambda" 以及您为自定义指标创建的指标命名空间。

包括 / 或任何其他不是字母、数字或下划线的字符在内的指标命名空间必须放在双引号中。有关更多信息，请参阅[什么内容需要引号或转义字符？](#) (p. 70)。

SCHEMA

一个可以在 FROM 子句中使用的可选表函数。使用 SCHEMA 将查询结果范围缩小到仅与维度列表完全匹配的指标或没有维度的指标。

如果您使用 SCHEMA 子句，它必须包含至少一个参数，并且第一个参数必须是被查询的指标命名空间。如果您仅用此命名空间参数指定 SCHEMA，则结果范围只限于没有任何维度的指标。

如果您使用其他参数指定 SCHEMA，命名空间参数之后的其他参数必须为标签键。标签键必须是维度名称。如果您指定了一个或多个标签键，则结果范围仅限于具有该确切维度集的指标。这些标签键的顺序无关紧要。

例如：

- SELECT AVG(CPUUtilization) FROM "AWS/EC2" 匹配 AWS/EC2 命名空间中的所有 CPUUtilization 指标（无论它们的维度如何），并且会返回单个聚合时间序列。
- SELECT AVG(CPUUtilization) FROM SCHEMA("AWS/EC2") 只匹配 AWS/EC2 命名空间中没有任何维度的 CPUUtilization 指标。
- SELECT AVG(CPUUtilization) FROM SCHEMA("AWS/EC2", InstanceId) 只匹配向 CloudWatch 报告且只有一个维度（即 CPUUtilization）的 InstanceId 指标。
- SELECT SUM(RequestCount) FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone) 只匹配从 AWS/ApplicationELB 向 CloudWatch 报告且只带有 LoadBalancer 和 AvailabilityZone 两个维度的 RequestCount 指标。

WHERE

可选。使用一个或多个标签键的特定标签值，将结果筛选为仅与指定表达式匹配的指标。例如，WHERE InstanceType = 'c3.4xlarge' 将结果筛选为仅 c3.4xlarge 实例类型，WHERE InstanceType != 'c3.4xlarge' 将结果筛选为除 c3.4xlarge 外的所有实例类型。

标签值必须始终放在单引号中。

支持的运算符

WHERE 子句支持以下运算符：

- = 标签值必须与指定的字符串匹配。
- != 标签值不得与指定的字符串匹配。
- AND 指定的两个条件都必须为真才能匹配。您可以使用多个 AND 关键字来指定两个或更多条件。

GROUP BY

可选。将查询结果分组为多个时间序列，每个时间序列对应指定的一个或多个标签键的不同值。例如，使用 GROUP BY InstanceId 为每个 InstanceId 值返回不同的时间序列。使用 GROUP BY ServiceName, Operation 为 ServiceName 和 Operation 值的每个可能组合创建不同的时间序列。

通过 GROUP BY 子句，默认情况下，将使用 GROUP BY 子句中指定的标签序列将结果按字母升序排列。要更改结果的顺序，请在查询中添加 ORDER BY 子句。

Note

如果某些匹配的指标不包含 GROUP BY 子句中指定的标签键，则会返回名为 other 的空组。例如，如果指定 GROUP BY ServiceName, Operation 而且某些返回的指标不包括作为维度的 ServiceName，则这些指标将显示为将 other 作为 ServiceName 的值。

ORDER BY

可选。如果查询返回多个时间序列，则指定返回的时间序列的使用顺序。该顺序基于您在 ORDER BY 子句中指定的 FUNCTION 找到的值。FUNCTION 用于计算每个返回的时间序列中的单个标量值，该值用于确定顺序。

还可以指定使用升序 ASC 或降序 DESC。如果您省略这一步，则默认为升序 ASC。

例如，添加 ORDER BY MAX() DESC 子句将按时间范围内观察到的最大数据点以降序对结果进行排序：这意味着具有最高最大数据点的时间序列会首先返回。

要在 ORDER BY 子句中使用的有效函数为 AVG()、COUNT()、MAX()、MIN() 和 SUM()。

如果您将 ORDER BY 子句与 LIMIT 子句配合使用，则生成的查询为“前 N 个”查询。ORDER BY 对于可能返回大量指标的查询也很有用，因为每个查询返回的时间序列不超过 500 个。如果查询匹配超过 500 个时间序列，并且您使用 ORDER BY 子句并对时间序列进行排序，则排序中前 500 个时间序列是返回的时间序列。

LIMIT

可选。将查询返回的时间序列数量限制为指定的值。您可以指定的最大值为 500，而不指定 LIMIT 的查询也可以返回不超过 500 个时间序列。

将 LIMITED 子句与 ORDER BY 子句配合使用，将获得“前 N 个”查询。

什么内容需要引号或转义字符？

在查询中，标签值必须始终放在单引号中。例如，SELECT MAX(CPUUtilization) FROM "AWS/EC2" WHERE AutoScalingGroupName = 'my-production-fleet'。

包含字母、数字和下划线 (_) 以外的字符的指标命名空间、指标名称和标签键必须放在双引号中。例如，SELECT MAX("My.Metric")。

如果其中一个已包含双引号或单引号（如 Bytes"Input"），您必须用反斜杠转义每个引号，如 SELECT AVG("Bytes\"Input\"")。

如果指标命名空间、指标名称或标签键包含的词为 Metrics Insights 中的保留关键字，则这些词也必须放在双引号中。例如，如果您有名为 LIMIT 的指标，您可以使用 SELECT AVG("LIMIT")。将任何命名空间、指标名称或标签（即使不包含保留关键字）放在双引号中也有效。

有关保留关键字的完整列表，请参阅 [保留关键字 \(p. 71\)](#)。

逐步构建丰富的查询

本节说明了如何逐步构建一个使用所有可能子句的完整示例。

我们从以下查询开始，该查询聚合了使用 LoadBalancer 和 AvailabilityZone 两个维度收集的所有 Application Load Balancer RequestCount 指标。

```
SELECT SUM(RequestCount)
```



```
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
```

现在，如果我们只想查看来自特定负载均衡器的指标，我们可以添加 WHERE 子句，将返回的指标限制为 LoadBalancer 维度的值为 app/load-balancer-1 的指标。

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
```

前面的查询将此负载均衡器的所有可用区中的 RequestCount 指标聚合为一个时间序列。如果我们想看到每个可用区的不同时间序列，可以添加 GROUP BY 子句。

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
```

接下来，我们可能希望对这些结果进行排序以首先查看最高值。以下 ORDER BY 子句按查询时间范围内每个时间序列报告的最大值以降序对时间序列进行排序：

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
ORDER BY MAX() DESC
```

最后，如果我们主要对“前 N 个”类型的查询感兴趣，我们可以使用 LIMIT 子句。这最后一个例子将结果限制为仅带有五个最高 MAX 值的时间序列。

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
ORDER BY MAX() DESC
LIMIT 5
```

保留关键字

以下是 CloudWatch Metrics Insights 中的保留关键字。如果查询中的命名空间、指标名称或标签键中包含这些词中的任何一个，则必须将其放在双引号中。保留关键字不区分大小写。

```
"ABORT" "ABORTSESSION" "ABS" "ABSOLUTE" "ACCESS" "ACCESSIBLE" "ACCESS_LOCK" "ACCOUNT"
"ACOS" "ACOSH" "ACTION" "ADD" "ADD_MONTHS"
"ADMIN" "AFTER" "AGGREGATE" "ALIAS" "ALL" "ALLOCATE" "ALLOW" "ALTER" "ALTERAND" "AMP"
"ANALYSE" "ANALYZE" "AND" "ANSIDATE" "ANY" "ARE" "ARRAY",
"ARRAY_AGG" "ARRAY_EXISTS" "ARRAY_MAX_CARDINALITY" "AS" "ASC" "ASENSITIVE" "ASIN" "ASINH"
"ASSERTION" "ASSOCIATE" "ASUTIME" "ASYMMETRIC" "AT",
"ATAN" "ATAN2" "ATANH" "ATOMIC" "AUDIT" "AUTHORIZATION" "AUX" "AUXILIARY" "AVE" "AVERAGE"
"AVG" "BACKUP" "BEFORE" "BEGIN" "BEGIN_FRAME" "BEGIN_PARTITION",
"BETWEEN" "BIGINT" "BINARY" "BIT" "BLOB" "BOOLEAN" "BOTH" "BREADTH" "BREAK" "BROWSE" "BT"
"BUFFERPOOL" "BULK" "BUT" "BY" "BYTE" "BYTEINT" "BYTES" "CALL",
"CALLED" "CAPTURE" "CARDINALITY" "CASCADE" "CASCADED" "CASE" "CASESPECIFIC" "CASE_N" "CAST"
"CATALOG" "CCSID" "CD" "CEIL" "CEILING" "CHANGE" "CHAR",
"CHAR2HEXINT" "CHARACTER" "CHARACTERS" "CHARACTER_LENGTH" "CHARS" "CHAR_LENGTH" "CHECK"
"CHECKPOINT" "CLASS" "CLASSIFIER" "CLOB" "CLONE" "CLOSE" "CLUSTER",
"CLUSTERED" "CM" "COALESCE" "COLLATE" "COLLATION" "COLLECT" "COLLECTION" "COLLID" "COLUMN"
"COLUMN_VALUE" "COMMENT" "COMMIT" "COMPLETION" "COMPRESS" "COMPUTE",
```


"CONCAT" "CONCURRENTLY" "CONDITION" "CONNECT" "CONNECTION" "CONSTRAINT" "CONSTRAINTS"
"CONSTRUCTOR" "CONTAINS" "CONTAINSTABLE" "CONTENT" "CONTINUE" "CONVERT",
"CONVERT_TABLE_HEADER" "COPY" "CORR" "CORRESPONDING" "COS" "COSH" "COUNT" "COVAR_POP"
"COVAR_SAMP" "CREATE" "CROSS" "CS" "CSUM" "CT" "CUBE" "CUME_DIST",
"CURRENT" "CURRENT_CATALOG" "CURRENT_DATE" "CURRENT_DEFAULT_TRANSFORM_GROUP"
"CURRENT_LC_CTYPE" "CURRENT_PATH" "CURRENT_ROLE" "CURRENT_ROW" "CURRENT_SCHEMA",
"CURRENT_SERVER" "CURRENT_TIME" "CURRENT_TIMESTAMP" "CURRENT_TIMEZONE"
"CURRENT_TRANSFORM_GROUP_FOR_TYPE" "CURRENT_USER" "CURRVAL" "CURSOR" "CV" "CYCLE" "DATA",
"DATABASE" "DATABASES" "DATABLOCKSIZE" "DATE" "DATEFORM" "DAY" "DAYS" "DAY_HOUR"
"DAY_MICROSECOND" "DAY_MINUTE" "DAY_SECOND" "DBCC" "DBINFO" "DEALLOCATE" "DEC",
"DECFLOAT" "DECIMAL" "DECLARE" "DEFAULT" "DEFERRABLE" "DEFERRED" "DEFINE" "DEGREES" "DEL"
"DELAYED" "DELETE" "DENSE_RANK" "DENY" "DEPTH" "DEREF" "DESC" "DESCRIBE",
"DESCRIPTOR" "DESTROY" "DESTRUCTOR" "DETERMINISTIC" "DIAGNOSTIC" "DIAGNOSTICS" "DICTIONARY"
"DISABLE" "DISABLED" "DISALLOW" "DISCONNECT" "DISK" "DISTINCT",
"DISTINCTROW" "DISTRIBUTED" "DIV" "DO" "DOCUMENT" "DOMAIN" "DOUBLE" "DROP" "DSSIZE" "DUAL"
"DUMP" "DYNAMIC" "EACH" "ECHO" "EDITPROC" "ELEMENT" "ELSE" "ELSEIF",
"EMPTY" "ENABLED" "ENCLOSED" "ENCODING" "ENCRYPTION" "END" "END-EXEC" "ENDING" "END_FRAME"
"END_PARTITION" "EQ" "EQUALS" "ERASE" "ERRLVL" "ERROR" "ERRORFILES",
"ERRORTABLES" "ESCAPE" "ESCAPED" "ET" "EVERY" "EXCEPT" "EXCEPTION" "EXCLUSIVE" "EXEC"
"EXECUTE" "EXISTS" "EXIT" "EXP" "EXPLAIN" "EXTERNAL" "EXTRACT" "FALLBACK"
"FALSE" "FASTEXPORT" "FENCED" "FETCH" "FIELDPROC" "FILE" "FILLFACTOR" "FILTER" "FINAL"
"FIRST" "FIRST_VALUE" "FLOAT" "FLOAT4" "FLOAT8" "FLOOR"
"FOR" "FORCE" "FOREIGN" "FORMAT" "FOUND" "FRAME_ROW" "FREE" "FREESPACE" "FREETEXT"
"FREETEXTTABLE" "FREEZE" "FROM" "FULL" "FULLTEXT" "FUNCTION"
"FUSION" "GE" "GENERAL" "GENERATED" "GET" "GIVE" "GLOBAL" "GO" "GOTO" "GRANT" "GRAPHIC"
"GROUP" "GROUPING" "GROUPS" "GT" "HANDLER" "HASH"
"HASHAMP" "HASHBAKAMP" "HASHBUCKET" "HASHROW" "HAVING" "HELP" "HIGH_PRIORITY" "HOLD"
"HOLDLOCK" "HOUR" "HOURS" "HOUR_MICROSECOND" "HOUR_MINUTE"
"HOUR_SECOND" "IDENTIFIED" "IDENTITY" "IDENTITYCOL" "IDENTITY_INSERT" "IF" "IGNORE" "ILIKE"
"IMMEDIATE" "IN" "INCLUSIVE" "INCONSISTENT" "INCREMENT"
"INDEX" "INDICATOR" "INFILE" "INHERIT" "INITIAL" "INITIALIZE" "INITIALLY" "INITIATE"
"INNER" "INOUT" "INPUT" "INS" "INSENSITIVE" "INSERT" "INSTEAD"
"INT" "INT1" "INT2" "INT3" "INT4" "INT8" "INTEGER" "INTEGERDATE" "INTERSECT" "INTERSECTION"
"INTERVAL" "INTO" "IO_AFTER_GTIDS" "IO_BEFORE_GTIDS"
"IS" "ISNULL" "ISOBID" "ISOLATION" "ITERATE" "JAR" "JOIN" "JOURNAL" "JSON_ARRAY"
"JSON_ARRAYAGG" "JSON_EXISTS" "JSON_OBJECT" "JSON_OBJECTAGG"
"JSON_QUERY" "JSON_TABLE" "JSON_TABLE_PRIMITIVE" "JSON_VALUE" "KEEP" "KEY" "KEYS" "KILL"
"KURTOSIS" "LABEL" "LAG" "LANGUAGE" "LARGE" "LAST"
"LAST_VALUE" "LATERAL" "LC_CTYPE" "LE" "LEAD" "LEADING" "LEAVE" "LEFT" "LESS" "LEVEL"
"LIKE" "LIKE_REGEX" "LIMIT" "LINEAR" "LINENO" "LINES"
"LISTAGG" "LN" "LOAD" "LOADING" "LOCAL" "LOCALE" "LOCALTIME" "LOCALTIMESTAMP" "LOCATOR"
"LOCATORS" "LOCK" "LOCKING" "LOCKMAX" "LOCKSIZE" "LOG"
"LOG10" "LOGGING" "LOGON" "LONG" "LONGBLOB" "LONGTEXT" "LOOP" "LOWER" "LOW_PRIORITY" "LT"
"MACRO" "MAINTAINED" "MAP" "MASTER_BIND"
"MASTER_SSL_VERIFY_SERVER_CERT" "MATCH" "MATCHES" "MATCH_NUMBER" "MATCH_RECOGNIZE"
"MATERIALIZED" "MAVG" "MAX" "MAXEXTENTS" "MAXIMUM" "MAXVALUE"
"MCHARACTERS" "MDIFF" "MEDIUMBLOB" "MEDIUMINT" "MEDIUMTEXT" "MEMBER" "MERGE" "METHOD"
"MICROSECOND" "MICROSECONDS" "MIDDLEINT" "MIN" "MINDEX"
"MINIMUM" "MINUS" "MINUTE" "MINUTES" "MINUTE_MICROSECOND" "MINUTE_SECOND" "MLINREG" "MLOAD"
"MLSLABEL" "MOD" "MODE" "MODIFIES" "MODIFY"
"MODULE" "MONITOR" "MONRESOURCE" "MONSESSION" "MONTH" "MONTHS" "MSUBSTR" "MSUM" "MULTISET"
"NAMED" "NAMES" "NATIONAL" "NATURAL" "NCHAR" "NCLOB"
"NE" "NESTED_TABLE_ID" "NEW" "NEW_TABLE" "NEXT" "NEXTVAL" "NO" "NOAUDIT" "NOCHECK"
"NOCOMPRESS" "NONCLUSTERED" "NONE" "NORMALIZE" "NOT" "NOTNULL"
"NOWAIT" "NO_WRITE_TO_BINLOG" "NTH_VALUE" "NTILE" "NULL" "NULLIF" "NULLIFZERO" "NULLS"
"NUMBER" "NUMERIC" "NUMPARTS" "OBID" "OBJECT" "OBJECTS"
"OCCURRENCES_REGEX" "OCTET_LENGTH" "OF" "OFF" "OFFLINE" "OFFSET" "OFFSETS" "OLD"
"OLD_TABLE" "OMIT" "ON" "ONE" "ONLINE" "ONLY" "OPEN" "OPENDATASOURCE"
"OPENQUERY" "OPENROWSET" "OPENXML" "OPERATION" "OPTIMIZATION" "OPTIMIZE" "OPTIMIZER_COSTS"
"OPTION" "OPTIONALLY" "OR" "ORDER" "ORDINALITY" "ORGANIZATION"
"OUT" "OUTER" "OUTFILE" "OUTPUT" "OVER" "OVERLAPS" "OVERLAY" "OVERRIDE" "PACKAGE" "PAD"
"PADDED" "PARAMETER" "PARAMETERS" "PART" "PARTIAL" "PARTITION"
"PARTITIONED" "PARTITIONING" "PASSWORD" "PATH" "PATTERN" "PCTFREE" "PER" "PERCENT"
"PERCENTILE" "PERCENTILE_CONT" "PERCENTILE_DISC" "PERCENT_RANK" "PERIOD" "PERM"
"PERMANENT" "PIECESIZE" "PIVOT" "PLACING" "PLAN" "PORTION" "POSITION" "POSITION_REGEX"
"POSTFIX" "POWER" "PRECEDES" "PRECISION" "PREFIX" "PREORDER"

"PREPARE" "PRESERVE" "PREVVAL" "PRIMARY" "PRINT" "PRIOR" "PRIQTY" "PRIVATE" "PRIVILEGES"
"PROC" "PROCEDURE" "PROFILE" "PROGRAM" "PROPORTIONAL"
"PROTECTION" "PSID" "PTF" "PUBLIC" "PURGE" "QUALIFIED" "QUALIFY" "QUANTILE" "QUERY"
"QUERYNO" "RADIANS" "RAISERROR" "RANDOM" "RANGE" "RANGE_N" "RANK"
"RAW" "READ" "READS" "READTEXT" "READ_WRITE" "REAL" "RECONFIGURE" "RECURSIVE" "REF"
"REFERENCES" "REFERENCING" "REFRESH" "REGEXP" "REGR_AVGX" "REGR_AVGY"
"REGR_COUNT" "REGR_INTERCEPT" "REGR_R2" "REGR_SLOPE" "REGR_SXX" "REGR_SXY" "REGR_SYY"
"RELATIVE" "RELEASE" "RENAME" "REPEAT" "REPLACE" "REPLICATION"
"REPOVERRIDE" "REQUEST" "REQUIRE" "RESIGNAL" "RESOURCE" "RESTART" "RESTORE" "RESTRICT"
"RESULT" "RESULT_SET_LOCATOR" "RESUME" "RET" "RETRIEVE" "RETURN"
"RETURNING" "RETURNS" "REVALIDATE" "REVERT" "REVOKE" "RIGHT" "RIGHTS" "RLIKE" "ROLE"
"ROLLBACK" "ROLLFORWARD" "ROLLUP" "ROUND_CEILING" "ROUND_DOWN"
"ROUND_FLOOR" "ROUND_HALF_DOWN" "ROUND_HALF_EVEN" "ROUND_HALF_UP" "ROUND_UP" "ROUTINE"
"ROW" "ROWCOUNT" "ROWGUIDCOL" "ROWID" "ROWNUM" "ROWS" "ROWSET"
"ROW_NUMBER" "RULE" "RUN" "RUNNING" "SAMPLE" "SAMPLEID" "SAVE" "SAVEPOINT" "SCHEMA"
"SCHEMAS" "SCOPE" "SCRATCHPAD" "SCROLL" "SEARCH" "SECOND" "SECONDS"
"SECOND_MICROSECOND" "SECQTY" "SECTION" "SECURITY" "SECURITYAUDIT" "SEEK" "SEL" "SELECT"
"SEMANTICKEYPHRASETABLE" "SEMANTICSIMILARITYDETAILSTABLE"
"SEMANTICSIMILARITYTABLE" "SENSITIVE" "SEPARATOR" "SEQUENCE" "SESSION" "SESSION_USER" "SET"
"SETRESRATE" "SETS" "SETSESSRATE" "SETUSER" "SHARE" "SHOW"
"SHUTDOWN" "SIGNAL" "SIMILAR" "SIMPLE" "SIN" "SINH" "SIZE" "SKEW" "SKIP" "SMALLINT" "SOME"
"SOUNDEX" "SOURCE" "SPACE" "SPATIAL" "SPECIFIC" "SPECIFICTYPE"
"SPOOL" "SQL" "SQLEXCEPTION" "SQLSTATE" "SQLTEXT" "SQLWARNING" "SQL_BIG_RESULT"
"SQL_CALC_FOUND_ROWS" "SQL_SMALL_RESULT" "SQRT" "SS" "SSL" "STANDARD"
"START" "STARTING" "STARTUP" "STAT" "STATE" "STATEMENT" "STATIC" "STATISTICS" "STAY"
"STDDEV_POP" "STDDEV_SAMP" "STEPINFO" "STOGROUP" "STORED" "STORES"
"STRAIGHT_JOIN" "STRING_CS" "STRUCTURE" "STYLE" "SUBMULTISET" "SUBSCRIBER" "SUBSET"
"SUBSTR" "SUBSTRING" "SUBSTRING_REGEX" "SUCCEEDS" "SUCCESSFUL"
"SUM" "SUMMARY" "SUSPEND" "SYMMETRIC" "SYNONYM" "SYSDATE" "SYSTEM" "SYSTEM_TIME"
"SYSTEM_USER" "SYSTEMTIMESTAMP" "TABLE" "TABLESAMPLE" "TABLESPACE" "TAN"
"TANH" "TBL_CS" "TEMPORARY" "TERMINATE" "TERMINATED" "TEXTSIZE" "THAN" "THEN" "THRESHOLD"
"TIME" "TIMESTAMP" "TIMEZONE_HOUR" "TIMEZONE_MINUTE" "TINYBLOB"
"TINYINT" "TINYTEXT" "TITLE" "TO" "TOP" "TRACE" "TRAILING" "TRAN" "TRANSACTION" "TRANSLATE"
"TRANSLATE_CHK" "TRANSLATE_REGEX" "TRANSLATION" "TREAT"
"TRIGGER" "TRIM" "TRIM_ARRAY" "TRUE" "TRUNCATE" "TRY_CONVERT" "TSEQUAL" "TYPE" "UC"
"UNESCAPE" "UID" "UNDEFINED" "UNDER" "UNDO" "UNION" "UNIQUE"
"UNKNOWN" "UNLOCK" "UNNEST" "UNPIVOT" "UNSIGNED" "UNTIL" "UPD" "UPDATE" "UPDATETEXT"
"UPPER" "UPPERCASE" "USAGE" "USE" "USER" "USING" "UTC_DATE"
"UTC_TIME" "UTC_TIMESTAMP" "VALIDATE" "VALIDPROC" "VALUE" "VALUES" "VALUE_OF" "VARIABLE"
"VARBYTE" "VARCHAR" "VARCHAR2" "VARCHAER" "VARGRAPHIC"
"VARIABLE" "VARIADIC" "VARIANT" "VARYING" "VAR_POP" "VAR_SAMP" "VCAT" "VERBOSE"
"VERSIONING" "VIEW" "VIRTUAL" "VOLATILE" "VOLUMES" "WAIT" "WAITFOR"
"WHEN" "WHenever" "WHERE" "WHILE" "WIDTH_BUCKET" "WINDOW" "WITH" "WITHIN" "WITHIN_GROUP"
"WITHOUT" "WLM" "WORK" "WRITE" "WRITETEXT" "XMLCAST" "XMLEXISTS"
"XMLNAMESPACES" "XOR" "YEAR" "YEARS" "YEAR_MONTH" "ZEROFILL" "ZEROIFNULL" "ZONE"

将 Metrics Insights 查询与指标数学配合使用

您可以使用 Metrics Insights 查询作为指标数学函数的输入。有关指标数学的更多信息，请参阅 [使用指标数学 \(p. 116\)](#)。

不包括 GROUP BY 子句的 Metrics Insights 查询将返回单个时间序列。因此，其返回的结果可以与以单个时间序列作为输入的任何指标数学函数一起使用。

包括 GROUP BY 子句的 Metrics Insights 查询将返回多个时间序列。因此，其返回的结果可以与以时间序列数组作为输入的任何指标数学函数一起使用。

例如，以下查询以时间序列数组的形式返回为区域中每个存储桶下载的总字节数：

```
SELECT SUM(BytesDownloaded)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
```

```
GROUP BY BucketName
```

在控制台或 [GetMetricData](#) 操作中的图表上，此查询的结果为 `q1`。此查询返回以字节为单位的結果，因此如果您想查看 MB 单位的结果，可以使用以下数学函数：

```
q1/1024/1024
```

SQL 推理

CloudWatch Metrics Insights 使用多种机制来推断给定 SQL 查询的意图。

主题

- [时间分段 \(p. 74\)](#)
- [字段投影 \(p. 74\)](#)
- [ORDER BY 全局聚合 \(p. 74\)](#)

时间分段

根据请求的时段，查询生成的时间序列数据点将累计到时间段中。要在标准 SQL 中聚合值，必须定义一个明确的 GROUP BY 子句，以便一起收集给定时段的所有观测值。由于这是查询时间序列数据的标准方式，CloudWatch Metrics Insights 可推断时间分段，而无需明确表达 GROUP BY 子句。

例如，当执行时间为 1 分钟的查询时，属于该分钟的所有观测值，直到下一分钟（排除）为止，都将累计到该时间段的开始时间。这使得 Metrics Insights SQL 语句更加简洁明了。

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
```

上一个查询返回单个时间序列（时间戳值对），表示所有 Amazon EC2 实例的平均 CPU 使用率。假设请求的时段为 1 分钟，则返回的每个数据点代表在特定 1 分钟间隔（包括开始时间，不包括结束时间）内测量的所有观测值的平均值。与特定数据点相关的时间戳是时间段的开始时间。

字段投影

Metrics Insights 查询始终返回时间戳投影。您不需要在 SELECT 子句中指定时间戳列来获取每个相应数据点值的时间戳。有关如何计算时间戳的详细信息，请参阅 [时间分段 \(p. 74\)](#)。

使用 GROUP BY 时，还可以在结果中推断和投影每个组名称，以便您可以对返回的时间序列进行分组。

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
```

上一个查询将返回每个 Amazon EC2 实例的时间序列。每个时间序列都在实例 ID 的值之后进行标注。

ORDER BY 全局聚合

使用 ORDER BY 时，FUNCTION() 推断要按哪个聚合函数排序（查询指标的数据点值）。聚合操作在查询的时间窗口中对每个单独时间序列的所有匹配数据点执行。

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
```

```
GROUP BY InstanceId
ORDER BY MAX()
LIMIT 10
```

上一个查询返回每个 Amazon EC2 实例的 CPU 使用率，并将结果集限制为 10 个条目。结果根据请求的时间窗口内各个时间序列的最大值进行排序。ORDER BY 子句在 LIMIT 之前应用，以便根据 10 个以上的时间序列计算排序。

Metrics Insights 示例查询

本节包含有用的 CloudWatch Metrics Insights 查询的示例，您可以直接复制和使用，或在查询编辑器中复制并修改。控制台中已经提供一些示例，您可以通过选择 Metrics (指标) 视图中的 Add query (添加查询) 来访问它们。

Application Load Balancer 示例

所有负载均衡器的请求总数

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer)
```

前 10 大最活跃的负载均衡器

```
SELECT MAX(ActiveConnectionCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer)
GROUP BY LoadBalancer
ORDER BY SUM() DESC
LIMIT 10
```

Amazon API 使用示例

按账户中的调用次数划分，排名前 20 的 Amazon API

```
SELECT COUNT(CallCount)
FROM SCHEMA("AWS/Usage", Class, Resource, Service, Type)
WHERE Type = 'API'
GROUP BY Service, Resource
ORDER BY COUNT() DESC
LIMIT 20
```

按调用排序的 CloudWatch API

```
SELECT COUNT(CallCount)
FROM SCHEMA("AWS/Usage", Class, Resource, Service, Type)
WHERE Type = 'API' AND Service = 'CloudWatch'
GROUP BY Resource
ORDER BY COUNT() DESC
```

DynamoDB 示例

按消耗的读取数排序的前 10 个表

```
SELECT SUM(ProvisionedWriteCapacityUnits)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
```

```
ORDER BY MAX() DESC LIMIT 10
```

按返回的字节数排序的前 10 个表

```
SELECT SUM(ReturnedBytes)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

按用户错误排序的前 10 个表

```
SELECT SUM(UserErrors)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

Amazon Elastic Block Store 示例

按写入字节数排序的前 10 个 Amazon EBS 卷

```
SELECT SUM(VolumeWriteBytes)
FROM SCHEMA("AWS/EBS", VolumeId)
GROUP BY VolumeId
ORDER BY SUM() DESC
LIMIT 10
```

Amazon EBS 卷的平均写入时间

```
SELECT AVG(VolumeTotalWriteTime)
FROM SCHEMA("AWS/EBS", VolumeId)
```

Amazon EC2 示例

按最高排序的 EC2 实例 CPU 使用率

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
ORDER BY AVG() DESC
```

整个机群的平均 CPU 使用率

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
```

按最高 CPU 使用率排序的前 10 个实例

```
SELECT MAX(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
ORDER BY MAX() DESC
LIMIT 10
```

在这种情况下，CloudWatch 代理正在根据应用程序收集 **CPUUtilization** 指标。此查询筛选特定应用程序名称的此指标的平均值。

```
SELECT AVG(CPUUtilization)
FROM "AWS/CWAgent"
WHERE ApplicationName = 'eCommerce'
```

Amazon Elastic Container Service 示例

所有 ECS 集群的平均 CPU 使用率

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/ECS", ClusterName)
```

按内存利用率排序的前 10 个集群

```
SELECT AVG(MemoryUtilization)
FROM SCHEMA("AWS/ECS", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC
LIMIT 10
```

按 CPU 使用率排序的前 10 项服务

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/ECS", ClusterName, ServiceName)
GROUP BY ClusterName, ServiceName
ORDER BY AVG() DESC
LIMIT 10
```

按运行任务排序的前 10 项服务 (Container Insights)

```
SELECT AVG(RunningTaskCount)
FROM SCHEMA("ECS/ContainerInsights", ClusterName, ServiceName)
GROUP BY ClusterName, ServiceName
ORDER BY AVG() DESC
LIMIT 10
```

Amazon Elastic Kubernetes Service Container Insights 示例

所有 EKS 集群的平均 CPU 使用率

```
SELECT AVG(pod_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
```

按节点 CPU 使用率排序的前 10 个集群

```
SELECT AVG(node_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC LIMIT 10
```

按 pod 内存利用率排序的前 10 个集群

```
SELECT AVG(pop_memory_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
GROUP BY ClusterName
```

```
ORDER BY AVG() DESC LIMIT 10
```

按 CPU 使用率排序的前 10 个节点

```
SELECT AVG(node_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName, NodeName)
GROUP BY ClusterName, NodeName
ORDER BY AVG() DESC LIMIT 10
```

按内存利用率排序的前 10 个 pod

```
SELECT AVG(pod_memory_utilization)
FROM SCHEMA("ContainerInsights", ClusterName, PodName)
GROUP BY ClusterName, PodName
ORDER BY AVG() DESC LIMIT 10
```

EventBridge 示例

按调用排序的前 10 条规则

```
SELECT SUM(Invocations)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

按失败调用排序的前 10 条规则

```
SELECT SUM(FailedInvocations)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

按匹配规则排序的前 10 条规则

```
SELECT SUM(MatchedEvents)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

Kinesis 示例

按写入字节数排序的前 10 个流

```
SELECT SUM(PutRecords.Bytes)
FROM SCHEMA("AWS/Kinesis", StreamName)
GROUP BY StreamName
ORDER BY SUM() DESC LIMIT 10
```

按流中最早的项目排序的前 10 个流

```
SELECT MAX(GetRecords.IteratorAgeMilliseconds)
FROM SCHEMA("AWS/Kinesis", StreamName)
GROUP BY StreamName
ORDER BY MAX() DESC LIMIT 10
```


Lambda 示例

按调用次数排序的 Lambda 函数

```
SELECT SUM(Invocations)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY SUM() DESC
```

按最长运行时排序的前 10 个 Lambda 函数

```
SELECT AVG(Duration)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY MAX() DESC
LIMIT 10
```

按错误计数排序的前 10 个 Lambda 函数

```
SELECT SUM(Errors)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY SUM() DESC
LIMIT 10
```

CloudWatch Logs 示例

按传入事件排序的前 10 个日志组

```
SELECT SUM(IncomingLogEvents)
FROM SCHEMA("AWS/Logs", LogGroupName)
GROUP BY LogGroupName
ORDER BY SUM() DESC LIMIT 10
```

按写入字节数排序的前 10 个日志组

```
SELECT SUM(IncomingBytes)
FROM SCHEMA("AWS/Logs", LogGroupName)
GROUP BY LogGroupName
ORDER BY SUM() DESC LIMIT 10
```

Amazon RDS 示例

按最高 CPU 使用率排序的前 10 个 Amazon RDS 实例

```
SELECT MAX(CPUUtilization)
FROM SCHEMA("AWS/RDS", DBInstanceIdentifier)
GROUP BY DBInstanceIdentifier
ORDER BY MAX() DESC
LIMIT 10
```

按写入数排序的前 10 个 Amazon RDS 集群

```
SELECT SUM(WriteIOPS)
FROM SCHEMA("AWS/RDS", DBClusterIdentifier)
```

```
GROUP BY DBClusterIdentifier
ORDER BY MAX() DESC
LIMIT 10
```

Amazon Simple Storage Service 示例

按存储桶排序的平均延迟

```
SELECT AVG(TotalRequestLatency)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
GROUP BY BucketName
ORDER BY AVG() DESC
```

按已下载字节数排序的前 10 个存储桶

```
SELECT SUM(BytesDownloaded)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
GROUP BY BucketName
ORDER BY SUM() DESC
LIMIT 10
```

Amazon Simple Notification Service 示例

SNS 主题发布的消息总数

```
SELECT SUM(NumberOfMessagesPublished)
FROM SCHEMA("AWS/SNS", TopicName)
```

按发布消息排序的前 10 个主题

```
SELECT SUM(NumberOfMessagesPublished)
FROM SCHEMA("AWS/SNS", TopicName)
GROUP BY TopicName
ORDER BY SUM() DESC
LIMIT 10
```

按消息传送失败排序的前 10 个主题

```
SELECT SUM(NumberOfNotificationsFailed)
FROM SCHEMA("AWS/SNS", TopicName)
GROUP BY TopicName
ORDER BY SUM() DESC
LIMIT 10
```

Amazon SQS 示例

按可见消息数量排序的前 10 个队列

```
SELECT AVG(ApproximateNumberOfMessagesVisible)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY AVG() DESC
LIMIT 10
```

前 10 个最活跃的队列

```
SELECT SUM(NumberOfMessagesSent)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY SUM() DESC
LIMIT 10
```

按最早消息的时间排序的前 10 个队列

```
SELECT AVG(ApproximateAgeOfOldestMessage)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY AVG() DESC
LIMIT 10
```

Metrics Insights 限制

CloudWatch Metrics Insights 目前有以下限制：

- 目前，您只能查询最近三个小时的数据。
- 单个查询可处理的指标不超过 10,000 个。这意味着，如果 SELECT、FROM 和 WHERE 子句匹配 10,000 个以上的指标，查询只处理找到的前 10,000 个指标。
- 单个查询可返回的时间序列不超过 500 个。这意味着，如果查询返回 500 个以上的指标，则查询结果中并非所有指标都将返回。如果您使用的是 ORDER BY 子句，则将对正在处理的所有指标进行排序，并根据您的 ORDER BY 子句，具有前 500 个最高值或最低值的指标将返回。

如果您不包括 ORDER BY 子句，则无法控制哪 500 个匹配的指标将返回。

- 每个 [GetMetricData](#) 操作只能有一个查询，但是您可以在控制面板中有多个小组件，每个小组件都包含一个查询。

Metrics Insights 术语表

标签

在 Metrics Insights 中，标签是一个键值对，用于确定查询范围以返回特定数据集，或者定义将查询结果分为单独时间序列的标准。标签键与 SQL 中的列名称类似。目前，标签必须是 CloudWatch 指标维度。

观测值

观测值是指在给定时间为给定指标记录的值。

Metrics Insights 问题排查

结果包括“其他”，但未将其作为维度

这意味着查询包含 GROUP BY 子句，该子句指定查询返回的某些指标中未使用的标签键。在这种情况下，会返回名为 Other 的空组。不包含该标签键的指标可能是聚合指标，这些指标返回该标签键的所有值之间聚合的值。

例如，假设我们有以下查询：

```
SELECT AVG(Faults)
FROM MyCustomNamespace
```

```
GROUP BY Operation, ServiceName
```

如果某些返回的指标不作为维度包括 ServiceName，则这些指标将显示为将 Other 作为 ServiceName 的值。

要防止在结果中看到“其他”，请在您的 FROM 子句中使用 SCHEMA，如以下示例所示：

```
SELECT AVG(Faults)
FROM SCHEMA(MyCustomNamespace, Operation)
GROUP BY Operation, ServiceName
```

这将返回的结果限制为同时具有 Operation 和 ServiceName 维度的指标。

我的图表中最早的时间戳的指标值比其他时间戳的指标值低

CloudWatch Metrics Insights 目前仅支持最近三个小时的数据。当您使用大于 1 分钟的时段（例如 5 分钟或 1 小时）绘制图表时，可能会出现最旧的数据点与预期值不同的情况。这是因为 Metrics Insights 查询只返回最近 3 个小时的数据，因此最早的数据点（早于 3 小时）只会考虑过去 3 个小时范围内测量的观测值。

使用 Metrics Explorer 按标签和属性监控资源

Metrics Explorer 是一个基于标签的工具，您可以使用它按标签和资源属性筛选、聚合和可视化指标，从而增强对服务的监控。这样，您可以灵活、动态地排查问题，一次创建多个图表并使用它们构建应用程序运行状况控制面板。

Metrics Explorer 可视化会动态变化，如果在创建 Metrics Explorer 小组件并将其添加到 CloudWatch 控制面板之后创建了匹配的资源，则新资源将自动显示在 Metrics Explorer 小组件中。

例如，如果您的所有 EC2 生产实例都具有 **production** 标签，则您可以使用 Metrics Explorer 筛选和聚合所有这些实例的指标，以了解其运行状况和性能。如果稍后创建了具有匹配标签的新实例，则该实例会自动添加到 Metrics Explorer 小组件中。

使用 Metrics Explorer，您可以选择如何聚合资源中与条件匹配的指标，以及是在单个图表中显示这些指标，还是在一个 Metrics Explorer 小组件中的不同图表上显示它们。

Metrics Explorer 包括您可以一键查看有用的可视化图表的模板，您还可以扩展这些模板以创建完全自定义的 Metrics Explorer 小组件。

Metrics Explorer 支持 Amazon 发布的指标和由 CloudWatch 代理发布的 EC2 指标，包括内存、磁盘和 CPU 指标。若要使用 Metrics Explorer 查看由 CloudWatch 代理发布的指标，您可能需要更新您的 CloudWatch 代理配置文件。有关更多信息，请参阅 [适合 Metrics Explorer 的 CloudWatch 代理配置 \(p. 83\)](#)

若要使用 Metrics Explorer 创建可视化图表并选择性地将其添加到控制面板，请按照下列步骤操作。

使用 Metrics Explorer 创建可视化图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格上，选择 Explorer。
3. 请执行下列操作之一：
 - 若要使用模板，请在当前显示 Empty Explorer（空 Explorer）的框中将其选中。

根据不同模板，Explorer 可能会立即显示指标图表。如果未显示，请在 From（从）框中选择一个或多个标签或属性，随后应会显示数据。如果未显示数据，请使用页面顶部的选项将图表中显示的时间范围扩大。

- 若要创建自定义可视化，请在 Metrics（指标）下，选择服务的单个指标或所有可用指标。

选择指标之后，您可以选择性地重复此步骤添加更多指标。

- 对于选定的每个指标，CloudWatch 会在指标名称后面显示其将随即使用的统计数据。若要更改此统计数据，请选择统计数据名称，然后选择您想要的统计数据。
- 在 From (从) 下，选择标签或资源属性来筛选结果。

执行此操作后，您可以选择性地重复此步骤以选择更多标签或资源属性。

如果您选择同一属性的多个值 (如两个 EC2 实例类型)，则 Explorer 将显示与任一选定属性匹配的所有资源。Explorer 将此选择视为 OR 操作。

如果您选择了不同的属性或标签，例如 **Production** 标签和 M5 实例类型，则仅显示与所有这些选择匹配的资源。Explorer 将此选择视为 AND 操作。

- (可选) 对于 Aggregate by (聚合依据)，选择要用于聚合指标的统计数据。然后，在 for (聚合方式) 旁边，选择如何聚合列表中的指标。您可以将当前显示的所有资源聚合在一起，也可以按单个标签或资源属性聚合。

根据您选择的聚合方式，结果可能是单个时间序列或多个时间序列。

- 在 Split by (拆分方式) 下，您可以选择将具有多个时间序列的单个图表拆分为多个图表。可以按您在 Split by (拆分方式) 下选择的多种标准进行拆分。
- 在 Graph options (绘制选项图形) 下，您可以通过更改周期、图表类型、图例放置和布局来优化图表。
- 若要将此可视化作为小组件添加到 CloudWatch 控制面板，请选择 Add to dashboard (添加到控制面板)。

适合 Metrics Explorer 的 CloudWatch 代理配置

要使 Metrics Explorer 能够发现 CloudWatch 代理发布的 EC2 指标，请确保 CloudWatch 代理配置文件包含以下值：

- 在 metrics 部分，请确保 aggregation_dimensions 参数包含 ["InstanceId"]。其还可以包含其他维度。
- 在 metrics 部分，请确保 append_dimensions 参数包含 { "InstanceId": "\${aws:InstanceId}" } 行。其还可以包含其他行。
- 在 metrics 部分 (metrics_collected 部分中)，检查您希望 Metrics Explorer 发现的每种资源类型所对应的部分，例如 cpu、disk 和 memory 部分。确保所有这些部分均包含 "resources": ["*"] line。。
- 在 cpu 部分 (metrics_collected 部分中)，请确保具有 "totalcpu": true 行。
- 您必须为 CloudWatch 代理收集的指标使用原定设置的 cwaAgent 命名空间，而不是自定义命名空间。

根据上一个列表中的设置，CloudWatch 代理会发布磁盘、CPU 和其它资源的聚合指标，这些指标可在 Metrics Explorer 中为使用它的所有实例标绘出来。

这些设置会重新发布您之前设置为以多个维度发布的指标，从而增加您的指标成本。

有关编辑 CloudWatch 代理配置文件的更多信息，请参阅 [手动创建或编辑 CloudWatch 代理配置文件 \(p. 686\)](#)。

使用指标流

您可以使用指标流持续地将 CloudWatch 指标流式传输到您所选的目标位置，实现近实时传送和低延迟。支持的目标位置包括 Amazon 目标位置 (例如 Amazon Simple Storage Service) 和多个第三方服务提供商目标位置。

CloudWatch 指标流有两种主要的使用情形：

- 数据湖 – 创建指标流并将其导向 Amazon Kinesis Data Firehose 传输流，该流将您的 CloudWatch 指标传送到数据湖（如 Amazon S3）。这样，您就能够持续更新监控数据，或者将此 CloudWatch 指标数据与账单和性能数据合并起来，创建丰富的数据集。然后，您可以使用 Amazon Athena 等工具深入了解成本优化、资源性能和资源利用率。
- 第三方提供商 – 使用第三方服务提供商使用流式 CloudWatch 数据监控和分析您的应用程序并排除应用程序故障。

通过将筛选选项与指标流结合使用，您可以流式传输所有 CloudWatch 指标、除您选择忽略的命名空间外的所有 CloudWatch 指标，或者仅您选择的范围更小的命名空间的指标。每个指标流最多可包含 1000 个包括或排除指标命名空间的筛选条件。单个指标流只能包括或排除筛选条件，不能同时既包括又排除。

创建指标流后，如果创建了与现有筛选条件匹配的新指标，则新指标将自动包含在指标流中。

每个账户或每个区域的指标流数量没有限制，正在流式传输的指标的更新数量也没有限制。

每个指标流可以使用 JSON 格式或 OpenTelemetry 0.7.0 格式。

指标流始终包括 Minimum、Maximum、Sample count 和 Sum 统计数据。您也可以选择包括其他统计数据，另外收费。有关更多信息，请参阅[可以流式传输的统计数据 \(p. 87\)](#)。

指标流定价基于指标更新的数量。对于用于指标流的传输流的 Kinesis Data Firehose，也会产生费用。有关更多信息，请参阅[Amazon CloudWatch 定价](#)。

主题

- [设置指标流 \(p. 84\)](#)
- [可以流式传输的统计数据 \(p. 87\)](#)
- [指标流操作和维护 \(p. 88\)](#)
- [使用 CloudWatch 指标监控您的指标流 \(p. 88\)](#)
- [CloudWatch 和 Kinesis Data Firehose 之间的信任关系 \(p. 89\)](#)
- [指标流输出格式 \(p. 90\)](#)
- [问题排查 \(p. 100\)](#)

设置指标流

按照以下部分中的步骤设置 CloudWatch 指标流。

创建指标流后，指标数据显示在目标位置所需的时间取决于对 Kinesis Data Firehose 传输流配置的缓冲设置。缓冲以最大负载大小或最长等待时间表示，以先达到者为准。如果缓冲设置为最小值（60 秒、1MB），则如果所选 CloudWatch 命名空间具有活动指标更新，则预期延迟将在 3 分钟内。

在 CloudWatch 指标流中，每分钟发送一次数据。数据可能无序到达最终目标位置。指定命名空间中的所有指标都在指标流中发送，但时间戳超过两小时的指标除外。

对于您流式传输的指标名称和命名空间的每个组合，该指标名称和命名空间的所有维度组合都将被流式传输。

要创建及管理指标流，您必须登录具有 CloudWatchFullAccess 策略和 iam:PassRole 权限的账户，或具有以下所列权限的账户：

- iam:PassRole
- cloudwatch:PutMetricStream

- `cloudwatch:DeleteMetricStream`
- `cloudwatch:GetMetricStream`
- `cloudwatch:ListMetricStreams`
- `cloudwatch:StartMetricStreams`
- `cloudwatch:StopMetricStreams`

如果您要让 CloudWatch 设置指标流所需的 IAM 角色，您还必须具有 `iam:CreateRole` 和 `iam:PutRolePolicy` 权限。

Important

具有 `cloudwatch:PutMetricStream` 的用户即使不具有 `cloudwatch:GetMetricData` 权限也可以访问正在流式传输的 CloudWatch 指标数据。

主题

- [设置要流式传输到 Amazon 服务的指标流 \(数据湖方案\) \(p. 85\)](#)
- [设置要流式传输到第三方解决方案的指标流 \(p. 87\)](#)

设置要流式传输到 Amazon 服务的指标流 (数据湖方案)

您可以使用 CloudWatch 控制台、Amazon CLI、Amazon CloudFormation 或 Amazon Cloud Development Kit (Amazon CDK) 设置指标流。

您用于指标流的 Kinesis Data Firehose 传输流必须位于指标流设置所在的同一账户和区域中。要实现跨区域功能，您可以将 Kinesis Data Firehose 传输流配置为流式传输到位于不同账户或不同区域中的最终目标位置。

CloudWatch 控制台

本部分介绍如何使用 CloudWatch 控制台设置要传输到另一 Amazon 服务的指标流。

在使用控制台设置指标流时，您可以选择 Quick S3 Setup (快速 S3 设置)。如果希望最终以 JSON 格式输出或由 Amazon Athena 查询，此方法非常有用。如果您希望最终格式为 Parquet 格式或 Optimized Row Columnar (ORC)，则应创建自己的 Kinesis Data Firehose 传输流，然后选择 Select an existing Firehose owned by your account (选择您账户拥有的现有 Firehose) 而不是选择 Quick S3 Setup (快速 S3 设置)。

设置要流式传输到 Amazon 服务的指标流

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)、Streams (流)。然后选择 Create metric stream (创建指标流)。
3. 选择要包括在指标流中的 CloudWatch 指标命名空间。
 - 若要将所有或大部分指标命名空间包括在指标流中，请选择 All metrics (所有指标)。

然后，如果要从流中排除某些指标命名空间，请选择 Exclude metric namespaces (排除指标命名空间)，然后选择要排除的命名空间。

- 若要在指标流中仅包括几个指标命名空间，请选择 Selected namespaces (已选命名空间)，然后选择要包括的命名空间。

Note

请仔细考虑是否要流式传输所有指标，因为流式传输的指标越多，您的指标流费用就越高。

4. 选择以下选项之一：

- 要快速设置，请选择 Quick S3 setup (快速 S3 设置)。CloudWatch 将创建所有必要的资源，包括 Kinesis Data Firehose 传输流和必要的 IAM 角色。此选项的默认格式为 JSON，但您可以在此程序的稍后步骤中更改格式。
- 若要使用已存在的 Kinesis Data Firehose 传输流，请选择 Select an existing Firehose owned by your account (选择您账户拥有的现有 Firehose)。Kinesis Data Firehose 传输流必须位于同一个账户中。此选项的默认格式为 OpenTelemetry，但您可以在此程序的稍后步骤中更改格式。

然后在 Select your Kinesis Data Firehose delivery stream (选择您的 Kinesis Data Firehose 传输流) 下选择要使用的 Kinesis Data Firehose 传输流。

5. (可选) 如果您使用的是 Quick S3 Setup (快速 S3 设置)，您可以选择 Select existing resources (选择现有资源) 以使用现有 S3 存储桶或现有 IAM 角色，而不是让 CloudWatch 为您创建新的存储桶或 IAM 角色。
6. (可选) 如果您使用的是 Select an existing Firehose owned by your account (选择您账户拥有的现有 Firehose)，您可以选择 Select existing service role (选择现有服务角色) 以使用现有 IAM 角色，而不是让 CloudWatch 为您创建新角色。
7. (可选) 要更改方案的默认输出格式，请选择 Change output format (更改输出格式)。支持的格式为 JSON 和 OpenTelemetry 0.7.0。
8. (可选) 要流式传输其中一些超出最小值、最大值、样本计数和总和的指标的其他统计数据，请选择 Add additional statistics (添加额外统计数据)。要么选择 Add recommended metrics (添加推荐的指标) 添加一些常用的统计数据，要么手动选择要针对其流式传输额外统计数据的命名空间和指标名称。接下来，选择要流式传输的额外统计数据。

然后，要选择另一组指标，以流式传输另一组额外统计数据，请选择 Add additional statistics (添加额外统计数据)。每个指标可以包含多达 20 个额外统计数据，一个指标流中有多达 100 个指标可以包含额外统计数据。

流式传输额外统计数据会产生更多费用。有关更多信息，请参阅[可以流式传输的统计数据 \(p. 87\)](#)。

有关额外统计数据的定义，请参阅[CloudWatch 统计数据定义 \(p. 138\)](#)。

9. (可选) 可以在 Metric stream name (指标流名称) 下自定义新指标流的名称。
10. 选择 Create metric filter (创建指标流)。

Amazon CLI 或 Amazon API

使用以下步骤创建要流式传输到另一 Amazon 服务的 CloudWatch 指标流。

使用 Amazon CLI 或 Amazon API 创建指标流

1. 如果您要流式传输到 Amazon S3，请先创建存储桶。有关更多信息，请参阅[创建存储桶](#)。
2. 创建 Kinesis Data Firehose 传输流。有关更多信息，请参阅[创建 Amazon Kinesis Data Firehose 传输流](#)。
3. 创建使 CloudWatch 能够向 Kinesis Data Firehose 传输流写入的 IAM 角色。有关该角色内容的更多信息，请参阅[CloudWatch 和 Kinesis Data Firehose 之间的信任关系 \(p. 89\)](#)。
4. 使用 `aws cloudwatch put-metric-stream` CLI 命令或 `PutMetricStream` API 来创建 CloudWatch 指标流。

Amazon CloudFormation

您可以使用 Amazon CloudFormation 设置指标流。有关更多信息，请参阅[AWS::CloudWatch::MetricStream](#)。

使用 Amazon CloudFormation 创建指标流

1. 如果您要流式传输到 Amazon S3，请先创建存储桶。有关更多信息，请参阅[创建存储桶](#)。

2. 创建 Kinesis Data Firehose 传输流。有关更多信息，请参阅[创建 Amazon Kinesis Data Firehose 传输流](#)。
3. 创建使 CloudWatch 能够向 Kinesis Data Firehose 传输流写入的 IAM 角色。有关该角色内容的更多信息，请参阅[CloudWatch 和 Kinesis Data Firehose 之间的信任关系 \(p. 89\)](#)。
4. 在 Amazon CloudFormation 中创建指标流。有关更多信息，请参阅[AWS::CloudWatch::MetricStream](#)。

Amazon Cloud Development Kit (Amazon CDK)

您可以使用 Amazon Cloud Development Kit (Amazon CDK) 设置指标流。

使用 Amazon CDK 创建指标流

1. 如果您要流式传输到 Amazon S3，请先创建存储桶。有关更多信息，请参阅[创建存储桶](#)。
2. 创建 Kinesis Data Firehose 传输流。有关更多信息，请参阅[创建 Amazon Kinesis Data Firehose 传输流](#)。
3. 创建使 CloudWatch 能够向 Kinesis Data Firehose 传输流写入的 IAM 角色。有关该角色内容的更多信息，请参阅[CloudWatch 和 Kinesis Data Firehose 之间的信任关系 \(p. 89\)](#)。
4. 创建指标流。指标流资源在 Amazon CDK 中作为名为 CfnMetricStream 的 Level 1 (L1) Construct 提供。有关更多信息，请参阅[使用 L1 Construct](#)。

设置要流式传输到第三方解决方案的指标流

要设置 CloudWatch 指标流以将指标流式传输到第三方解决方案，请按照该第三方提供商提供的说明进行操作。

可以流式传输的统计数据

指标流始终包括以下统计数据：Minimum、Maximum、Sample count 和 Sum。您还可以选择在指标流中包括以下其他统计数据。此选择基于每个指标。有关这些统计数据的更多信息，请参阅[CloudWatch 统计数据定义 \(p. 138\)](#)。

- 百分位数值，例如 p95 或 p99（适用于 JSON 或 OpenTelemetry 格式的流）
- 切尾均值（仅适用于 JSON 格式的流）
- 缩尾均值（仅适用于 JSON 格式的流）
- 切尾计数（仅适用于 JSON 格式的流）
- 切尾总和（仅适用于 JSON 格式的流）
- 百分位数排名（仅适用于 JSON 格式的流）
- 四分位数均值（仅适用于 JSON 格式的流）

流式传输额外统计数据会产生额外费用。流式传输某一特定指标的其中 1 个到 5 个额外统计数据，将作为一次额外指标更新来计费。此后，每增加一组（最多包含其中 5 个统计数据），将作为另一次指标更新来计费。

例如，假设对于某一指标，您正在流式传输以下 6 个额外统计数据：p95、p99、p99.9、切尾均值、缩尾均值和切尾总和。此指标的每次更新均作为三次指标更新计费：一个用于包含默认统计数据的指标更新，一个用于前 5 个额外统计数据，还有一个用于第 6 个额外统计数据。再添加最多 4 个额外统计数据，共 10 个，不会增加计费，但是第 11 个额外统计数据就会增加计费。

当您指定指标名称和命名空间组合，以流式传输额外统计数据时，该指标名称和命名空间的所有维度组合都将与这些额外统计数据一起流式传输。

CloudWatch 指标流发布了一个新指标，`TotalMetricUpdate`，它反映了指标更新加上由流式传输额外统计数据产生的额外指标更新的基本数量。有关更多信息，请参阅[使用 CloudWatch 指标监控您的指标流 \(p. 88\)](#)。

有关更多信息，请参阅 [Amazon CloudWatch 定价](#)。

Note

有些指标不支持百分位数。这些指标的百分位数统计数据将从流式传输中排除，因而不会产生指标流费用。这些不支持百分位数的统计数据的一个示例是 AWS/ECS 命名空间的一些指标。

仅当您配置的额外统计数据与流的筛选条件相匹配时，才会流式传输这些统计数据。例如，如果您创建的流仅将 EC2 和 RDS 包含在包含筛选条件中，然后您的统计数据配置列出 EC2 和 Lambda，则该流将包含 EC2 指标（其中包含额外统计数据）、RDS 指标（仅包含默认统计数据），但根本不包含 Lambda 统计数据。

指标流操作和维护

指标流始终处于两种状态之一，即正在运行或已停止状态。

- 正在运行 – 指标流正在正确运行。由于指标流的筛选条件，可能没有任何已流式传输到目标位置的指标数据。
- 已停止 – 指标流已被某人明确停止，且不是因错误而停止。停止指标流以暂时暂停流式传输数据而不删除指标流可能会很有帮助。

如果停止并重新启动指标流，则在指标流停止期间发布到 CloudWatch 的指标数据不会回填到该指标流。

如果更改指标流的输出格式，在某些情况下，您可能会看到少量指标数据以新旧两种格式写入到目标位置。为了避免这种情况，您可以使用与当前传输流相同的配置创建新的 Kinesis Data Firehose 传输流，然后更改为新的 Kinesis Data Firehose 传输流并同时更改输出格式。这样，具有不同输出格式的 Kinesis 记录将存储在 Amazon S3 中的单独对象中。您稍后可以将流量导向回原始 Kinesis Data Firehose 传输流，并删除另一个传递流。

查看、编辑、停止和启动指标流

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics（指标）、Streams（流）。

此时将显示流列表，并且 Status（状态）列显示每个流是处于正在运行还是已停止状态。

3. 要停止或启动指标流，请选择该流，然后选择 Stop（停止）或 Start（启动）。
4. 要查看有关指标流的详细信息，请选择该流，然后选择 View details（查看详细信息）。
5. 若要更改流的输出格式、筛选条件、目标 Kinesis Data Firehose 流或角色，请选择 Edit（编辑）并执行所需的更改。

如果您更改筛选条件，则在过渡期间指标数据中可能存在一些差异。

使用 CloudWatch 指标监控您的指标流

指标流会在 AWS/CloudWatch/MetricStreams 命名空间中发出关于自身运行状况和操作的 CloudWatch 指标。将发出以下指标。发出的这些指标都会存在具有 MetricStreamName 维度和没有维度两种情况。您可以使用没有维度的指标来查看所有指标流的已聚合指标。您可以使用具有 MetricStreamName 维度的指标以查看仅与该指标流有关的指标。

对于所有这些指标，均仅针对处于 Running（正在运行）状态的指标流发出数值。

指标	描述
MetricUpdate	发送到指标流的指标更新次数。如果在某个时间段内未流式传输有指标更新，则会为此指标发出值 0。 如果停止指标流，则将停止发出该指标，直到该指标流再次启动。 有效统计数据：Sum 单位：无
TotalMetricUpdate	计算方法为 MetricUpdate + 一个基于正在流式传输的额外统计数据的数字。 对于每个唯一的命名空间和指标名称组合，流式传输 1-5 个额外统计数据将向 TotalMetricUpdate 加 1，流式传输 6-10 个额外统计数据将向 TotalMetricUpdate 加 2，以此类推。 有效统计数据：Sum 单位：无
PublishErrorRate	在将数据放入 Kinesis Data Firehose 传输流时发生的不可恢复的错误数。如果在某个时间段内没有发生错误，则会为此指标发出值 0。 如果停止指标流，则将停止发出该指标，直到该指标流再次启动。 有效统计数据：Average，用于查看无法写入的指标更新速率。此值必须介于 0.0 到 1.0 之间。 单位：无

CloudWatch 和 Kinesis Data Firehose 之间的信任关系

Kinesis Data Firehose 传输流必须通过对 Kinesis Data Firehose 具有写入权限的 IAM 角色来信任 CloudWatch。这些权限可仅限为 CloudWatch 指标流使用的单个 Kinesis Data Firehose 传输流。IAM 角色必须信任 `streams.metrics.cloudwatch.amazonaws.com` 服务主体。

如果您使用 CloudWatch 控制台创建指标流，则可以让 CloudWatch 创建具有正确权限的角色。如果您使用其他方法创建指标流，或者希望创建 IAM 角色本身，则该角色必须包含以下权限策略和信任策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:firehose:region:account-id:deliverystream/*"
    }
  ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
    "Effect": "Allow",
    "Principal": {
      "Service": "streams.metrics.cloudwatch.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
  }
]
```

指标数据由 CloudWatch 代表拥有该指标流资源的源将其流式传输到目标 Kinesis Data Firehose 传输流。

指标流输出格式

CloudWatch 指标流中的数据可为 JSON 格式或 OpenTelemetry 格式。目前，支持的 OpenTelemetry 版本为 0.7.0。

目录

- [JSON 格式 \(p. 90\)](#)
 - [我应该对 JSON 输出格式使用哪个 Amazon Glue 架构？ \(p. 91\)](#)
- [OpenTelemetry 0.7.0 format \(p. 92\)](#)
 - [转换为 OpenTelemetry 0.7.0 格式 \(p. 96\)](#)
 - [如何解析 OpenTelemetry 0.7.0 消息 \(p. 97\)](#)

JSON 格式

在使用 JSON 格式的 CloudWatch 指标流中，每条 Kinesis Data Firehose 记录都包含多个由换行符 (\n) 分隔的 JSON 对象。每个对象包含单个指标的单一数据点。

所使用的 JSON 格式与 Amazon Glue 和 Amazon Athena 完全兼容。如果您的 Kinesis Data Firehose 传输流和 Amazon Glue 表格的格式均正确，则格式可以自动转换为 Parquet 格式或 Optimized Row Columnar (ORC) 格式，然后再存储在 S3 中。有关转换格式的更多信息，请参阅在 [Kinesis Data Firehose 中转换输入记录格式](#)。有关 Amazon Glue 的正确格式的更多信息，请参阅 [我应该对 JSON 输出格式使用哪个 Amazon Glue 架构？ \(p. 91\)](#)。

在 JSON 格式中，unit 的有效值与 MetricDatum API 结构中 unit 的值相同。有关更多信息，请参阅 [MetricDatum](#)。timestamp 字段的值以纪元毫秒为单位，例如 1616004674229。

以下为格式示例。在本示例中，JSON 所用格式的目的是便于查看，但实际上，整个格式应位于一行上。

```
{
  "metric_stream_name": "MyMetricStream",
  "account_id": "1234567890",
  "region": "us-east-1",
  "namespace": "AWS/EC2",
  "metric_name": "DiskWriteOps",
  "dimensions": {
    "InstanceId": "i-123456789012"
  },
  "timestamp": 1611929698000,
  "value": {
    "count": 3.0,
    "sum": 20.0,
    "max": 18.0,
    "min": 0.0,
    "p99": 17.56,
    "p99.9": 17.8764,
    "TM(25%;75%)": 16.43
  },
}
```

```
    "unit": "Seconds"  
  }
```

我应该对 JSON 输出格式使用哪个 Amazon Glue 架构？

以下是 Amazon Glue 表的 `StorageDescriptor` 的 JSON 表示（之后由 Kinesis Data Firehose 使用）示例。有关 `StorageDescriptor` 的更多信息，请参阅 [StorageDescriptor](#)。

```
{  
  "Columns": [  
    {  
      "Name": "metric_stream_name",  
      "Type": "string"  
    },  
    {  
      "Name": "account_id",  
      "Type": "string"  
    },  
    {  
      "Name": "region",  
      "Type": "string"  
    },  
    {  
      "Name": "namespace",  
      "Type": "string"  
    },  
    {  
      "Name": "metric_name",  
      "Type": "string"  
    },  
    {  
      "Name": "timestamp",  
      "Type": "timestamp"  
    },  
    {  
      "Name": "dimensions",  
      "Type": "map<string,string>"  
    },  
    {  
      "Name": "value",  
      "Type":  
"struct<min:double,max:double,count:double,sum:double,p99:double,p99.9:double>"  
    },  
    {  
      "Name": "unit",  
      "Type": "string"  
    }  
  ],  
  "Location": "s3://my-s3-bucket/",  
  "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",  
  "OutputFormat": "org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat",  
  "SerdeInfo": {  
    "SerializationLibrary": "org.apache.hive.hcatalog.data.JsonSerDe"  
  },  
  "Parameters": {  
    "classification": "json"  
  }  
}
```

上面的示例适用于以 JSON 格式在 Amazon S3 上写入的数据。将以下字段中的值替换为指定的值，以便以 Parquet 格式或 Optimized Row Columnar (ORC) 格式存储数据。

- Parquet :

- inputFormat: org.apache.hadoop.hive ql.io.parquet.MapredParquetInputFormat
- outputFormat: org.apache.hadoop.hive ql.io.parquet.MapredParquetOutputFormat
- SerDeInfo.serializationLib: org.apache.hadoop.hive ql.io.parquet.serde.ParquetHiveSerDe
- parameters.classification: parquet
- ORC :
 - inputFormat: org.apache.hadoop.hive ql.io.orc.OrcInputFormat
 - outputFormat: org.apache.hadoop.hive ql.io.orc.OrcOutputFormat
 - SerDeInfo.serializationLib: org.apache.hadoop.hive ql.io.orc.OrcSerde
 - parameters.classification: orc

OpenTelemetry 0.7.0 format

OpenTelemetry 是工具、API 和软件开发工具包的集合。您可以使用它来测量、生成、收集及导出遥测数据（指标、日志和跟踪）以进行分析。OpenTelemetry 属于云原生计算基金会。有关更多信息，请参阅 [OpenTelemetry](#)。

有关完整 OpenTelemetry 0.7.0 规范的信息，请参阅 [v0.7.0 版本](#)。

Note

虽然指标流已正式发布，但 OpenTelemetry 格式 0.7.0 尚未正式发布。即使 OpenTelemetry 格式版本 1.0 正式发布，指标流也将继续提供 OpenTelemetry 格式 0.7.0。

一条 Kinesis 记录可以包含一个或多个 ExportMetricsServiceRequest OpenTelemetry 数据结构。每个数据结构都以一个带有指示记录长度（字节）的 UnsignedVarInt32 的标头开头。每个 ExportMetricsServiceRequest 可同时包含来自多个指标的数据。

以下是 ExportMetricsServiceRequest OpenTelemetry 数据结构形式的消息的字符串表示。OpenTelemetry 会序列化 Google Protocol Buffers 二进制协议，此协议人类不可读。

```
resource_metrics {
  resource {
    attributes {
      key: "cloud.provider"
      value {
        string_value: "aws"
      }
    }
    attributes {
      key: "cloud.account.id"
      value {
        string_value: "2345678901"
      }
    }
    attributes {
      key: "cloud.region"
      value {
        string_value: "us-east-1"
      }
    }
    attributes {
      key: "aws.exporter.arn"
      value {
        string_value: "arn:aws:cloudwatch:us-east-1:123456789012:metric-stream/MyMetricStream"
      }
    }
  }
  instrumentation_library_metrics {
```



```
metrics {
  name: "amazonaws.com/AWS/DynamoDB/ConsumedReadCapacityUnits"
  unit: "1"
  double_summary {
    data_points {
      labels {
        key: "Namespace"
        value: "AWS/DynamoDB"
      }
      labels {
        key: "MetricName"
        value: "ConsumedReadCapacityUnits"
      }
      labels {
        key: "TableName"
        value: "MyTable"
      }
      start_time_unix_nano: 1604948400000000000
      time_unix_nano: 1604948460000000000
      count: 1
      sum: 1.0
      quantile_values {
        quantile: 0.0
        value: 1.0
      }
      quantile_values {
        quantile: 0.95
        value: 1.0
      }
      quantile_values {
        quantile: 0.99
        value: 1.0
      }
      quantile_values {
        quantile: 1.0
        value: 1.0
      }
    }
    data_points {
      labels {
        key: "Namespace"
        value: "AWS/DynamoDB"
      }
      labels {
        key: "MetricName"
        value: "ConsumedReadCapacityUnits"
      }
      labels {
        key: "TableName"
        value: "MyTable"
      }
      start_time_unix_nano: 1604948460000000000
      time_unix_nano: 1604948520000000000
      count: 2
      sum: 5.0
      quantile_values {
        quantile: 0.0
        value: 2.0
      }
      quantile_values {
        quantile: 1.0
        value: 3.0
      }
    }
  }
}
```

```
}  
}
```

用于序列化 OpenTelemetry 指标数据的顶级对象

`ExportMetricsServiceRequest` 是用于序列化 OpenTelemetry 导出器负载的顶级包装器。它包含一个或多个 `ResourceMetrics`。

```
message ExportMetricsServiceRequest {  
  // An array of ResourceMetrics.  
  // For data coming from a single resource this array will typically contain one  
  // element. Intermediary nodes (such as OpenTelemetry Collector) that receive  
  // data from multiple origins typically batch the data before forwarding further and  
  // in that case this array will contain multiple elements.  
  repeated opentelemetry.proto.metrics.v1.ResourceMetrics resource_metrics = 1;  
}
```

`ResourceMetrics` 是表示 `MetricData` 对象的顶级对象。

```
// A collection of InstrumentationLibraryMetrics from a Resource.  
message ResourceMetrics {  
  // The resource for the metrics in this message.  
  // If this field is not set then no resource info is known.  
  opentelemetry.proto.resource.v1.Resource resource = 1;  
  
  // A list of metrics that originate from a resource.  
  repeated InstrumentationLibraryMetrics instrumentation_library_metrics = 2;  
}
```

资源对象

`Resource` 对象是一个值对象，其中包含有关生成指标的资源的一些信息。对于由 Amazon 创建的指标，数据结构包含与指标相关的资源的 Amazon Resource Name (ARN)，例如 EC2 实例或 S3 存储桶。

`Resource` 对象包含名为 `attributes` 的属性，其会存储键值对列表。

- `cloud.account.id` 包含账户 ID
- `cloud.region` 包含区域
- `aws.exporter.arn` 包含指标流 ARN
- `cloud.provider` 始终为 `aws`。

```
// Resource information.  
message Resource {  
  // Set of labels that describe the resource.  
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 1;  
  
  // dropped_attributes_count is the number of dropped attributes. If the value is 0,  
  // no attributes were dropped.  
  uint32 dropped_attributes_count = 2;  
}
```

`InstrumentationLibraryMetrics` 对象

将不会填写 `instrumentation_library` 字段。我们将仅填写正在导出的指标字段。

```
// A collection of Metrics produced by an InstrumentationLibrary.  
message InstrumentationLibraryMetrics {  
  // The instrumentation library information for the metrics in this message.  
  // If this field is not set then no library info is known.
```

```
opentelemetry.proto.common.v1.InstrumentationLibrary instrumentation_library = 1;
// A list of metrics that originate from an instrumentation library.
repeated Metric metrics = 2;
}
```

指标对象

指标对象包含一个 DoubleSummary 数据字段，该字段包含一个 DoubleSummaryDataPoint 列表。

```
message Metric {
  // name of the metric, including its DNS name prefix. It must be unique.
  string name = 1;

  // description of the metric, which can be used in documentation.
  string description = 2;

  // unit in which the metric value is reported. Follows the format
  // described by http://unitsofmeasure.org/ucum.html.
  string unit = 3;

  oneof data {
    IntGauge int_gauge = 4;
    DoubleGauge double_gauge = 5;
    IntSum int_sum = 6;
    DoubleSum double_sum = 7;
    IntHistogram int_histogram = 8;
    DoubleHistogram double_histogram = 9;
    DoubleSummary double_summary = 11;
  }
}

message DoubleSummary {
  repeated DoubleSummaryDataPoint data_points = 1;
}
```

MetricDescriptor 对象

MetricDescriptor 对象包含元数据。有关更多信息，请参阅 GitHub 上的 [metrics.proto](#)。

对于指标流，MetricDescriptor 包含以下内容：

- name 将为 `amazonaws.com/metric_namespace/metric_name`
- description 将为空。
- unit 将通过将指标基准单位映射为计量单位统一代码的变体（区分大小写）来填充。有关更多信息，请参阅 [转换为 OpenTelemetry 0.7.0 格式 \(p. 96\)](#) 和 [计量单位统一代码](#)。
- type 将为 SUMMARY。

DoubleSummaryDataPoint 对象

DoubleSummaryDataPoint 对象包含 DoubleSummary 指标时间序列中单个数据点的值。

```
// DoubleSummaryDataPoint is a single data point in a timeseries that describes the
// time-varying values of a Summary metric.
message DoubleSummaryDataPoint {
  // The set of labels that uniquely identify this timeseries.
  repeated opentelemetry.proto.common.v1.StringKeyValue labels = 1;

  // start_time_unix_nano is the last time when the aggregation value was reset
  // to "zero". For some metric types this is ignored, see data types for more
  // details.
  //
```

```
// The aggregation value is over the time interval (start_time_unix_nano,
// time_unix_nano].
//
// Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
// 1970.
//
// Value of 0 indicates that the timestamp is unspecified. In that case the
// timestamp may be decided by the backend.
fixed64 start_time_unix_nano = 2;

// time_unix_nano is the moment when this aggregation value was reported.
//
// Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
// 1970.
fixed64 time_unix_nano = 3;

// count is the number of values in the population. Must be non-negative.
fixed64 count = 4;

// sum of the values in the population. If count is zero then this field
// must be zero.
double sum = 5;

// Represents the value at a given quantile of a distribution.
//
// To record Min and Max values following conventions are used:
// - The 1.0 quantile is equivalent to the maximum value observed.
// - The 0.0 quantile is equivalent to the minimum value observed.
message ValueAtQuantile {
  // The quantile of a distribution. Must be in the interval
  // [0.0, 1.0].
  double quantile = 1;

  // The value at the given quantile of a distribution.
  double value = 2;
}

// (Optional) list of values at different quantiles of the distribution calculated
// from the current snapshot. The quantiles must be strictly increasing.
repeated ValueAtQuantile quantile_values = 6;
}
```

有关更多信息，请参阅 [转换为 OpenTelemetry 0.7.0 格式 \(p. 96\)](#)。

转换为 OpenTelemetry 0.7.0 格式

CloudWatch 会执行一些转换，将 CloudWatch 数据转换为 OpenTelemetry 格式。

转换命名空间、指标名称和维度

这些属性是在映射中编码的键值对。

- 其中一对包含指标的命名空间
- 另一对包含指标的名称
- 对于每个维度，CloudWatch 都存储以下键值对：`metricDatum.Dimensions[i].Name`，`metricDatum.Dimensions[i].Value`

转换平均值、总和、样本计数、最小值和最大值

摘要数据点使 CloudWatch 能够使用一个数据点导出所有这些统计数据。

- `startTimeUnixNano` 包含 CloudWatch `startTime`

- timeUnixNano 包含 CloudWatch endTime
- sum 包含总和统计数据。
- count 包含样本数统计数据。
- quantile_values 包含两个 valueAtQuantile.value 对象：
 - valueAtQuantile.quantile = 0.0 与 valueAtQuantile.value = *Min value*
 - valueAtQuantile.quantile = 0.99 与 valueAtQuantile.value = *p99 value*
 - valueAtQuantile.quantile = 0.999 与 valueAtQuantile.value = *p99.9 value*
 - valueAtQuantile.quantile = 1.0 与 valueAtQuantile.value = *Max value*

使用该指标流的资源可以按总和/样本数来计算平均值统计数据。

转换单位

CloudWatch 单位映射到计量单位统一代码区分大小写的变体，如下表所示。有关更多信息，请参阅[计量单位统一代码](#)。

CloudWatch	OpenTelemetry
秒	s
秒	s
Microsecond	us
毫秒	ms
字节	By
千字节	kBy
兆字节	MBy
千兆字节	GBy
千吉字节	TBy
Bits	bit
千位	kbit
兆位	MBit
千兆位	GBit
太位	Tbit
百分比	%
计数	{Count}
无	1

由斜线组合而成的单位通过同时对斜线前后两个单位应用 OpenTelemetry 转换来进行映射。例如，字节/秒映射为 By/s。

如何解析 OpenTelemetry 0.7.0 消息

本部分提供了有助于您开始解析 OpenTelemetry 0.7.0 的信息。

首先，您应获得特定于语言的绑定，这样您能够以您的首选语言解析 OpenTelemetry 0.7.0 消息。

获取特定于语言的绑定

- 根据您的首选语言选择以下步骤。
 - 若要使用 Java，请将以下 Maven 依赖项添加到您的 Java 项目中：[OpenTelemetry Java >> 0.14.1](#)。
 - 若要使用任何其他语言，请按照下列步骤操作：
 - a. 检查[生成类](#)中的列表，确保您的语言受支持。
 - b. 按照[下载协议缓冲区](#)中的步骤安装 Protobuf 编译器。
 - c. 在 [v0.7.0 版本](#) 中下载 OpenTelemetry 0.7.0 ProtoBuf 定义。
 - d. 确认您位于下载的 OpenTelemetry 0.7.0 ProtoBuf 定义的根文件夹中。创建一个 src 文件夹，然后运行命令以生成特定于语言的绑定。有关更多信息，请参阅[生成类](#)。

以下示例展示了如何生成 Javascript 绑定。

```
protoc --proto_path=./ --js_out=import_style=commonjs,binary:src \
opentelemetry/proto/common/v1/common.proto \
opentelemetry/proto/resource/v1/resource.proto \
opentelemetry/proto/metrics/v1/metrics.proto \
opentelemetry/proto/collector/metrics/v1/metrics_service.proto
```

以下部分包括使用特定于语言的绑定的示例，您可以使用前面的说明来构建这些绑定。

Java

```
package com.example;

import io.opentelemetry.proto.collector.metrics.v1.ExportMetricsServiceRequest;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class MyOpenTelemetryParser {

    public List<ExportMetricsServiceRequest> parse(InputStream inputStream) throws
    IOException {
        List<ExportMetricsServiceRequest> result = new ArrayList<>();

        ExportMetricsServiceRequest request;
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
        records, each of them starting with a header with an
        UnsignedVarInt32 indicating the record length in bytes:
        -----
        |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
        -----
        */
        while ((request = ExportMetricsServiceRequest.parseDelimitedFrom(inputStream)) !=
        null) {
            // Do whatever we want with the parsed message
            result.add(request);
        }

        return result;
    }
}
```

Javascript

本示例假定具有所生成的绑定的根文件夹为 ./

函数 parseRecord 的数据参数可以是以下类型之一：

- Uint8Array，此类型最佳
- Buffer，在节点下最佳
- Array.*number*，8 位整数

```
const pb = require('google-protobuf')
const pbMetrics =
  require('./opentelemetry/proto/collector/metrics/v1/metrics_service_pb')

function parseRecord(data) {
  const result = []

  // Loop until we've read all the data from the buffer
  while (data.length) {
    /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
       records, each of them starting with a header with an
       UnsignedVarInt32 indicating the record length in bytes:
       -----
       |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
       -----
    */
    const reader = new pb.BinaryReader(data)
    const messageLength = reader.decoder_.readUnsignedVarint32()
    const messageFrom = reader.decoder_.cursor_
    const messageTo = messageFrom + messageLength

    // Extract the current `ExportMetricsServiceRequest` message to parse
    const message = data.subarray(messageFrom, messageTo)

    // Parse the current message using the ProtoBuf library
    const parsed =
      pbMetrics.ExportMetricsServiceRequest.deserializeBinary(message)

    // Do whatever we want with the parsed message
    result.push(parsed.toObject())

    // Shrink the remaining buffer, removing the already parsed data
    data = data.subarray(messageTo)
  }

  return result
}
```

Python

您必须自行读取 var-int 分隔符或使用内部方法 `_VarintBytes(size)` 和 `_DecodeVarint32(buffer, position)`。它们会返回刚好位于缓冲区中大小字节之后的位置。读取端构建一个新的缓冲区，该缓冲区仅限于读取消息的字节。

```
size = my_metric.ByteSize()
f.write(_VarintBytes(size))
f.write(my_metric.SerializeToString())
msg_len, new_pos = _DecodeVarint32(buf, 0)
msg_buf = buf[new_pos:new_pos+msg_len]
request = metrics_service_pb.ExportMetricsServiceRequest()
request.ParseFromString(msg_buf)
```


Go

使用 `Buffer.DecodeMessage()`。

C#

使用 `CodedInputStream`。此类可以读取分隔了大小的消息。

C++

`google/protobuf/util/delimited_message_util.h` 中描述的函数可以读取分隔了大小的消息。

其他语言

有关使用其他语言，请参阅[下载协议缓冲区](#)。

在实现解析器时，请注意，一条 Kinesis 记录可以包含多个 `ExportMetricsServiceRequest` 协议缓冲区消息，每个消息都以具有指示记录长度（字节）的 `UnsignedVarInt32` 的标头开头。

问题排查

如果您在最终目标位置看不到指标数据，请检查以下各项：

- 检查指标流是否处于正在运行状态。有关如何使用 CloudWatch 控制台执行此操作的步骤，请参阅 [指标流操作和维护 \(p. 88\)](#)。
- 无法流式传输发布时间超过两个小时的指标。要确定是否流式传输特定指标，请在 CloudWatch 控制台中绘制指标图表，然后检查最后一个可见数据点的存在时间。如果发布时间超过两个小时，那么指标流将不会获取此指标。
- 检查指标流发出的指标。在 CloudWatch 控制台中，在 Metrics（指标）下，查看 `MetricUpdate`、`TotalMetricUpdate` 和 `PublishErrorRate` 指标的 `AWS/CloudWatch/MetricStreams` 命名空间。
- 如果 `PublishErrorRate` 指标较高，请确认 Kinesis Data Firehose 传输流使用的目标存在，并且指标流配置中指定的 IAM 角色授予了 CloudWatch 服务主要权限以写入其中。有关更多信息，请参阅[CloudWatch 和 Kinesis Data Firehose 之间的信任关系 \(p. 89\)](#)。
- 检查 Kinesis Data Firehose 传输流是否有权限写入最终目标位置。
- 在 Kinesis Data Firehose 控制台中，查看用于指标流的 Kinesis Data Firehose 传输流，并检查 Monitoring（监控）选项卡以查看 Kinesis Data Firehose 传输流是否在接收数据。
- 确认您已为 Kinesis Data Firehose 传输流配置了正确的详细信息。
- 检查 Kinesis Data Firehose 传输流写入的最终目标位置的任何可用日志或指标。
- 若要获取更多详细信息，请在 Kinesis Data Firehose 传输流上启用 CloudWatch Logs 错误日志记录。有关更多信息，请参阅[使用 CloudWatch Logs 监控 Kinesis Data Firehose](#)。

查看可用指标

指标首先按命名空间进行分组，然后按各命名空间内的各种维度组合进行分组。例如，您可以查看所有 EC2 指标、按实例分组的 EC2 指标或按 Auto Scaling 组分组的 EC2 指标。

只有您使用的 Amazon 服务会将指标发送到 Amazon CloudWatch。

有关向 CloudWatch 发送指标的 Amazon 服务列表，请参阅[发布 CloudWatch 指标的 Amazon 服务 \(p. 768\)](#)。在此页面中，您还可以查看这些服务中的每个服务所发布的指标和维度。

Note

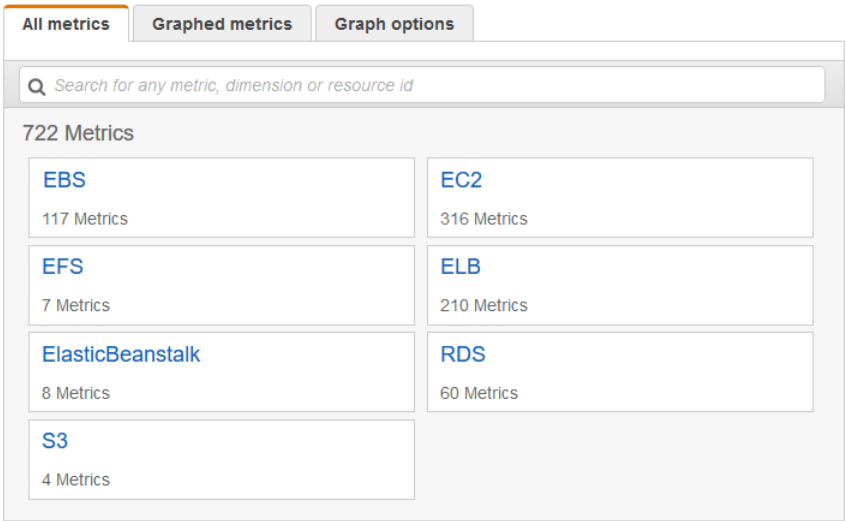
控制台中不会显示在过去两周内没有任何新数据点的指标。当您在控制台的全局 All metrics（全部指标）选项卡的搜索框中键入指标名称或维度名称时，它们也不会显示，并且 `list-metrics` 命令的结

果中不会返回它们。检索这些指标的最佳方法是使用 Amazon CLI 中的 `get-metric-data` 或者 `get-metric-statistics` 命令。
如果要查看的旧指标有一个具有相似维度的当前指标，则可以查看该当前相似指标，然后选择 Source (源) 选项卡，并将指标名称和维度字段更改为所需的指标，此外将时间范围更改为报告指标的时间。

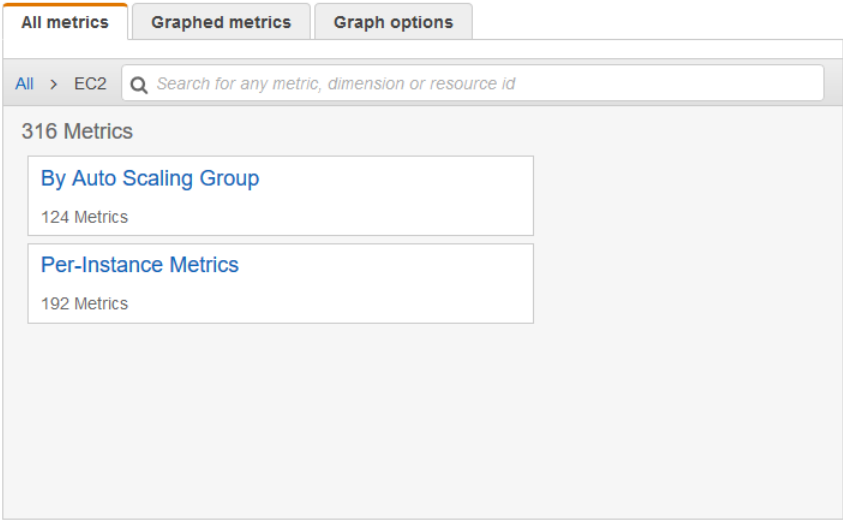
以下步骤可帮助您浏览指标命名空间以查找和查看指标。您还可以使用目标搜索词搜索指标。有关更多信息，请参阅[搜索可用指标 \(p. 103\)](#)。

使用控制台按命名空间和维度查看可用指标

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)，然后选择 All metrics (所有指标)。
3. 选择指标命名空间 (例如 EC2)。

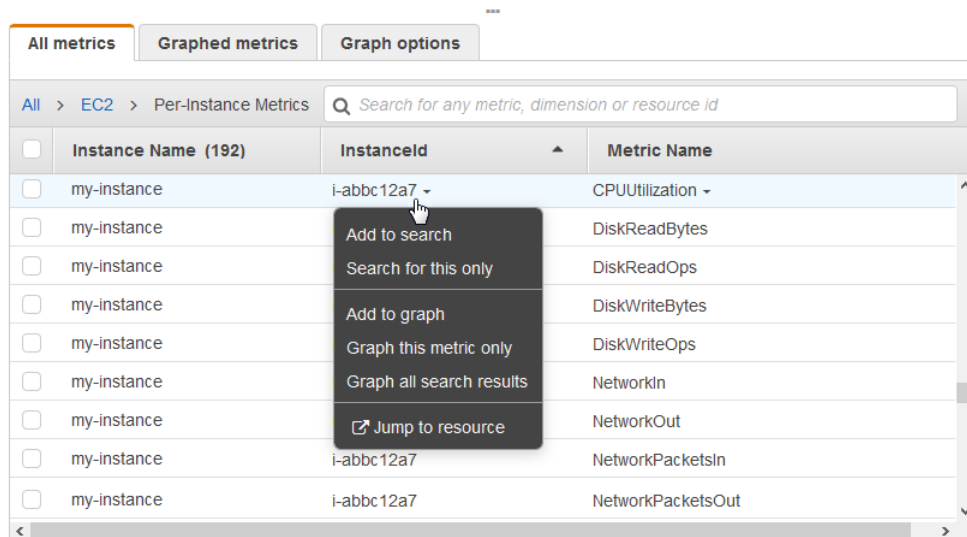


4. 选择指标维度 (例如 Per-Instance Metrics (每个实例的指标))。



5. All metrics 选项卡显示命名空间中该维度的所有指标。您可执行以下操作：
 - a. 要对表进行排序，请使用列标题。
 - b. 要为指标绘制图表，请选中该指标旁的复选框。要选择所有指标，请选中表的标题行中的复选框。

- c. 要按资源进行筛选，请选择资源 ID，然后选择 Add to search。
- d. 要按指标进行筛选，请选择指标名称，然后选择 Add to search。



6. (可选) 要将此图表添加到 CloudWatch 控制面板，请选择 Actions (操作)，然后选择 Add to dashboard (添加到控制面板)。

使用 Amazon CLI 按命名空间、维度或指标查看可用指标

使用 `list-metrics` 命令列出 CloudWatch 指标。有关发布指标的所有服务的命名空间、指标和维度的列表，请参阅 [发布 CloudWatch 指标的 Amazon 服务 \(p. 768\)](#)。

以下示例指定 AWS/EC2 命名空间以查看 Amazon EC2 的所有指标。

```
aws cloudwatch list-metrics --namespace AWS/EC2
```

下面是示例输出。

```
{
  "Metrics" : [
    ...
    {
      "Namespace": "AWS/EC2",
      "Dimensions": [
        {
          "Name": "InstanceId",
          "Value": "i-1234567890abcdef0"
        }
      ],
      "MetricName": "NetworkOut"
    },
    {
      "Namespace": "AWS/EC2",
      "Dimensions": [
        {
          "Name": "InstanceId",
          "Value": "i-1234567890abcdef0"
        }
      ],
      "MetricName": "CPUUtilization"
    }
  ]
}
```

```
    },  
    {  
      "Namespace": "AWS/EC2",  
      "Dimensions": [  
        {  
          "Name": "InstanceId",  
          "Value": "i-1234567890abcdef0"  
        }  
      ],  
      "MetricName": "NetworkIn"  
    },  
    ...  
  ]  
}
```

列出指定资源的所有可用指标

以下示例指定 AWS/EC2 命名空间和 InstanceId 维度以仅查看指定实例的结果。

```
aws cloudwatch list-metrics --namespace AWS/EC2 --dimensions  
Name=InstanceId,Value=i-1234567890abcdef0
```

列出所有资源的指标

以下示例指定 AWS/EC2 命名空间和指标名称以仅查看指定指标的结果。

```
aws cloudwatch list-metrics --namespace AWS/EC2 --metric-name CPUUtilization
```

搜索可用指标

您可以使用目标搜索词，对您账户中的所有指标进行搜索。之后会返回在命名空间、指标名称或维度内具有匹配结果的指标。

Note

控制台中不会显示在过去两周内没有任何新数据点的指标。当您在控制台的所有指标 (全部指标) 选项卡的搜索框中键入指标名称或维度名称时，它们也不会显示，并且 `list-metrics` 命令的结果中不会返回它们。检索这些指标的最佳方法是使用 Amazon CLI 中的 `get-metric-data` 或者 `get-metric-statistics` 命令。

在 CloudWatch 中搜索可用指标

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 在 All metrics (所有指标) 选项卡上的搜索字段中，输入搜索词 (例如，指标名称、命名空间、维度名称或值，或者资源名称)。这将为显示带有匹配此搜索词的指标的所有命名空间。

例如，如果搜索 **volume**，则将显示其名称中包含该搜索词的指标的命名空间。

有关搜索的更多信息，请参阅 [在图表中使用搜索表达式 \(p. 131\)](#)。

4. 要绘制所有搜索结果的图表，请选择 Graph search (图表搜索)

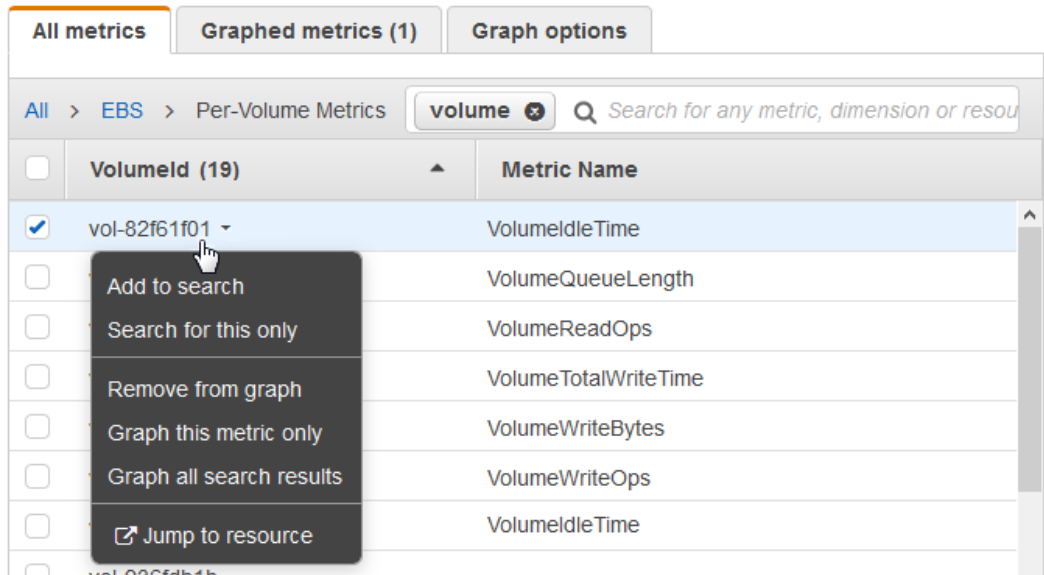
或

选择命名空间以查看该命名空间中的指标。然后，您可执行以下操作：

- a. 要为一个或多个指标绘制图表，请选中每个指标旁边的复选框。要选择所有指标，请选中表的标题行中的复选框。

- b. 要细化您的搜索，请将鼠标指针悬停在某个指标名称上，然后选择 Add to search (添加到搜索) 或 Search for this only (仅为此搜索)。
- c. 要在控制台中查看某个资源，请选择资源 ID，然后选择 Jump to resource (跳转到资源)。
- d. 要查看指标的帮助，请选择指标名称，然后选择 What is this?。

所选指标会显示在图表上。



5. (可选) 选择搜索栏中的一个按钮以编辑搜索词的该部分。

绘制指标的图表

使用 CloudWatch 控制台绘制其他 Amazon 服务生成的指标数据的图表。这样可以更高效地查看服务上的指标活动。以下程序介绍了如何在 CloudWatch 中绘制指标图表。

目录

- [绘制指标的图表 \(p. 104\)](#)
- [使用动态标签 \(p. 107\)](#)
- [修改图表的时间范围或时区格式 \(p. 109\)](#)
- [放大折线图或堆叠面积图 \(p. 110\)](#)
- [修改图表的 Y 轴 \(p. 112\)](#)
- [从图表上的指标创建告警 \(p. 112\)](#)

绘制指标的图表

您可以使用 CloudWatch 控制台选择指标并创建指标数据图表。

CloudWatch 支持指标的以下统计数据：Average、Minimum、Maximum、Sum 和 SampleCount。有关更多信息，请参阅[统计数据 \(p. 5\)](#)。

您可以在不同的详细程度下查看您的数据。例如，您可以选择一分钟视图，这在您排查问题时会有非常有用。或者，选择不太详细的一小时视图。这在您查看更大的时间范围（例如 3 天）来了解一段时间内的趋势时会很有用。有关更多信息，请参阅[时间段 \(p. 5\)](#)。

创建图表

绘制指标图表

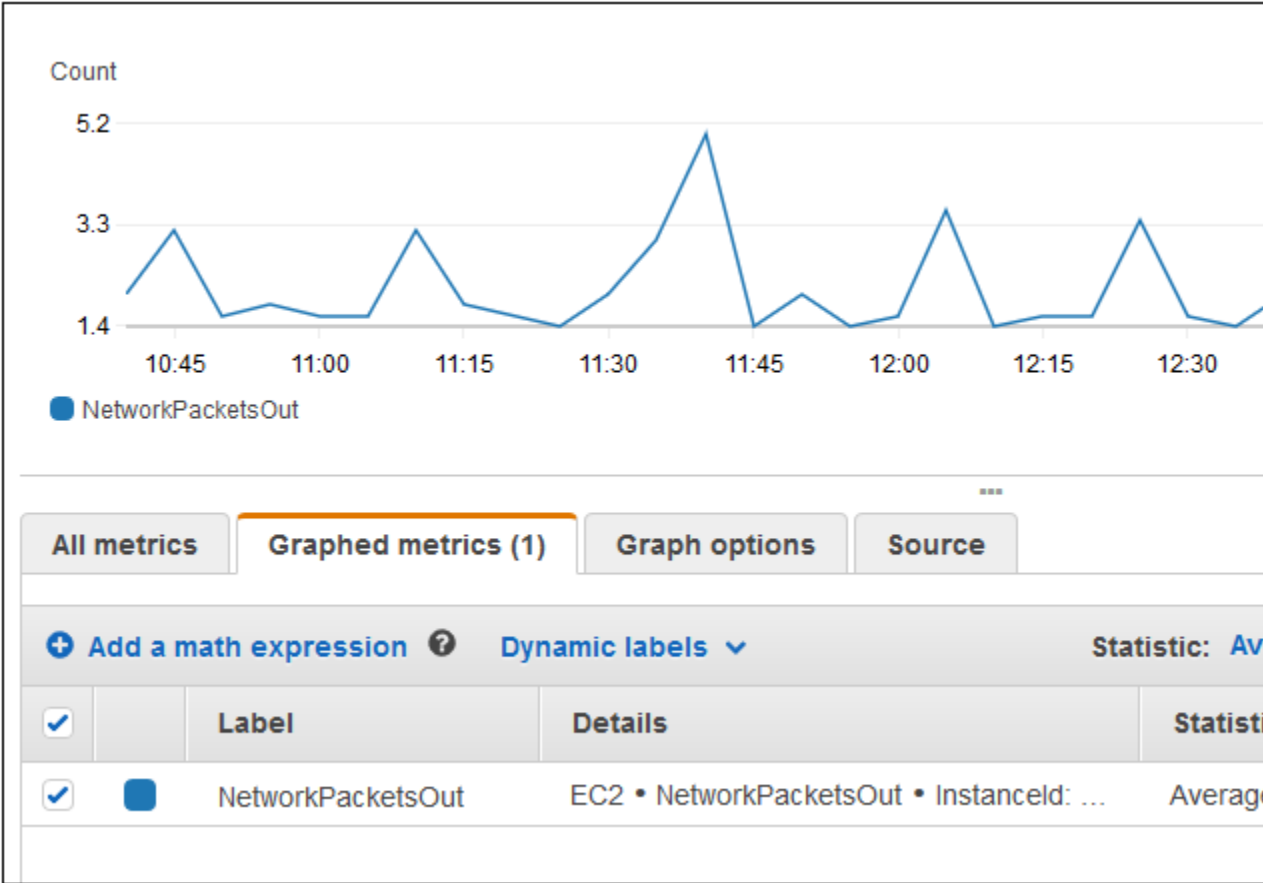
1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Metrics (指标)。
3. 在全部指标选项卡上的搜索字段中，输入搜索词 (例如，指标名称或资源名称)，然后按 Enter。

例如，如果您搜索 CPUUtilization 指标，则将显示具有该指标的命名空间和维度。

4. 选择某个搜索结果以查看指标。
5. 要为一个或多个指标绘制图表，请选中每个指标旁边的复选框。要选择所有指标，请选中表的标题行中的复选框。
6. 选择 View graphed metrics (查看已绘制图表指标)。
7. (可选) 若要更改图表中使用的统计数据，请选择指标名称旁边 Statistic (统计数据) 列中的新统计数据。

有关 CloudWatch 统计数据的更多信息，请参阅 [CloudWatch 统计数据定义 \(p. 138\)](#)。有关 pxx 百分位数统计数据的更多信息，请参阅 [百分位数 \(p. 6\)](#)。

8. (可选) 若要更改图表类型，请选择 Graph options (图表选项)。然后，您可以从折线图、堆叠面积图、条形图、饼形图或数字中选择。
9. (可选) 要添加显示指标的预期值的异常检测范围，请选择指标旁边的操作下的异常检测图标。



CloudWatch 使用指标最多两周的最新历史数据来计算预期值的模型。然后将此预期值范围以条带形式显示在图表上。CloudWatch 会在指标下新增一行，以显示异常检测范围数学表达式，并标记为

ANOMALY_DETECTION_BAND。最近的历史数据存在时，您可以立即看到一个预览异常检测范围，该范围是模型生成的异常检测范围的近似值。实际异常检测范围可能需要长达 15 分钟才能显示出来。

预设情况下，CloudWatch 使用范围阈值的默认值 2 创建预期值范围的上限和下限。要更改此数字，请更改范围的 Details (详细信息) 下的公式末尾的值。

- (可选) 选择 Edit model (编辑模型) 来更改计算异常检测模型的方式。您可以排除过去和未来的时间段，不在模型计算训练中使用。从训练数据中排除系统中断、部署和假日等异常事件至关重要。您还可以指定要用于夏令时更改模型的时区。

有关更多信息，请参阅[使用 CloudWatch 异常检测 \(p. 113\)](#)。

要在图表中隐藏模型，请在 ANOMALY_DETECTION_BAND 函数所在行中取消选中选中标记或者选择 x 图标。要完全删除模型，请选择 Edit model (编辑模型)，然后选择 Delete model (删除模型)。

10. (可选) 在选择要绘制图表的指标时，为每个指标指定一个在图表图例上显示的动态标签。动态标签显示有关指标的统计数据，并在刷新控制面板或图表时自动进行更新。要添加动态标签，请选择 Graphed metrics (已绘制图表指标)，然后选择 Dynamic labels (动态标签)。

默认情况下，添加到标签的动态值显示在标签开头。然后，您可以选择指标的标签值以编辑标签。有关更多信息，请参阅[使用动态标签 \(p. 107\)](#)。

11. 要查看有关要为其绘制图表的指标的更多信息，请将鼠标指针暂停在图例上。
12. 横向注释可帮助图表用户更高效地查看指标的峰值何时达到特定级别以及指标是否在预定义的范围内。要添加横向注释，请选择 Graph options (图表选项)，然后选择 Add horizontal annotation (添加横向注释)：
 - a. 对于 Label (标签)，输入注释的标签。
 - b. 对于 Value (值)，输入显示横向注释的指标值。
 - c. 对于 Fill，指定是否将填充阴影用于此注释。例如，为要填充的相应区域选择 Above 或 Below。如果您指定 Between，则另一个 Value 字段将出现，并且将填充两个值之间的图表区域。
 - d. 对于 Axis，指定 Value 中的数字是否在图表包含多个指标的情况下表示与左侧 Y 轴或右侧 Y 轴关联的指标。

您可以通过在注释的左列中选择颜色方块来更改注释的填充色。

重复这些步骤可向同一个图表添加多个横向注释。

要隐藏注释，请清除该注释的左列中的复选框。

要删除注释，请在 Actions 列中选择 x。

13. 要获取图表的 URL，请依次选择 Actions 和 Share。复制要保存或共享的 URL。
14. 要将图表添加到控制面板，请依次选择 Actions 和 Add to dashboard。

更新图表

更新图表

1. 要更改图表的名称，请选择铅笔图标。
2. 要更改时间范围，请选择某个预定义的值或选择 custom。有关更多信息，请参阅[修改图表的时间范围或时区格式 \(p. 109\)](#)。
3. 要更改统计数据，请选择 Graphed metrics (已绘制图表指标) 选项卡。选择列标题或单个值，然后选择某个统计数据或预定义百分位数，或指定自定义百分位数 (例如 **p95.45**)。
4. 要更改时间段，请选择 Graphed metrics (已绘制图表指标) 选项卡。选择列标题或单个值，然后选择其他值。

5. 要添加横向注释，请选择 Graph options (图表选项)，然后选择 Add horizontal annotation (添加横向注释)：
 - a. 对于 Label (标签)，输入注释的标签。
 - b. 对于 Value (值)，输入显示横向注释的指标值。
 - c. 对于 Fill，指定是否将填充阴影用于此注释。例如，为要填充的相应区域选择 Above 或 Below。如果您指定 Between，则另一个 Value 字段将出现，并且将填充两个值之间的图表区域。
 - d. 对于 Axis，指定 Value 中的数字是否在图表包含多个指标的情况下表示与左侧 Y 轴或右侧 Y 轴关联的指标。

您可以通过在注释的左列中选择颜色方块来更改注释的填充色。

重复这些步骤可向同一个图表添加多个横向注释。

要隐藏注释，请清除该注释的左列中的复选框。

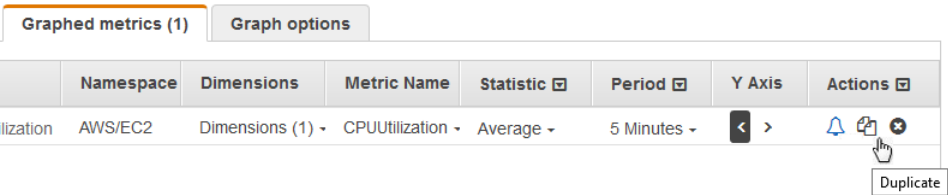
要删除注释，请在 Actions 列中选择 x。

6. 要更改刷新间隔，请选择 Refresh options (刷新选项)，然后选择 Auto refresh (自动刷新) 或者选择 1 Minute (1 分钟)、2 Minutes (2 分钟)、5 Minutes (5 分钟) 或 15 Minutes (15 分钟)。

复制指标

复制指标

1. 选择 Graphed metrics (已绘制图表指标) 选项卡。
2. 对于 Actions，选择 Duplicate 图标。



3. 根据需要更新重复指标。

使用动态标签

您可以在图表中使用动态标签。动态标签在选定指标的标签中添加动态更新的值。您可以向标签添加多种值，如下表所示。

在标签中显示的动态值来自于当前在图表上显示的时间范围。在刷新控制面板或图表时，将自动更新标签的动态部分。

如果将动态标签与搜索表达式一起使用，则动态标签适用于搜索返回的每个指标。

您可以使用 CloudWatch 控制台向标签添加动态值、编辑标签、更改动态值在标签列中的位置以及自定义其他内容。

动态标签

在动态标签中，您可以使用与指标属性相关的以下值：

动态标签实时值	描述
<code>\${AVG}</code>	当前在图表中显示的时间范围内的值的平均值。
<code>\${DATAPOINT_COUNT}</code>	图表中当前所示时间范围内的数据点的数量。
<code>\${FIRST}</code>	图表中当前所示时间范围内的最早指标值。
<code>\${FIRST_LAST_RANGE}</code>	图表中当前所示最早数据点和最新数据点的指标值之间的差值。
<code>\${FIRST_LAST_TIME_RANGE}</code>	图表中当前所示最早数据点和最新数据点之间的绝对时间范围。
<code>\${FIRST_TIME}</code>	图表中当前所示时间范围内的最早数据点的时间戳。
<code>\${FIRST_TIME_RELATIVE}</code>	此刻与图表中当前所示时间范围内的最早数据点的时间戳之间的绝对时间差。
<code>\${LABEL}</code>	指标的默认标签的表示形式。
<code>\${LAST}</code>	图表中当前所示时间范围内的指标值的最新值。
<code>\${LAST_TIME}</code>	图表中当前所示时间范围内的最新数据点的时间戳。
<code>\${LAST_TIME_RELATIVE}</code>	此刻与图表中当前所示时间范围内的最新数据点的时间戳之间的绝对时间差。
<code>\${MAX}</code>	当前在图表中显示的时间范围内的值的最大值。
<code>\${MAX_TIME}</code>	图表中当前所示数据点中具有最高指标值的数据点的时间戳。
<code>\${MAX_TIME_RELATIVE}</code>	此刻与图表中当前所示数据点中具有最高值的数据点的时间戳之间的绝对时间差。
<code>\${MIN}</code>	当前在图表中显示的时间范围内的值的最小值。
<code>\${MIN_MAX_RANGE}</code>	图表中当前所示数据点中具有最高指标值和最低指标值的数据点之间的指标值之差。
<code>\${MIN_MAX_TIME_RANGE}</code>	图表中当前所示数据点中具有最高指标值和最低指标值的数据点之间的绝对时间范围。
<code>\${MIN_TIME}</code>	图表中当前所示数据点中具有最低指标值的数据点的时间戳。
<code>\${MIN_TIME_RELATIVE}</code>	此刻与图表中当前所示数据点中具有最低值的数据点的时间戳之间的绝对时间差。
<code>\${PROP('AccountId')}</code>	指标的 Amazon 账户 ID。
<code>\${PROP('Dim.<i>dimension_name</i>')}</code>	指定维度的值。将 <i>dimension_name</i> 替换为您的维度的区分大小写名称。
<code>\${PROP('MetricName')}</code>	指标的名称。
<code>\${PROP('Namespace')}</code>	指标的命名空间。
<code>\${PROP('Period')}</code>	指标的时间段 (以秒为单位) 。
<code>\${PROP('Region')}</code>	发布指标的 Amazon 区域。
<code>\${PROP('Stat')}</code>	正在用于绘制图表的指标统计数据。
<code>\${SUM}</code>	当前在图表中显示的时间范围内的值的总和。

例如，假设您使用 `SEARCH(' {AWS/Lambda, FunctionName} Errors ', 'Sum', 300)` 搜索表达式，它查找每个 Lambda 函数的 `Errors`。如果将标签设置为 `[max: ${MAX} Errors for Function Name ${LABEL}]`，则每个指标的标签为 `[max: number Errors for Function Name Name]`。

您最多可以在标签中添加五个动态值。您只能在每个标签中使用一次 `${LABEL}` 占位符。

修改图表的时间范围或时区格式

本节介绍了如何在 CloudWatch 指标图表上修改日期、时间和时区格式。它还介绍了如何放大图表以应用特定的时间范围。有关创建图表的信息，请参阅 [绘制指标的图表](#)。

设置相对时间范围

New interface

指定图表的相对时间范围

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics（指标），然后选择 All metrics（所有指标）。在屏幕右上角，您可以选择从 1 小时到 1 周的某个预定义的时间范围（1h（1 小时）、3h（3 小时）、12h（12 小时）、1d（1 天）、3d（3 天）或者 1w（1 周））。或者，您也可以选择 Custom（自定义）来设置自己的时间范围。
3. 选择 Custom（自定义），然后选择框的左上角的 Relative（相对）选项卡。您可以以 Minutes（分钟）、Hours（小时）、Days（天）、Weeks（周）、Months（月）为单位指定时间范围。
4. 在指定时间范围后，选择 Apply（应用）。

Original interface

指定图表的相对时间范围

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics（指标），然后选择 All metrics（所有指标）。在屏幕右上角，您可以选择从 1 小时到 1 周的某个预定义的时间范围（1h（1 小时）、3h（3 小时）、12h（12 小时）、1d（1 天）、3d（3 天）或者 1w（1 周））。或者，您也可以选择 custom（自定义）来设置自己的时间范围。
3. 选择 custom（自定义），然后选择框的左上角的 Relative（相对）。您可以以 Minutes（分钟）、Hours（小时）、Days（天）、Weeks（周）或 Months（月）为单位指定时间范围。

设置绝对时间范围

New interface

指定图表的绝对时间范围

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics（指标），然后选择 All metrics（所有指标）。在屏幕右上角，您可以选择从 1 小时到 1 周的某个预定义的时间范围（1h（1 小时）、3h（3 小时）、12h（12 小时）、1d（1 天）、3d（3 天）或者 1w（1 周））。或者，您也可以选择 Custom（自定义）来设置自己的时间范围。
3. 选择 Custom（自定义），然后选择框的左上角的 Absolute（绝对）选项卡。使用日历选取器或文本字段框指定时间范围。
4. 在指定时间范围后，选择 Apply（应用）。

Original interface

指定图表的绝对时间范围

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)，然后选择 All metrics (所有指标)。在屏幕右上角，您可以选择从 1 小时到 1 周的某个预定义的时间范围 (1h (1 小时)、3h (3 小时)、12h (12 小时)、1d (1 天)、3d (3 天) 或者 1w (1 周))。或者，您也可以选择 custom (自定义) 来设置自己的时间范围。
3. 选择 custom (自定义)，然后选择框的左上角的 Absolute (绝对)。使用日历选取器或文本字段框指定时间范围。
4. 在指定时间范围后，选择 Apply (应用)。

设置时区格式

New interface

指定图表的时区

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)，然后选择 All metrics (所有指标)。在屏幕右上角，您可以选择从 1 小时到 1 周的某个预定义的时间范围 (1h (1 小时)、3h (3 小时)、12h (12 小时)、1d (1 天)、3d (3 天) 或者 1w (1 周))。或者，您也可以选择 Custom (自定义) 来设置自己的时间范围。
3. 选择 Custom (自定义)，然后选择框右上角的下拉菜单。您可以将时区更改为 UTC (协调世界时) 或 Local time zone (本地时区)。
4. 在更改之后，选择 Apply (应用)。

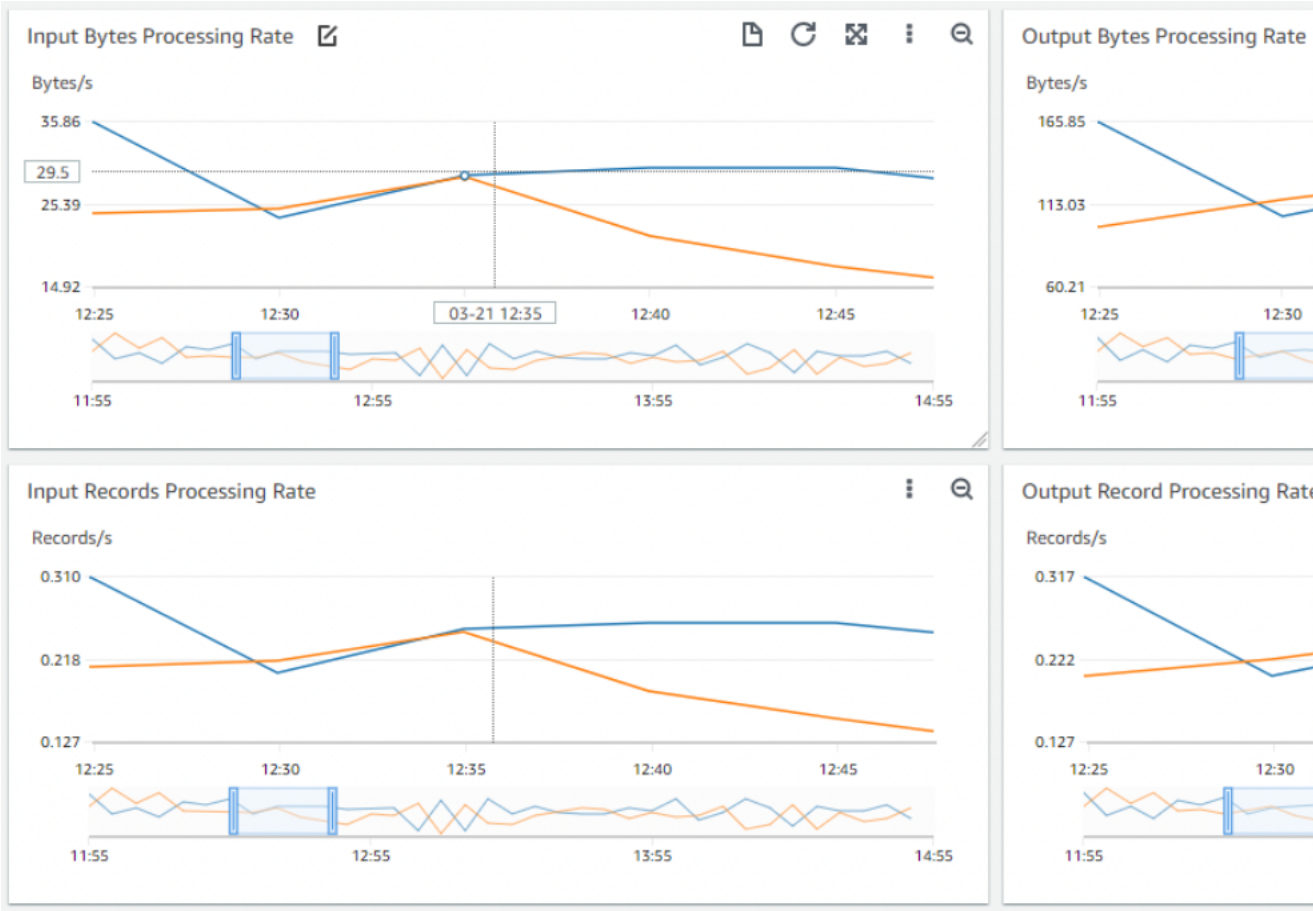
Original interface

指定图表的时区

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)，然后选择 All metrics (所有指标)。在屏幕右上角，您可以选择从 1 小时到 1 周的某个预定义的时间范围 (1h (1 小时)、3h (3 小时)、12h (12 小时)、1d (1 天)、3d (3 天) 或者 1w (1 周))。或者，您也可以选择 custom (自定义) 来设置自己的时间范围。
3. 选择 custom (自定义)，然后选择框右上角的下拉菜单。您可以将时区更改为 UTC (协调世界时) 或 Local timezone (本地时区)。

放大折线图或堆叠面积图

在 CloudWatch 控制台中，您可以使用缩微贴图缩放功能来重点查看折线图和堆叠面积图的各个部分，而无需在放大和缩小视图之间进行切换。例如，您可以使用缩微贴图缩放功能来重点查看折线图上的峰值，以便将该峰值与同一时间线的控制面板中的其他指标进行比较。本节内容中的程序描述如何使用缩放功能。



在上图中，缩放功能突出了与输入字节处理速率相关的折线图的峰值，它同时还在控制面板中显示其他折线图，突出同一时间线的各个部分。

New interface

放大图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)，然后选择 All metrics (所有指标)。
3. 选择 Browse (浏览)，然后选择要生成图表的一个或多个指标。
4. 选择 Options (选项)，然后选择 Widget type (小组件类型) 下方的 Line (折线图)。
5. 选择并拖动要突出的图表区域，然后释放拖动对象。
6. 要重置缩放，选择 Reset zoom (重置缩放) 图标，该图标看起来像放大镜里面包含减号 (-) 符号。

Original interface

放大图表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)，然后选择 All metrics (所有指标)。
3. 选择 All metrics (所有指标)，然后选择要生成图表的指标。
4. 选择 Graph options (图表选项)。在 Widget type (小组件类型) 下方，选择 Line (折线图)。

- 5. 选择并拖动要突出的图表区域，然后释放拖动对象。
- 6. 要重置缩放，选择 Reset zoom (重置缩放) 图标，该图标看起来像放大镜里面包含减号 (-) 符号。

Tip

如果您已创建包含折线图或堆叠面积图的控制面板，可以转至控制面板并开始使用缩放功能。

修改图表的 Y 轴

您可以在图表上设置 Y 轴的自定义范围，帮助您更好地查看数据。例如，您可以将 CPUUtilization 图表上的范围更改为 100%，这样，您就能轻松看到 CPU 使用率是低（绘制的线条靠近图表底部）还是高（绘制的线条靠近图表顶部）。

您可以在图表的两个不同的 Y 轴之间切换。这在图表包含单位不同或者值范围差异很大的指标时非常有用。

修改图表上的 Y 轴

- 1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
- 2. 在导航窗格中，请选择指标。
- 3. 选择一个指标命名空间（例如 EC2），然后选择一个指标维度（例如 Per-Instance Metrics (每个实例的指标)）。
- 4. All metrics 选项卡显示此命名空间中该维度的所有指标。要为指标绘制图表，请选中该指标旁的复选框。
- 5. 在 Graph options (绘制图表选项) 选项卡上，指定 Left Y Axis (左侧 Y 轴) 的 Min (最小值) 和 Max (最大值)。Min (最小值) 的值不能大于 Max (最大值) 的值。

All metricsGraphed metrics (1)Graph options

Left Y Axis

Limits

Min

0

Max

100

Right Y Axis

Limits

Min

Auto

Max

Auto

- 6. 要创建第二个 Y 轴，请指定 Right Y Axis (右侧 Y 轴) 的 Min (最小值) 和 Max (最大值)。
- 7. 要在两个 Y 轴之间切换，请选择 Graphed metrics (已绘制图表指标) 选项卡。对于 Y Axis，请选择 Left Y Axis 或 Right Y Axis。

Graphed metrics (1)Graph options

	Namespace	Dimensions	Metric Name	Statistic	Period	Y Axis	Actions
Utilization	AWS/EC2	Dimensions (1)	CPUUtilization	Average	5 Minutes	< > Right Y Axis	🔔 📄 ⚙️

从图表上的指标创建告警

您可以绘制指标的图表，然后从图表上的指标创建警报，这样做的优点是可为您填充很多警报字段。

从图表上的指标创建警报

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 选择一个指标命名空间（例如 EC2），然后选择一个指标维度（例如 Per-Instance Metrics（每个实例的指标））。
4. All metrics 选项卡显示此命名空间中该维度的所有指标。要为指标绘制图表，请选中该指标旁的复选框。
5. 要为指标创建警报，请选择 Graphed metrics（已绘制图表指标）选项卡。对于 Actions，选择警报图标。



6. 在条件下，选择静态或异常检测，以指定对警报使用静态阈值还是异常检测模型。

根据您的选择，输入警报条件的其余数据。

7. 选择其他配置。对于触发警报的数据点数，指定必须有多少个评估期（数据点）处于 ALARM 状态才能触发警报。如果此处的两个值匹配，则会创建一个告警；如果多个连续评估期违例，该告警将变为 ALARM（告警）状态。

要创建“M（最大为 N）”告警，为第一个值指定的数字应小于为第二个值指定的数字。有关更多信息，请参阅 [评估告警 \(p. 152\)](#)。

8. 对于缺失数据处理，选择在缺失某些数据点时的警报行为。有关更多信息，请参阅 [配置 CloudWatch 告警处理缺失数据的方式 \(p. 153\)](#)。
9. 选择 Next（下一步）。
10. 在通知下面，选择一个在警报处于 ALARM、OK 或 INSUFFICIENT_DATA 状态时通知的 SNS 主题。

要使告警为相同告警状态或不同告警状态发送多个通知，请选择 Add notification（添加通知）。

要让警报不发送通知，请选择删除。

11. 要让警报执行 Auto Scaling 或 EC2 操作，请选择相应的按钮，然后选择警报状态和要执行的操作。
12. 在完成后，选择下一步。
13. 输入警报的名称和说明。名称只能包含 ASCII 字符。然后选择下一步。
14. 在 Preview and create 下面，确认具有所需的信息和条件，然后选择 Create alarm。

使用 CloudWatch 异常检测

为指标启用异常检测时，CloudWatch 将应用统计算法和机器学习算法。这些算法只需最少的用户干预，即可持续分析系统和应用程序的指标，确定正常基线和表面异常。

这些算法会生成异常检测模型。该模型生成表示正常指标行为的预期值范围。

您可以通过两种方式使用预期值模型：

- 根据指标的预期值创建异常检测警报。这些类型的警报没有用于确定警报状态的静态阈值。相反，它们将根据异常检测模型将指标的值与预期值进行比较。

您可以选择当指标值高于预期值范围和/或低于预期值范围时是否触发警报。

有关更多信息，请参阅[根据异常检测创建 CloudWatch 告警 \(p. 161\)](#)。

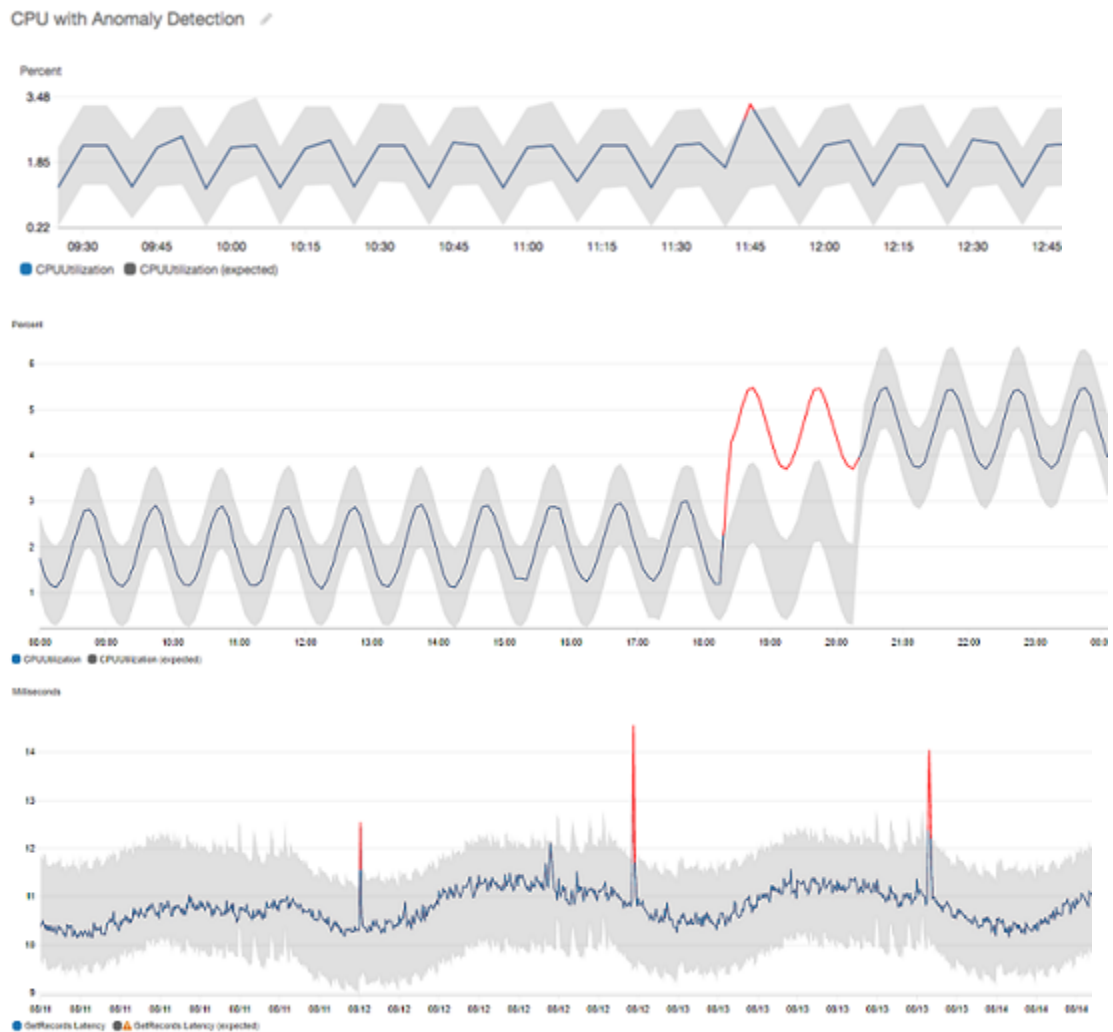
- 查看指标数据的图表时，将预期值叠加到图表上作为范围。这样可以清晰直观地看出图中的哪些值不在正常范围内。有关更多信息，请参阅[创建图表 \(p. 105\)](#)。

您可以使用 Amazon Web Services Management Console、Amazon CLI、Amazon CloudFormation 或 Amazon SDK 启用异常检测。您可以对由 Amazon 出售的指标以及自定义指标启用异常检测。

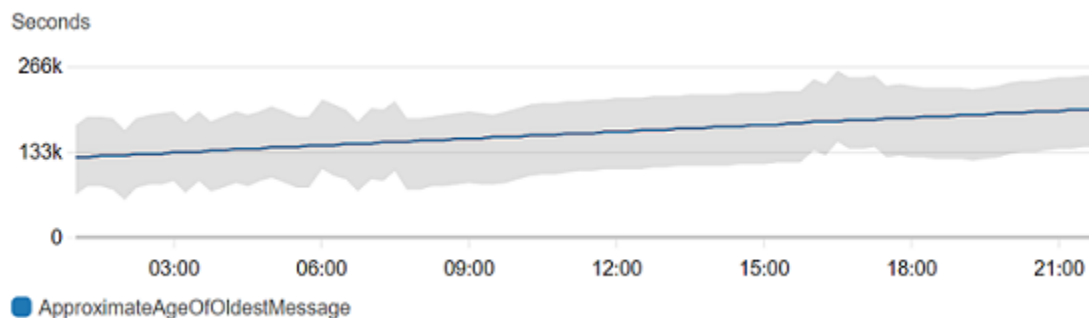
您还可以通过将 `GetMetricData` API 请求与 `ANOMALY_DETECTION_BAND` 指标数学函数结合使用来检索模型范围的上限值和下限值。有关更多信息，请参阅[GetMetricData](#)。

在具有异常检测的图表中，预期的值范围显示为灰色范围。如果指标的实际值超出此范围，则在此期间将显示为红色。

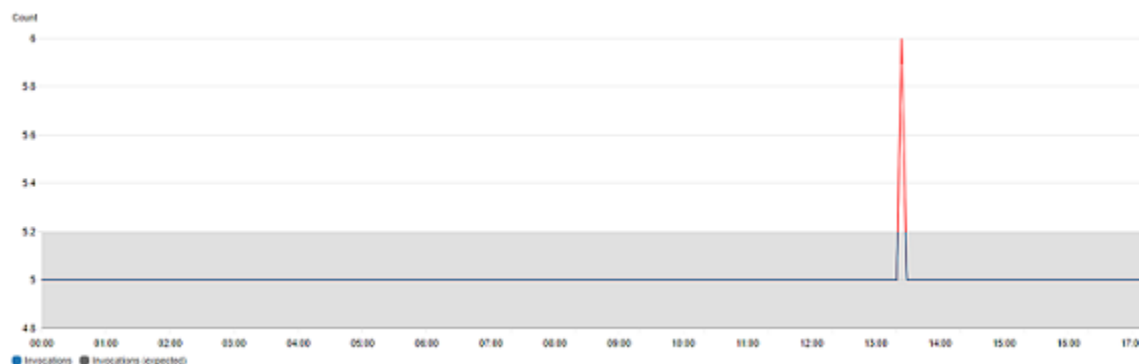
异常检测算法将指标的季节性变化和趋势变化考虑在内。季节性变化可以是每小时、每天或每周，如以下示例所示。



较长期的趋势可能是向下或向上。



异常检测也适用于具有平面模式的指标。



CloudWatch 异常检测的工作原理

在为指标启用异常检测后，CloudWatch 会将机器学习算法应用于指标的过去数据，以创建指标的预期值模型。该模型评估指标的趋势以及每小时、每日和每周模式。算法训练最多两周的指标数据，但即使指标没有完整的两周数据，您也可以为指标启用异常检测。

您需要为异常检测阈值指定一个值，CloudWatch 使用该值和该模型确定指标值的“正常”范围。异常检测阈值的值越高，生成的“正常”值范围越大。

机器学习模型特定于指标和统计数据。例如，如果您使用 AVG 统计数据为指标启用异常检测，则模型特定于 AVG 统计数据。

当 CloudWatch 为许多 Amazon 服务中的常见指标创建模型时，它可确保检测范围不会超出逻辑值。例如，不能为负的统计数据检测范围永远不会到零以下，百分比指标的检测范围将保持在 0 到 100 之间。

创建模型后，CloudWatch 异常检测会持续评估模型并对其进行调整，以确保模型尽可能准确。这包括重新训练模型，以便在指标值随时间推移而变化或有突然变化时进行调整，还包括用于改进季节性、峰值或稀疏指标模型的预测器。

在为指标启用异常检测后，您可以选择排除指标的指定时间段，使其不用于训练模型。这样，您就可以排除部署或其他不寻常的事件，使其不用于训练模型，从而确保创建最精确的模型。

对警报使用异常检测模型会使您的 Amazon 账户产生费用。有关更多信息，请参阅 [Amazon CloudWatch 定价](#)。

指标数学异常检测

指标数学异常检测是一项您可以用于针对指标数学表达式的输出创建异常检测告警的功能。您可以使用这些表达式来创建能可视化异常检测范围的图表。该功能支持基本的算术函数、比较和逻辑运算符以及大多数其他函数。有关不支持的函数的信息，请参阅 [Amazon CloudWatch 用户指南](#) 中的使用指标数学。

您可以根据指标数学表达式创建异常检测模型，创建方式类似于创建异常检测模型。在 CloudWatch 控制台中，您可以将异常检测应用于指标数学表达式，并选择异常检测作为这些表达式的阈值类型。

Note

仅可在最新版本的指标用户界面中启用和编辑指标数学异常检测。在新版本指标用户界面中根据指标数学表达式创建异常检测器时，您可以在旧版本中查看这些检测器，但不能对它们进行编辑。

有关如何为异常检测和指标数学创建告警和模型的信息，请参阅以下部分：

- [根据异常检测创建 CloudWatch 告警](#)
- [根据指标数学表达式创建 CloudWatch 告警](#)

您还可以将 CloudWatch API 与 `PutAnomalyDetector`、`DeleteAnomalyDetector` 和 `DescribeAnomalyDetectors` 结合起来使用，根据指标数学表达式创建、删除及发现异常检测模型。有关这些 API 操作的信息，请参阅 Amazon CloudWatch API 参考中的以下部分。

- [PutAnomalyDetector](#)
- [DeleteAnomalyDetector](#)
- [DescribeAnomalyDetectors](#)

有关异常检测告警定价方式的信息，请参阅 [Amazon CloudWatch 定价](#)。

使用指标数学

通过指标数学可以查询多个 CloudWatch 指标，可以使用数学表达式基于这些指标创建新的时间序列。您可以在 CloudWatch 控制台上直观显示生成的时间序列，并将其添加到控制面板中。以 Amazon Lambda 指标为例，您可以将 `Errors` 指标除以 `Invocations` 指标来获得错误率。然后，将生成的时间序列添加到 CloudWatch 控制面板上的图表中。

您也可以使用 `GetMetricData` API 操作以编程方式执行指标数学。有关更多信息，请参阅 [GetMetricData](#)。

向 CloudWatch 图表中添加数学表达式

您可以向 CloudWatch 控制面板上的图表中添加数学表达式。每个图表限制为使用最多 500 个指标和表达式，因此仅当图表具有 499 个或更少的指标时，您才可以添加数学表达式。即使并非所有指标都显示在图表上，这一点也适用。

向图表中添加数学表达式

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 创建或编辑图表。图表中至少需要一个指标。
3. 选择 Graphed metrics (已绘制图表指标)。
4. 选择 Math expression (数学表达式)、Start with empty expression (从空表达式开始)。将为表达式显示一个新行。
5. 在 Details (详细信息) 列下方的新行中，输入数学表达式。“指标数学语法和函数”部分中的表列出了可以在表达式中使用的函数。

要使用一个指标或另一个表达式的结果作为此表达式的公式的一部分，请使用 Id 列中显示的值：例如，`m1+m2` 或 `e1-MIN(e1)`。

您可以更改 Id 的值。它可以包括数字、字母和下划线，并且必须以小写字母开头。将 Id 的值更改为更有意义的名称也可以使图表更易于理解；例如，从 m1 和 m2 更改为 errors 和 requests。

Tip

选择 Math Expression (数学表达式) 旁边的向下箭头可查看受支持的函数列表，您可以在创建表达式时使用这些函数。

6. 对于表达式的 Label (标签) 列，输入一个描述表达式正在计算的内容的名称。

如果某个表达式的结果是一组时间序列，则这些时间序列将在图表上单独的行中以不同的颜色显示。图表的下方是图表中的每个行的图例。对于生成多个时间序列的单个表达式，这些时间序列的图例文字的格式为 **Expression-Label Metric-Label**。例如，如果图表包含一个具有标签 Errors 的指标和一个具有标签 Filled With 0: 的表达式 FILL(METRICS(), 0)，则图例中的一行将为 Filled With 0: Errors。要使图例仅显示原始指标标签，请将 **Expression-Label** 设置为空。

当一个表达式在图表上生成了一组时间序列时，您无法更改用于这些时间序列的颜色。

7. 在添加所需的表达式后，您可以通过隐藏某些原始指标来简化图表。要隐藏某个指标或表达式，请清除 Id 字段左侧的复选框。

指标数学语法和函数

以下各部分解释可用于指标数学的函数。所有函数都必须用大写字母编写（例如 AVG），所有指标和数学表达式的 Id 字段都必须以小写字母开头。

任何数学表达式的最终结果都必须是单个时间序列或一组时间序列。某些函数会生成标量数字。您可以在一个更大的函数中使用这些函数，从而最终生成一个时间序列。例如，采用单个时间序列的 AVG 会生成标量数字，因此它不能是最终的表达式结果。但您可以在函数 m1-AVG(m1) 中使用它，以显示每个单独数据点与时间序列中平均值之差的时间序列。

数据类型缩写

某些函数仅对某些类型的数据有效。在函数表中使用以下列表中的缩写来代表每个函数支持的数据类型：

- S 代表标量数字，例如 2、-5 或 50.25。
- TS 是时间序列（单个 CloudWatch 指标随时间变化的一系列值）：例如，实例 i-1234567890abcdef0 在过去 3 天的 CPUUtilization 指标
- TS [] 是一个时间序列数组，例如多个指标的时间序列

METRICS() 函数

METRICS() 函数将返回请求中的所有指标。数学表达式不包括在内。

您可以在一个更大的表达式中使用 METRICS()，从而最终生成单个时间序列或一组时间序列。例如，表达式 SUM(METRICS()) 将返回作为所有绘成图表的指标值的总和的时间序列 (TS)。METRICS()/100 将返回一组时间序列，其中的每个时间序列都显示其中一个指标的各个数据点除以 100 的结果。

您可以将 METRICS() 函数与一个字符串一起使用，仅返回在其 Id 字段中包含该字符串的绘成图表的指标。例如，表达式 SUM(METRICS("errors")) 一个是所有在其 Id 字段中具有“errors”的绘成图表的指标值的总和的时间序列。您还可以使用 SUM([METRICS("4xx"), METRICS("5xx")]) 来匹配多个字符串。

基本算术函数

下表列出了受支持的基本算术函数。时间序列中缺少的值被视为 0。如果数据点的值导致函数试图除以零，则会丢弃该数据点。

运算	Arguments	示例
算术运算符: + - * / ^	S, S S, TS TS, TS S, TS[] TS, TS[]	PERIOD(m1)/60 5 * m1 m1 - m2 SUM(100/[m1, m2]) AVG(METRICS()) METRICS()*100
一元减法 -	S TS TS[]	-5*m1 -m1 SUM(-[m1, m2])

比较运算符和逻辑运算符

您可以将比较运算符和逻辑运算符与一对时间序列或一对单标量值结合使用。在将比较运算符与一对时间序列结合使用时，运算符将返回一个时间序列，其中每个数据点为 0 (false) 或 1 (true)。如果在一对标量值上使用比较运算符，则将返回一个单标量值 (0 或 1)。

如果在两个时间序列之间使用比较运算符，并且仅一个时间序列具有特定时间戳值，则该函数会将另一个时间序列中的缺失值视为 0。

您可以将逻辑运算符与比较运算符结合使用来创建更复杂的函数。

下表列出了受支持的运算符。

运算符类型	支持的运算符
比较运算符	== != <= >= < >
逻辑运算符	AND 或 && OR 或

为了说明如何使用这些运算符，假设我们有两个时间序列：metric1 具有值 [30, 20, 0, 0]，metric2 具有值 [20, -, 20, -]，其中 - 指示未提供该时间戳的值。

表达式	输出
(metric1 < metric2)	0, 0, 1, 0

表达式	输出
(metric1 >= 30)	1, 0, 0, 0
(metric1 > 15 AND metric2 > 15)	1, 0, 0, 0

指标数学支持的函数

下表描述了可在数学表达式中使用的函数。用大写字母输入所有函数。

任何数学表达式的最终结果都必须是单个时间序列或一组时间序列。以下部分中的表中的某些函数会生成标量数字。您可以在一个更大的函数中使用这些函数，从而最终生成一个时间序列。例如，采用单个时间序列的 AVG 会生成标量数字，因此它不能是最终的表达式结果。但您可以在函数 m1-AVG(m1) 中使用它来显示每个数据点和该数据点的平均值之间的差异的时间序列。

在下表中，Examples (示例) 列中的每个示例都是一个表达式，用于生成单个时间序列或一组时间序列。这些示例说明如何将返回标量数字的函数用作生成单个时间序列的有效表达式的一部分。

函数	Argument	返回类型*	描述	示例	支持跨账户？
ABS	TS TS[]	TS TS[]	返回每个数据点的绝对值。	ABS(m1-m2) MIN(ABS([m1, m2])) ABS(METRICS())	✓
ANOMALY_DETECTION_BAND	TS, S TS, S	TS TS	返回指定指标的异常检测范围。该范围由两个时间序列组成，一个表示指标的“正常”预期值的上限，另一个表示下限。该函数可接受两个参数。第一个参数是要为其创建范围的指标的 ID。第二个参数是要用于范围的标准差的数目。如果您不指定此参数，则使用默认值 2。有关更多信息，请参阅 使用 CloudWatch 异常检测 (p. 113) 。	ANOMALY_DETECTION_BAND(m1) ANOMALY_DETECTION_BAND(m1,4)	
AVG	TS TS[]	S TS	单个时间序列的 AVG 将返回一个标量，表示指标中所有数据点的平均值。一组时间序列的 AVG 将返回单个时间序列。缺少的值被视为 0。 Note 我们建议您不要在 CloudWatch 告警中使用此函数。每当告警评估是否更改状态时，CloudWatch 都会尝试检索高于	SUM([m1,m2])/AVG(m2) AVG(METRICS())	✓

函数	Argument	返回类型*	描述	示例	支持跨账户？
			评估期指定的数量的数据点数。当请求了额外数据时，此函数的行为将有所不同。		
CEIL	TS TS[]	TS TS[]	返回每个指标的上限。上限是大于或等于每个值的最小整数。	CEIL(m1) CEIL(METRICS()) SUM(CEIL(METRICS()))	✓
DATAPOINT_COUNT	TS TS[]	S TS	返回报告了值的数据点的计数。这对于计算稀疏指标的平均值非常有用。 Note 我们建议您不要在 CloudWatch 告警中使用此函数。每当告警评估是否更改状态时，CloudWatch 都会尝试检索高于评估期指定的数量的数据点数。当请求了额外数据时，此函数的行为将有所不同。	SUM(m1)/ DATAPOINT_COUNT(m1) DATAPOINT_COUNT(METRICS())	✓
DIFF	TS TS[]	TS TS[]	返回时间序列中各值与该时间序列中的先前值之间的差值。	DIFF(m1)	✓
DIFF_TIME	TS TS[]	TS TS[]	返回时间序列中各值的时间戳与该时间序列中的先前值的时间戳之间的差值（以秒为单位）。	DIFF_TIME(METRICS())	✓

函数	Argument	返回类型*	描述	示例	支持跨账户？
FILL	TS, [S REPEAT LINEAR] TS[], [TS S REPEAT LINEAR]	TS TS[]	填充时间序列的缺失值。有几个选择可用作缺失值的填充内容： - 您可以指定要用作填充值的值。 - 您可以指定要用作填充值的指标。 - 您可以使用 REPEAT 关键字，以使用缺失值之前的指标的最新实际值来填充缺失值。 - 您可以使用 LINEAR 关键字，以使用在缺口的开头和结尾的值之间创建线性插值的值来填充缺失值。 Note 当您在告警中使用此函数时，如果您的指标发布稍有延迟，且最近一分钟从未有数据，则可能会出现这个问题。在这种情况下，FILL 将会以请求的值来替代缺失的数据点。这会导致指标的最近数据点始终为 FILL 值，从而导致告警卡在 OK (正常) 或 ALARM (告警) 状态。您可以通过使用 M 个告警 (共 N 个) 来解决此问题。有关更多信息，请参阅评估告警 (p. 152)。	FILL(m1,10) FILL(METRICS(), 0) FILL(METRICS(), m1) FILL(m1, MIN(m1)) FILL(m1, REPEAT) FILL(METRICS(), LINEAR)	✓
FIRST LAST	TS[]	TS	返回时间序列数组中的第一个或最后一个时间序列。这在与 SORT 函数结合使用时非常有用。它还可用于从 ANOMALY_DETECTION_BAND 函数中获取高阈值和低阈值。	IF(FIRST(SORT(METRICS(), AVG, DESC))>100, 1, 0) 查看数组中的最高指标，该指标通过 AVG 进行排序。然后，它为每个数据点返回 1 或 0，具体取决于相应数据点的值是否大于 100。 LAST(ANOMALY_DETECTION_BAND(m1)) 返回异常预测带的下限。	✓

函数	Argument	返回类型*	描述	示例	支持跨账户？
FLOOR	TS TS[]	TS TS[]	返回每个指标的下限。下限是小于或等于每个值的最大整数。	FLOOR(m1) FLOOR(METRICS())	✓
IF	IF 表达式	TS	通过将 IF 与比较运算符结合使用，可以从时间序列中筛选出数据点或创建由多个已整理的时间序列组成的混合时间序列。有关更多信息，请参阅 使用 IF 表达式 (p. 128) 。	有关示例，请参阅 使用 IF 表达式 (p. 128) 。	✓
INSIGHT_RULE_METRIC	INSIGHT_RULE_METRIC(metricName)	TS	使用 INSIGHT_RULE_METRIC(ruleName, INSIGHT_RULE_METRIC(metricName)) 可在 Contributor Insights 中从规则提取统计数据。有关更多信息，请参阅 绘制规则生成的指标的图表 (p. 505) 。		
LOG	TS TS[]	TS TS[]	时间序列的 LOG 返回的是该时间序列中每个值的自然对数值。	LOG(METRICS())	✓
LOG10	TS TS[]	TS TS[]	时间序列的 LOG10 返回的是该时间序列中每个值的以 10 为底的对数值。	LOG10(m1)	✓
MAX	TS TS[]	S TS	单个时间序列的 MAX 将返回一个标量，表示指标中所有数据点的最大值。一组时间序列的 MAX 值将返回单个时间序列。 Note 我们建议您不要在 CloudWatch 告警中使用此函数。每当告警评估是否更改状态时，CloudWatch 都会尝试检索高于评估期指定的数量的数据点数。当请求了额外数据时，此函数的行为将有所不同。	MAX(m1)/m1 MAX(METRICS())	✓
METRIC_COUNT	TS[]	S	返回时间序列数组中的指标数量。	m1/ METRIC_COUNT(METRICS())	✓

函数	Argument	返回类型*	描述	示例	支持跨账户？
METRICS	null 字符串	TS[]	METRICS() 函数返回的是请求中的所有 CloudWatch 指标。数学表达式不包括在内。 您可以在一个更大的表达式中使用 METRICS()，从而最终生成单个时间序列或一组时间序列。 您可以将 METRICS() 函数与一个字符串一起使用，仅返回在其 Id 字段中包含该字符串的绘成图表的指标。例如，表达式 SUM(METRICS("errors")) 一个是所有在其 Id 字段中具有“errors”的绘成图表的指标值的总和的时间序列。您还可以使用 SUM([METRICS("4xx"), METRICS("5xx")]) 来匹配多个字符串。	AVG(METRICS()) SUM(METRICS("errors"))	✓
MIN	TS TS[]	S TS	单个时间序列的 MIN 将返回一个标量，表示指标中所有数据点的最小值。一组时间序列的 MIN 将返回单个时间序列。 Note 我们建议您不要在 CloudWatch 告警中使用此函数。每当告警评估是否更改状态时，CloudWatch 都会尝试检索高于评估期指定的数量的数据点数。当请求了额外数据时，此函数的行为将有所不同。	m1-MIN(m1) MIN(METRICS())	✓

函数	Argument	返回类型*	描述	示例	支持跨账户？
MINUTE HOUR DAY DATE MONTH YEAR EPOCH	TS	TS	这些函数采用时间序列的周期和范围，并返回一个新的非稀疏时间序列，其中每个值都基于其时间戳。 - MINUTE 返回 0 到 59 之间的整数非稀疏时间序列，表示原始时间序列中每个时间戳的 UTC 分钟。 - HOUR 返回 0 到 23 之间的整数非稀疏时间序列，表示原始时间序列中每个时间戳的 UTC 小时。 - DAY 返回 1 到 7 之间的整数非稀疏时间序列，表示原始时间序列中每个时间戳的 UTC 星期。1 表示星期一，7 表示星期日。 - DATE 返回 1 到 31 之间的整数非稀疏时间序列，表示原始时间序列中每个时间戳的 UTC 日期。 - MONTH 返回 1 到 12 之间的整数非稀疏时间序列，表示原始时间序列中每个时间戳的 UTC 月份。1 表示 1 月，12 表示 12 月。 - YEAR 返回一个整数非稀疏时间序列，表示原始时间序列中每个时间戳的 UTC 年份。 - EPOCH 返回一个整数非稀疏时间序列，表示自原始时间序列中每个时间戳的纪元以来的 UTC 时间（以秒为单位）。纪元为 1970 年 1 月 1 日。	MINUTE(m1) IF(DAY(m1)<6,m1) 仅返回工作日（即星期一到星期五）的指标。 IF(MONTH(m1) == 4,m1) 仅返回 4 月发布的指标。	✓
PERIOD	TS	S	以秒为单位返回度量的时间段。有效输入是指标，而不是其他表达式的结果。	m1/PERIOD(m1)	✓
RATE	TS TS[]	TS TS[]	返回指标每秒的变化率。此项的计算结果是最新数据点值和上一个数据点值之差除以两个值的秒数之差。	RATE(m1) RATE(METRICS())	✓

函数	Argument	返回类型*	描述	示例	支持跨账户？
REMOVE_EMPTY	TS[]	TS[]	从时间序列数组中删除所有不带数据点的时间序列。最终将获得一个时间序列数组，其中每个时间序列包含至少一个数据点。 Note 我们建议您不要在 CloudWatch 告警中使用此函数。每当告警评估是否更改状态时，CloudWatch 都会尝试检索高于评估期指定的数量的数据点数。当请求了额外数据时，此函数的行为将有所不同。	REMOVE_EMPTY(METRICS())	
RUNNING_SUM	TS TS[]	TS TS[]	返回一个时间序列，其中包含原始时间序列中值的运行总和。 Note 我们建议您不要在 CloudWatch 告警中使用此函数。每当告警评估是否更改状态时，CloudWatch 都会尝试检索高于评估期指定的数量的数据点数。当请求了额外数据时，此函数的行为将有所不同。	RUNNING_SUM([m1,m2])	✓

函数	Argument	返回类型*	描述	示例	支持跨账户？
SEARCH	搜索表达式	一个或多个 TS	返回一个或多个与您指定的搜索条件匹配的时间序列。SEARCH 函数使您能够使用一个表达式将多个相关时间序列添加到图表中。该图表会动态更新，以包含稍后添加并与搜索条件匹配的新指标。有关更多信息，请参阅在图表中使用搜索表达式 (p. 131)。 您无法根据 SEARCH 表达式创建告警。原因在于搜索表达式会返回多个时间序列，而基于数学表达式的告警只能监视一个时间序列。		✓
SERVICE_QUOTA	作为使用指标的 TS	TS	返回给定使用指标的服务配额。您可以使用此选项来可视化当前用量与配额的比较情况，并设置在接近配额时向您发送警告的警报。有关更多信息，请参阅Amazon 使用情况指标 (p. 775)。		
SLICE	(TS[], S, S) 或 (TS[], S)	TS[] TS	检索时间序列数组的一部分。这在与 SORT 结合使用时特别有用。例如，可以从时间序列数组中排除顶部结果。 可以使用两个标量参数来定义要返回的时间序列集。这两个标量将定义要返回的数组的开始（包含）和结束（不包含）。该数组从零开始建立索引，因此，数组中的第一个时间序列是时间序列 0。或者，可以仅指定一个值，CloudWatch 将返回以该值开头的所有时间序列。	SLICE(SORT(METRICS(), SUM, DESC), 0, 10) 从请求中的指标数组返回具有最高 SUM 值的 10 个指标。 SLICE(SORT(METRICS(), AVG, ASC), 5) 按 AVG 统计数据对指标数组进行排序，然后返回所有时间序列（具有最低 AVG 的 5 个时间序列除外）。	✓

函数	Argumer	返回类型*	描述	示例	支持跨账户？
SORT	(TS[], TS[] FUNCTION, SORT_ORDER) (TS[], FUNCTION, SORT_ORDER, S)		根据您指定的函数，对时间序列数组进行排序。您使用的函数可以是 AVG、MIN、MAX 或 SUM。排序顺序可以是 ASC (升序，最低值排第一位) 或 DESC (降序，最高值排第一位)。您可以选择在排序顺序后指定一个数字来充当限制。例如，如果指定的限制为 5，则仅返回排序中的前 5 个时间序列。 当图表上显示此数学函数时，图表中每个指标的标签也将进行排序和编号。	SORT(METRICS(), AVG, DESC, 10) 计算每个时间序列的平均值，对时间序列进行排序（开头为最高值），并仅返回具有最高平均值的 10 个时间序列。 SORT(METRICS(), MAX, ASC) 根据 MAX 统计数据对指标数组进行排序，然后以升序顺序返回所有指标。	✓
STDDEV	TS TS[]	S TS	单个时间序列的 STDDEV 将返回一个标量，表示指标中所有数据点的标准偏差。一组时间序列的 STDDEV 将返回单个时间序列。 Note 我们建议您不要在 CloudWatch 告警中使用此函数。每当告警评估是否更改状态时，CloudWatch 都会尝试检索高于评估期指定的数量的数据点数。当请求了额外数据时，此函数的行为将有所不同。	m1/STDDEV(m1) STDDEV(METRICS())	✓

函数	Argument	返回类型*	描述	示例	支持跨账户？
SUM	TS TS[]	S TS	单个时间序列的 SUM 将返回一个标量，表示指标中所有数据点的值的总和。一组时间序列的 SUM 将返回单个时间序列。 Note 我们建议您不要在 CloudWatch 告警中使用此函数。每当告警评估是否更改状态时，CloudWatch 都会尝试检索高于评估期指定的数量的数据点数。当请求了额外数据时，此函数的行为将有所不同。	SUM(METRICS())/SUM(m1) SUM([m1,m2]) SUM(METRICS("errors"))/SUM(METRICS("requests"))*100	✓
TIME_SERIES	S	TS	返回一个非稀疏时间序列，其中每个值都设置为标量参数。	TIME_SERIES(MAX(m1)) TIME_SERIES(5*AVG(m1)) TIME_SERIES(10)	✓

*使用仅返回标量数字的函数是无效的，因为表达式的所有最终结果都必须是单个时间序列或一组时间序列。而应将这些函数用作返回时间序列的较大表达式的一部分。

使用 IF 表达式

通过将 IF 与比较运算符结合使用，可以从时间序列中筛选出数据点或创建由多个已整理的时间序列组成的混合时间序列。

IF 使用以下参数：

```
IF(condition, trueValue, falseValue)
```

如果条件数据点的值为 0，则条件的计算结果为 FALSE；如果条件的值为任何其他值（无论该值是正值还是负值），则条件的计算结果为 TRUE。如果条件是一个时间序列，则会为每个时间戳单独计算该条件。

下面列出了有效语法。对于其中的每个语法，输出是单时间序列。

- IF(TS ##### S, S | TS, S | TS)
- IF(TS, TS, TS)
- IF(TS, S, TS)
- IF(TS, TS, S)
- IF(TS, S, S)
- IF(S, TS, TS)

以下部分提供了这些语法的更多详细信息和示例。

IF(TS ##### S, scalar2 | metric2, scalar3 | metric3)

对应的输出时间序列值：

- 如果 TS ##### S 为 TRUE，则值为 scalar2 或 metric2
- 如果 TS ##### S 为 FALSE，则值为 scalar3 或 metric3
- 是一个空白时间序列（如果 metric3 中没有对应的数据点，或者如果表达式中省略了 scalar3/metric3）

IF(metric1, metric2, metric3)

对于 metric1 的每个数据点，对应的输出时间序列值：

- 值为 metric2（如果 metric1 的对应数据点为 TRUE）。
- 值为 metric3（如果 metric1 的对应数据点为 FALSE）。
- 值为 0（如果 metric1 的对应数据点为 TRUE 且 metric2 中没有对应的数据点）。
- 被删除（如果 metric1 的对应数据点为 FALSE 且 metric3 中没有对应的数据点，或者如果表达式中省略了 metric3）。

下表显示了此语法示例。

指标或函数	值
(metric1)	[1, 1, 0, 0, -]
(metric2)	[30, -, 0, 0, 30]
(metric3)	[0, 0, 20, -, 20]
IF(metric1, metric2, metric3)	[30, 0, 20, -, -]

IF(metric1, scalar2, metric3)

对于 metric1 的每个数据点，对应的输出时间序列值：

- 值为 scalar2（如果 metric1 的对应数据点为 TRUE）。
- 值为 metric3（如果 metric1 的对应数据点为 FALSE）。
- 被删除（如果 metric1 的对应数据点为 FALSE 且 metric3 中没有对应的数据点，或者如果表达式中省略了 metric3）。

指标或函数	值
(metric1)	[1, 1, 0, 0, -]
scalar2	5
(metric3)	[0, 0, 20, -, 20]
IF(metric1, scalar2, metric3)	[5, 5, 20, -, -]

IF(metric1, metric2, scalar3)

对于 metric1 的每个数据点，对应的输出时间序列值：

- 值为 metric2 (如果 metric1 的对应数据点为 TRUE)。
- 值为 scalar3 (如果 metric1 的对应数据点为 FALSE)。
- 值为 0 (如果 metric1 的对应数据点为 TRUE 且 metric2 中没有对应的数据点)。
- 如果 metric1 中的相应数据点不存在，则被删除。

指标或函数	值
(metric1)	[1, 1, 0, 0, -]
(metric2)	[30, -, 0, 0, 30]
scalar3	5
IF(metric1, metric2, scalar3)	[30, 0, 5, 5, -]

IF(scalar1, metric2, metric3)

对应的输出时间序列值：

- 值为 metric2 (如果 scalar1 为 TRUE)。
- 值为 metric3 (如果 scalar1 为 FALSE)。
- 是一个空白时间序列 (如果表达式中省略了 metric3)。

IF 表达式的使用案例示例

以下示例说明 IF 函数的可能用法。

- 要仅显示指标的低值，请使用：

IF(metric1<400, metric1)

- 要将指标中的每个数据点更改为两个值之一，以显示原始指标的相对高值和低值，请使用：

IF(metric1<400, 10, 2)

- 要为延迟超过阈值的每个时间戳显示 1，并为所有其他数据点显示 0，请使用：

IF(latency>threshold, 1, 0)

将指标数学与 GetMetricData API 操作一起使用

您可以使用 `GetMetricData` 来通过数学表达式执行计算，还可以在一个 API 调用中检索大量指标数据。有关更多信息，请参阅 [GetMetricData](#)。

指标数学异常检测

指标数学异常检测是一项功能，您可以使用该功能针对单个指标和指标数学表达式的输出创建异常检测告警。您可以使用这些表达式来创建能可视化异常检测范围的图表。该功能支持基本的算术函数、比较和逻辑运算符以及大多数其他函数。

指标数学异常检测不支持以下功能：

- 在同一行中包含多个 `ANOMALY_DETECTION_BAND` 的表达式。
- 包含 10 个以上指标或数学表达式的表达式。

- 包含 METRICS 表达式的表达式。
- 包含 SEARCH 函数的表达式。
- 以不同时段使用指标的表达式。
- 使用高精度指标作为输入的指标数学异常检测器。

有关此功能的更多信息，请参阅 Amazon CloudWatch 用户指南中的[使用 CloudWatch 异常检测](#)。

在图表中使用搜索表达式

搜索表达式是一种类型的数学表达式，您可以将其添加到 CloudWatch 图表中。搜索表达式可让您快速向图表添加多个相关指标。它们还使您能够创建动态图表，从而自动将相应的指标添加到其显示中，即使您在第一次创建图表时不存在这些指标也是如此。

例如，您可以创建一个搜索表达式来显示区域中所有实例的 AWS/EC2 CPUUtilization 指标。如果您稍后启动新实例，则新实例的 CPUUtilization 会自动添加到图表中。

在图表中使用搜索表达式时，搜索会在指标名称、命名空间、维度名称和维度值中查找搜索表达式。您可以使用布尔运算符进行更复杂、更强大的搜索。

您无法根据 SEARCH 表达式创建告警。原因在于搜索表达式会返回多个时间序列，而基于数学表达式的告警只能监视一个时间序列。

主题

- [CloudWatch 搜索表达式语法 \(p. 131\)](#)
- [CloudWatch 搜索表达式示例 \(p. 135\)](#)
- [使用搜索表达式创建 CloudWatch 图表 \(p. 136\)](#)

CloudWatch 搜索表达式语法

有效的搜索表达式具有以下格式。

```
SEARCH(' {Namespace, DimensionName1, DimensionName2, ...} SearchTerm', 'Statistic', Period)
```

例如：

```
SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization"', 'Average', 300)
```

- 查询中单词 SEARCH 之后的第一部分（用大括号括起来）是要搜索的指标架构。指标架构包含指标命名空间以及一个或多个维度名称。在搜索查询中包含指标架构是可选的。如果指定，则指标架构必须包含命名空间，并且可以（可选）包含在该命名空间中有效的一个或多个维度名称。

除非命名空间或维度名称包含空格或非字母数字字符，否则不需要在指标架构中使用引号。在这种情况下，您必须用双引号将包含这些字符的名称引起来。

- SearchTerm 也是可选的，但有效的搜索必须包含指标架构和/或 SearchTerm。SearchTerm 通常包含一个或多个指标名称或维度值。SearchTerm 可以通过部分匹配和精确匹配来包含要搜索的多个词语。它还可以包含布尔运算符。

SearchTerm 可以包含一个或多个指示符，例如 MetricName=（如本例中所示），但不需要使用指示符。

指标架构和 SearchTerm 必须一起包含在一对单引号中。

- **Statistic** 是任何有效 CloudWatch 统计数据的名称。它必须用单引号引起来。有关更多信息，请参阅 [统计数据 \(p. 5\)](#)。
- **Period** 是聚合时间段（以秒为单位）。

上述示例在 AWS/EC2 命名空间中搜索任何具有 InstanceId 作为维度名称的指标。它返回找到的所有 CPUUtilization 指标，图表显示 Average 统计数据，聚合期为 5 分钟。

搜索表达式限制

最大搜索表达式查询大小为 1024 个字符。一个图表上最多可以包含 100 个搜索表达式。一个图表可以显示多达 500 个时间序列。

CloudWatch 搜索表达式：令牌化

当您指定 SearchTerm 时，搜索函数会搜索令牌，即 CloudWatch 自动根据完整的指标名称、维度名称、维度值和命名空间生成的子字符串。CloudWatch 生成的令牌在原始字符串中以驼峰式大小写区分。数字字符也可用作新令牌的开头，非字母数字字符用作分隔符，从而在非字母数字字符之前和之后创建令牌。

相同类型的令牌分隔符字符的连续字符串会生成一个令牌。

所有生成的令牌都是小写的。下表显示了一些生成的令牌示例。

原始字符串	生成的令牌
CustomCount1	customcount1, custom, count, 1
SDBFailure	sdbfailure, sdb, failure
Project2-trial333	project2trial333, project, 2, trial, 333

CloudWatch 搜索表达式：部分匹配项

当您指定 SearchTerm 时，搜索词也会被令牌化。CloudWatch 根据部分匹配项查找指标，部分匹配项是根据搜索词生成的单个令牌与根据指标名称、命名空间、维度名称或维度值生成的单个令牌的匹配项。

与单个令牌匹配的部分匹配搜索是不区分大小写的。例如，使用以下任一搜索词可能会返回 CustomCount1 指标：

- count
- Count
- COUNT

但是，使用 couNT 作为搜索词不会找到 couNT，因为搜索词 CustomCount1 中的大写会被令牌化为 cou 和 NT。

搜索还可以匹复合令牌，复合令牌是在原始名称中连续出现的多个令牌。为与复合令牌匹配，搜索是区分大小写的。例如，如果原始词是 CustomCount1，则 CustomCount 或 Count1 的搜索成功，但 customcount 或 count1 的搜索不成功。

CloudWatch 搜索表达式：精确匹配项

您可以定义搜索，以便通过使用双引号将搜索词中需要精确匹配的部分引起来，仅查找搜索词的精确匹配项。这些双引号被括在将整个搜索词引起来的单引号中。例如，**SEARCH(' {MyNamespace}, "CustomCount1" ', 'Maximum', 120)** 找到确切的字符串 CustomCount1（如果它在名为 MyNamespace 的命名空间中作为指标名称、维度名称或维度值存在）。但是，

搜索 `SEARCH(' {MyNamespace}, "customcount1" ', 'Maximum', 120)` 或 `SEARCH(' {MyNamespace}, "Custom" ', 'Maximum', 120)` 找不到该字符串。

您可以在单个搜索表达式中组合部分匹配词和精确匹配词。例如，`SEARCH(' {AWS/NetworkELB, LoadBalancer} "ConsumedLCUs" OR flow ', 'Maximum', 120)` 会返回名为 ConsumedLCUs 的 Elastic Load Balancing 指标和包含令牌 flow 的所有 Elastic Load Balancing 指标或维度。

使用精确匹配也是查找具有特殊字符（例如非字母数字字符或空格）的名称的好方法，如下例所示。

```
SEARCH(' {"My Namespace", "Dimension@Name"}, "Custom:Name[Special_Characters" ', 'Maximum', 120)
```

CloudWatch 搜索表达式：排除指标架构

到目前为止显示的所有示例都包括指标架构（在大括号中）。省略指标架构的搜索也是有效的。

例如，`SEARCH(' "CPUUtilization" ', 'Average', 300)` 返回所有与字符串 CPUUtilization 精确匹配的指标名称、维度名称、维度值和命名空间。在 Amazon 指标命名空间中，这可以包括多个服务（包括 Amazon EC2、Amazon ECS、SageMaker 等）中的指标。

要将此搜索范围缩小到只有一个 Amazon 服务，最佳做法是在指标架构中指定命名空间以及任何必要的维度，如下例所示。虽然这会缩小对 AWS/EC2 命名空间的搜索范围，但如果您已将 CPUUtilization 定义为其他指标的维度值，它仍会返回这些指标的结果。

```
SEARCH(' {AWS/EC2, InstanceType} "CPUUtilization" ', 'Average', 300)
```

或者，您可以在 SearchTerm 中添加命名空间，如下例所示。但是在本示例中，搜索将与任何 AWS/EC2 字符串匹配，即使它是自定义维度名称或值。

```
SEARCH(' "AWS/EC2" MetricName="CPUUtilization" ', 'Average', 300)
```

CloudWatch 搜索表达式：在搜索中指定属性名称

以下搜索 "CustomCount1" 的精确匹配将返回具有该名称的所有指标。

```
SEARCH(' "CustomCount1" ', 'Maximum', 120)
```

但它也返回维度名称、维度值或命名空间为 CustomCount1 的指标。要进一步构建搜索，您可以指定要在搜索中查找的对象类型的属性名称。以下示例搜索所有命名空间并返回名为 CustomCount1 的指标。

```
SEARCH(' MetricName="CustomCount1" ', 'Maximum', 120)
```

您还可以使用命名空间和维度名称/值对作为属性名称，如下例所示。这些示例中的第一个还演示了您也可以将属性名称与部分匹配搜索一起使用。

```
SEARCH(' InstanceType=micro ', 'Average', 300)
```

```
SEARCH(' InstanceType="t2.micro" Namespace="AWS/EC2" ', 'Average', 300)
```

CloudWatch 搜索表达式：非字母数字字符

非字母数字字符用作分隔符，并标记要将指标名称、维度名称、命名空间名称和搜索词分隔成令牌的位置。当词语被令牌化时，非字母数字字符会被剥离掉，不会出现在令牌中。例如，Network-Errors_2 生成令牌 network、errors 和 2。

您的搜索词可以包含任何非字母数字字符。如果这些字符出现在搜索词中，则可以在部分匹配中指定复合令牌。例如，以下所有搜索都将查找名为 `Network-Errors-2` 或 `NetworkErrors2` 的指标。

```
network/errors
network+errors
network-errors
Network_Errors
```

当您进行精确值搜索时，精确搜索中使用的任何非字母数字字符都必须是在要搜索的字符串中的正确字符。例如，如果您要查找 `Network-Errors-2`，则 `"Network-Errors-2"` 的搜索成功，但 `"Network_Errors_2"` 搜索不成功。

执行精确匹配搜索时，必须使用反斜杠转义以下字符。

```
" \ ( )
```

例如，要按精确匹配查找指标名称 `Europe\France Traffic(Network)`，请使用搜索词 `"Europe\France Traffic(Network)"`

CloudWatch 搜索表达式：布尔运算符

搜索支持在 `SearchTerm` 中使用布尔运算符 `AND`、`OR` 和 `NOT`。布尔运算符被括在用于将整个搜索词引起来的单引号中。布尔运算符区分大小写，因此 `and`、`or` 和 `not` 作为布尔运算符无效。

您可以在搜索中显式使用 `AND`，例如 `SEARCH('{AWS/EC2,InstanceId} network AND packets', 'Average', 300)`。在搜索词之间不使用任何布尔运算符会隐式地搜索它们，就好像存在 `AND` 运算符，因此 `SEARCH('{AWS/EC2,InstanceId} network packets ', 'Average', 300)` 产生相同的搜索结果。

使用 `NOT` 从结果中排除数据子集。例如，`SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization" NOT i-1234567890123456 ', 'Average', 300)` 返回您的所有实例（`i-1234567890123456` 实例除外）的 `CPUUtilization`。您还可以使用 `NOT` 子句作为唯一的搜索词。例如，`SEARCH('NOT Namespace=AWS ', 'Maximum', 120)` 产生所有自定义指标（具有不包括 AWS 的命名空间的指标）。

您可以在一个查询中使用多个 `NOT` 短语。例如，`SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization" NOT "ProjectA" NOT "ProjectB" ', 'Average', 300)` 返回区域中的所有实例（但维度值为 `ProjectA` 或 `ProjectB` 的实例除外）的 `CPUUtilization`。

您可以组合布尔运算符以进行更强大、更详细的搜索，如以下示例所示。使用括号对运算符进行分组。

接下来的两个示例都返回包含 `EC2` 和 `EBS` 命名空间中的 `ReadOps` 的所有指标名称。

```
SEARCH(' (EC2 OR EBS) AND MetricName=ReadOps ', 'Maximum', 120)
```

```
SEARCH(' (EC2 OR EBS) MetricName=ReadOps ', 'Maximum', 120)
```

以下示例将以前的搜索范围缩小为仅包含 `ProjectA` 的结果，这可能是维度的值。

```
SEARCH(' (EC2 OR EBS) AND ReadOps AND ProjectA ', 'Maximum', 120)
```

以下示例使用嵌套分组。它返回所有函数的 `Errors` 的 `Lambda` 指标，以及名称中包含字符串 `ProjectA` 或 `ProjectB` 的函数的 `Invocations`。

```
SEARCH('{AWS/Lambda,FunctionName} MetricName="Errors" OR (MetricName="Invocations" AND (ProjectA OR ProjectB)) ', 'Average', 600)
```


CloudWatch 搜索表达式：使用数学表达式

您可以在图表中的数学表达式内使用搜索表达式。

例如，`SUM(SEARCH(' {AWS/Lambda, FunctionName} MetricName="Errors" ', 'Sum', 300))` 返回您的所有 Lambda 函数的 Errors 指标的总和。

对搜索表达式和数学表达式使用单独的行可能会产生更有用的结果。例如，假设您在图表中使用以下两个表达式。第一行显示您的各个 Lambda 函数的单独 Errors 行。该表达式的 ID 为 e1。第二行添加另一行，以显示所有函数的错误总和。

```
SEARCH(' {AWS/Lambda, FunctionName}, MetricName="Errors" ', 'Sum', 300)
SUM(e1)
```

CloudWatch 搜索表达式示例

以下示例演示了更多搜索表达式用法和语法。让我们首先在区域内的所有实例中搜索 CPUUtilization，然后查看变体。

此示例为区域中的每个实例显示一行，从而显示 AWS/EC2 命名空间中的 CPUUtilization 指标。

```
SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" ', 'Average', 300)
```

将 InstanceId 更改为 InstanceType 会更改图表，以便为区域中使用的每个实例类型显示一行。来自每个类型的所有实例的数据针对该实例类型聚合为一行。

```
SEARCH(' {AWS/EC2,InstanceType} MetricName="CPUUtilization" ', 'Average', 300)
```

删除维度名称但将名称空间保留在架构中（如以下示例所示）会生成单个行，从而显示区域中所有实例的 CPUUtilization 指标的聚合。

```
SEARCH(' {AWS/EC2} MetricName="CPUUtilization" ', 'Average', 300)
```

以下示例按实例类型聚合 CPUUtilization，并为包含字符串 micro 的每个实例类型显示一行。

```
SEARCH(' {AWS/EC2,InstanceType} InstanceType=micro MetricName="CPUUtilization" ',
'Average', 300)
```

此示例缩小了上一个示例，将 InstanceType 更改为 t2.micro 实例的精确搜索。

```
SEARCH(' {AWS/EC2,InstanceType} InstanceType="t2.micro" MetricName="CPUUtilization" ',
'Average', 300)
```

以下搜索删除了查询的 {metric schema} 部分，因此所有命名空间中的 CPUUtilization 指标都显示在图表中。这可以返回不少结果，因为该图表为来自每个 Amazon 服务的 CPUUtilization 指标包含多个行（沿不同维度聚合）。

```
SEARCH(' MetricName="CPUUtilization" ', 'Average', 300)
```

要稍微缩小这些结果范围，可以指定两个特定的指标名称空间。

```
SEARCH(' MetricName="CPUUtilization" AND ("AWS/ECS" OR "AWS/ES") ', 'Average', 300)
```

前面的示例是使用一个搜索查询来搜索特定的多个名称空间的唯一方法，因为您在每个查询中只能指定一个指标架构。但是，要添加更多结构，可以在图表中使用两个查询，如以下示例所示。在此示例中，还通过指定用于聚合 Amazon ECS 数据的维度来添加更多结构。

```
SEARCH(' {AWS/ECS, ClusterName}, MetricName="CPUUtilization" ', 'Average', 300)
SEARCH(' {AWS/EBS}, MetricName="CPUUtilization" ', 'Average', 300)
```

以下示例返回名为 ConsumedLCUs 的 Elastic Load Balancing 指标和包含令牌 flow 的所有 Elastic Load Balancing 指标或维度。

```
SEARCH(' {AWS/NetworkELB, LoadBalancer} "ConsumedLCUs" OR flow ', 'Maximum', 120)
```

以下示例使用嵌套分组。它返回所有函数的 Errors 的 Lambda 指标，以及名称中包含字符串 ProjectA 或 ProjectB 的函数的 Invocations。

```
SEARCH(' {AWS/Lambda,FunctionName} MetricName="Errors" OR (MetricName="Invocations" AND (ProjectA OR ProjectB)) ', 'Average', 600)
```

以下示例显示了所有自定义指标，不包括 Amazon 服务生成的指标。

```
SEARCH(' NOT Namespace=AWS ', 'Average', 120)
```

以下示例显示具有包含字符串 Errors 作为其名称一部分的指标名称、命名空间、维度名称和维度值的指标。

```
SEARCH(' Errors ', 'Average', 300)
```

以下示例将搜索缩小到精确匹配。例如，此搜索会查找指标名称 Errors，但不会查找名为 ConnectionErrors 或 errors 的指标。

```
SEARCH(' "Errors" ', 'Average', 300)
```

以下示例说明如何在搜索词的指标架构部分中指定包含空格或特殊字符的名称。

```
SEARCH(' {"Custom-Namespace", "Dimension Name With Spaces"}, ErrorCount ', 'Maximum', 120)
```

使用搜索表达式创建 CloudWatch 图表

在 CloudWatch 控制台上，在将图表添加到控制面板时或通过使用 Metrics（指标）视图，可以访问搜索功能。

您无法根据 SEARCH 表达式创建告警。原因在于搜索表达式会返回多个时间序列，而基于数学表达式的告警只能监视一个时间序列。

将具有搜索表达式的图表添加到现有控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards，然后选择一个控制面板。
3. 选择 Add widget。
4. 选择折线或堆积面积，然后选择配置。
5. 在 Graphed metrics（已绘制图表指标）选项卡上，选择 Add a math expression（添加数学表达式）。

6. 对于 Details (详细信息), 输入所需的搜索表达式。例如, **SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization"', 'Average', 300)**
7. (可选) 要向图表中添加其他搜索表达式或数学表达式, 请选择 Add a math expression (添加数学表达式)
8. (可选) 在添加搜索表达式后, 您可以为每个指标指定在图表图例上显示的动态标签。动态标签显示有关指标的统计数据, 并在刷新控制面板或图表时自动进行更新。要添加动态标签, 请选择 Graphed metrics (绘制的指标), 然后选择动态标签。

默认情况下, 添加到标签的动态值显示在标签开头。然后, 您可以单击指标的标签值以编辑标签。有关更多信息, 请参阅 [使用动态标签 \(p. 107\)](#)。

9. (可选) 要向图表中添加单个指标, 请选择 All metrics (所有指标) 选项卡并向下钻取到所需的指标。
10. (可选) 要更改图表上显示的时间范围, 请选择图表顶部的 custom (自定义), 或 custom (自定义) 左侧的一个时间段。
11. (可选) 横向注释帮助控制面板用户快速查看指标的峰值何时达到特定级别, 或者指标是否在预定义的范围内。要添加横向注释, 请选择 Graph options (图表选项), 然后选择 Add horizontal annotation (添加横向注释):
 - a. 对于 Label (标签), 输入注释的标签。
 - b. 对于 Value (值), 输入显示横向注释的指标值。
 - c. 对于 Fill, 指定是否将填充阴影用于此注释。例如, 为要填充的相应区域选择 Above 或 Below。如果您指定 Between, 则另一个 Value 字段将出现, 并且将填充两个值之间的图表区域。
 - d. 对于 Axis, 指定 Value 中的数字是否在图表包含多个指标的情况下表示与左侧 Y 轴或右侧 Y 轴关联的指标。

您可以通过在注释的左列中选择颜色方块来更改注释的填充色。

重复这些步骤可向同一个图表添加多个横向注释。

要隐藏注释, 请清除该注释的左列中的复选框。

要删除注释, 请在 Actions 列中选择 x。

12. (可选) 垂直注释可帮助您标记图表中的里程碑, 例如操作事件或部署的开始和结束。要添加垂直注释, 请选择 Graph options (图表选项), 然后选择 Add vertical annotation (添加垂直注释):
 - a. 对于 Label (标签), 输入注释的标签。要仅在注释中显示日期和时间, 请将 Label (标签) 字段留空。
 - b. 对于 Date (日期), 指定垂直注释出现的日期和时间。
 - c. 对于 Fill (填充), 指定是否在垂直注释前面或后面使用填充阴影或者在两个垂直注释之间使用填充阴影。例如, 为要填充的相应区域选择 Before 或 After。如果您指定 Between, 则另一个 Date 字段将出现, 并且将填充两个值之间的图表区域。

重复这些步骤可向同一个图表添加多个垂直注释。

要隐藏注释, 请清除该注释的左列中的复选框。

要删除注释, 请在 Actions 列中选择 x。

13. 选择 Create widget。
14. 选择 Save dashboard。

使用“Metrics”(指标) 视图绘制所搜索指标的图表

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中, 请选择指标。
3. 在搜索字段中, 输入要搜索的令牌: 例如, **cpuutilization t2.small**。

将会显示与您的搜索匹配的结果。

4. 要绘制与搜索匹配的所有指标的图表，请选择 Graph search (图表搜索)。

或

要细化您的搜索，请选择搜索结果中显示的一个命名空间。

5. 如果您选择了命名空间来缩小结果范围，则可以执行以下操作：
 - a. 要为一个或多个指标绘制图表，请选中每个指标旁边的复选框。要选择所有指标，请选中表的标题行中的复选框。
 - b. 要细化您的搜索，请将鼠标指针悬停在某个指标名称上，然后选择 Add to search (添加到搜索) 或 Search for this only (仅为此搜索)。
 - c. 要查看指标的帮助，请选择指标名称，然后选择 What is this?。

所选指标会显示在图表上。

6. (可选) 选择搜索栏中的一个按钮以编辑搜索词的该部分。
7. (可选) 要将图表添加到控制面板，请选择 Actions (操作)，然后选择 Add to dashboard (添加到控制面板)。

获取指标的统计数据

CloudWatch 统计数据定义

统计数据是指定时间段内的指标数据汇总。当您绘制某个指标的图表或检索其统计数据时，您可以指定时间段（例如 5 分钟），用于计算每个统计值。例如，如果时间段为 5 分钟，则总数是在此五分钟内所收集的所有样本值的总数，而最小值是在此五分钟内收集的最低值。

CloudWatch 支持指标的以下统计数据。

- 样本数是在该时间段内的数据点数。
- 总数是在该时间段内收集的所有数据点值的总数。
- 平均值是指定时间段内 Sum/SampleCount 的值。
- 最小值是指定时间段内的最小观察值。
- 最大值是指定时间段内的最大观察值。
- 百分位数 (p) 表示某个值在数据集中的相对位置。例如，p95 指第 95 个百分位数，表示该时间段内 95% 的数据低于此值，5% 的数据高于此值。百分位数可帮助您更好地了解指标数据的分布情况。
- 切尾均值 (TM) 是两个指定边界之间的所有值的均值。计算此均值时，两个边界以外的值均被忽略。您可以将边界定义为 0 到 100 之间的一个或两个数字，最多 10 位小数。这些数字可以是绝对值或百分比。例如，tm90 计算的是去掉 10% 具有最高值的数据点后的平均值。TM(2%:98%) 计算的是去掉 2% 最小数据点和 2% 最大数据点后的平均值。TM(150:1000) 计算的是去掉所有小于或等于 150 或大于 1000 的数据点后的平均值。
- 四分位均值 (IQM) 是四分位间距的切尾均值，或中间 50% 的值。它等同于 TM(25%:75%)。
- 缩尾均值 (WM) 类似于切尾均值。但是，计算缩尾均值时，不会忽略边界以外的值，而是将它们视为等于相应边界边缘的值。经过此规范化处理之后再计算平均值。您可以将边界定义为 0 到 100 之间的一个或两个数字，最多 10 位小数。例如，wm98 将计算平均值，同时将 2% 的最高值视为等于第 98 个百分位数的值。WM(10%:90%) 将计算平均值，同时将 10% 的最大数据点视为 90% 的边界值，并将 10% 的最小数据点视为 10% 的边界值。
- 百分等级 (PR) 是满足某一固定阈值的值的百分比。例如，PR(:300) 返回的是值等于或小于 300 的数据点的百分比。PR(100:2000) 返回的是值介于 100 到 2000 之间的数据点的百分比。

- 切尾计数 (TC) 指所选切尾均值统计数据范围内的数据点数。例如，tc90 返回的是不包括处于 10% 最高值区间内的任何数据点的的数据点数。TC(0.005:0.030) 返回的是值介于 0.005 (不含) 和 0.030 (含) 之间的数据点数。
- 切尾总数 (TS) 指所选切尾均值统计数据范围内数据点的值的总数。它等同于 (切尾均值) * (切尾计数)。例如，ts90 返回的是不包括处于 10% 最高值区间内的任何数据点的的数据点总数。TS(80%:) 返回的是不包括值处于 80% 最低值区间内的任何数据点的值的总数。

Note

对于“切尾均值”、“切尾计数”、“切尾总数”和“缩尾均值”，如果将两个边界定义为固定值而不是百分比，则计算时将包括等于较高边界的值，但不包括等于较低边界的值。

语法

对于“切尾均值”、“切尾计数”、“切尾总数”和“缩尾均值”，以下语法规则适用：

- 使用括号和带有百分号的一个或两个数字来定义边界，以用作数据集中位于指定的两个百分位数之间的值。例如，TM(10%:90%) 将仅使用位于第 10 和第 90 个百分位数之间的值。TM(:95%) 将使用处于数据集最低值区间到第 95 个百分位数之间的值，忽略 5% 具有最高值的数据点。
- 使用括号和无百分号的一个或两个数字来定义边界，以用作数据集中位于指定的确切值之间的值。例如，TC(80:500) 将仅使用介于 80 (不含) 和 500 (含) 之间的值。TC(:0.5) 将仅使用等于或小于 0.5 的值。
- 使用一个数字 (无括号) 以百分比形式来计算，计算过程忽略高于指定百分位数的数据点。例如，tm99 将计算均值，同时忽略 1% 具有最高值的数据点。其与 TM(:99%) 的计算结果相同。
- 在指定范围时，切尾均值、切尾计数、切尾总数和缩尾均值均可以使用大写字母缩写表示，例如 TM(5%:95%)、TM(100:200) 或 TM(:95%)。仅指定一个数字时，只能使用小写字母缩写，例如 tm99。

统计数据使用案例

- 切尾均值对于样本规模较大的指标 (例如网页延迟) 最实用。例如，tm99 会忽略可能由网络问题或人为错误导致的极高异常值，以便为典型请求的平均延迟提供更准确的数据。同样，TM(10%:) 会忽略 10% 的最低延迟值，例如由缓存命中导致的延迟值。此外，TM(10%:99%) 将排除这两种类型的异常值。
- 每当您使用切尾均值时，最好随时关注切尾计数，以确保在切尾均值计算中使用的值的数量足以具有统计意义。
- 百分等级使您能够将值归入范围的“条柱”中，而且您可以使用它来手动创建直方图。为此，请将值归入各种不同的条柱，例如 PR(:1)、PR(1:5)、PR(5:10) 和 PR(10:)。以条形图的方式可视化呈现其中每个条柱，这样就得到一个直方图。

百分位数与切尾均值

百分位数 (如 p99) 和切尾均值 (如 tm99) 衡量的是相似但并不完全相同的值。p99 和 tm99 二者都会忽略 1% 具有最高值的数据点，这些数据点被视为异常值。在此之后，p99 为剩余 99% 值的最大值，而 tm99 为剩余 99% 的值的平均值。如果您正在研究 Web 请求的延迟，p99 告诉您的是最糟糕的客户体验，而 tm99 告诉您的是平均客户体验，二者都会忽略异常值。

如果您正寻求优化客户体验，那么切尾均值是适用于监测的理想延迟统计数据。对于针对延迟作出告警，我们建议使用百分位数统计数据，以便在出现会导致部分服务丢失的问题时尽早收到提示。

对使用百分位数、切尾均值和其他一些统计数据的要求

CloudWatch 需要原始数据点来计算以下统计数据：

- 百分位数
- 切尾均值
- 四分位均值
- 缩尾均值
- 切尾总数
- 切尾计数
- 百分等级

如果您不使用原始数据而是改用统计数据发布数据，只有满足以下条件之一，才能检索此数据的这些类型统计数据：

- 统计数据集的 SampleCount 值为 1，且最小值、最大值和总和均相等。
- 最小值和最大值相等，总和等于最小值乘以 SampleCount。

以下 Amazon 服务包含支持这些类型统计数据的指标。

- API Gateway
- Application Load Balancer
- Amazon EC2
- 弹性负载均衡
- Kinesis
- Amazon RDS

此外，当任何指标值为负数时，指标的这些类型统计数据将不可用。

以下示例说明如何获取资源（例如 EC2 实例）的 CloudWatch 指标的统计数据。

示例

- [获取特定资源的统计数据 \(p. 140\)](#)
- [跨资源聚合统计数据 \(p. 143\)](#)
- [按 Auto Scaling 组聚合统计数据 \(p. 145\)](#)
- [按 Amazon Machine Image \(AMI\) 聚合统计数据 \(p. 146\)](#)

获取特定资源的统计数据

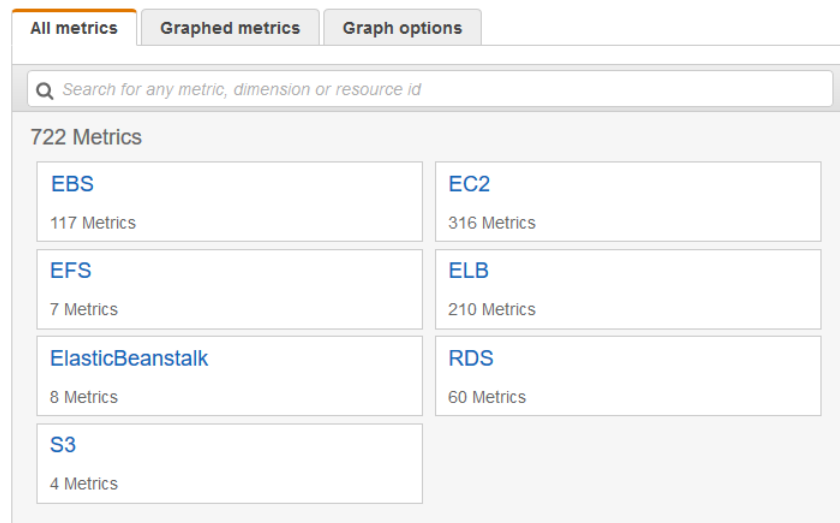
以下示例说明如何确定特定 EC2 实例的最大 CPU 使用率。

要求

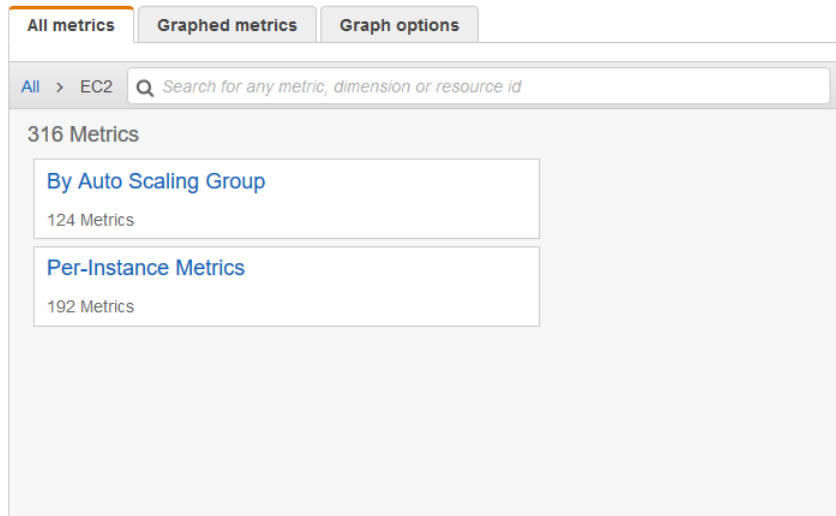
- 您必须拥有实例的 ID。可使用 Amazon EC2 控制台或 [describe-instances](#) 命令获取实例 ID。
- 默认情况下，基本监控已启用，但您可以启用详细监控。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的[为您的实例启用或禁用详细监控](#)。

使用控制台显示特定实例的平均 CPU 使用率

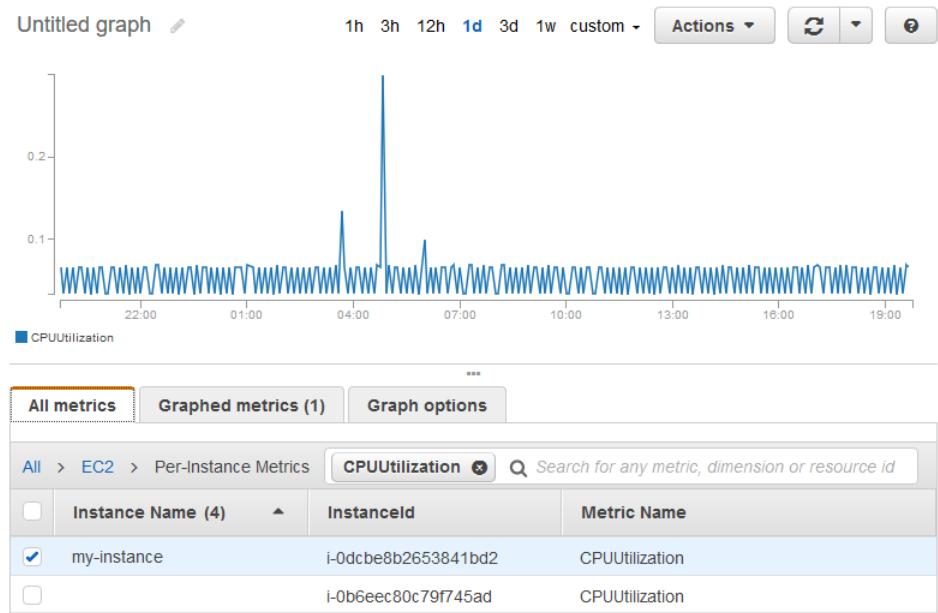
1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 选择 EC2 指标命名空间。



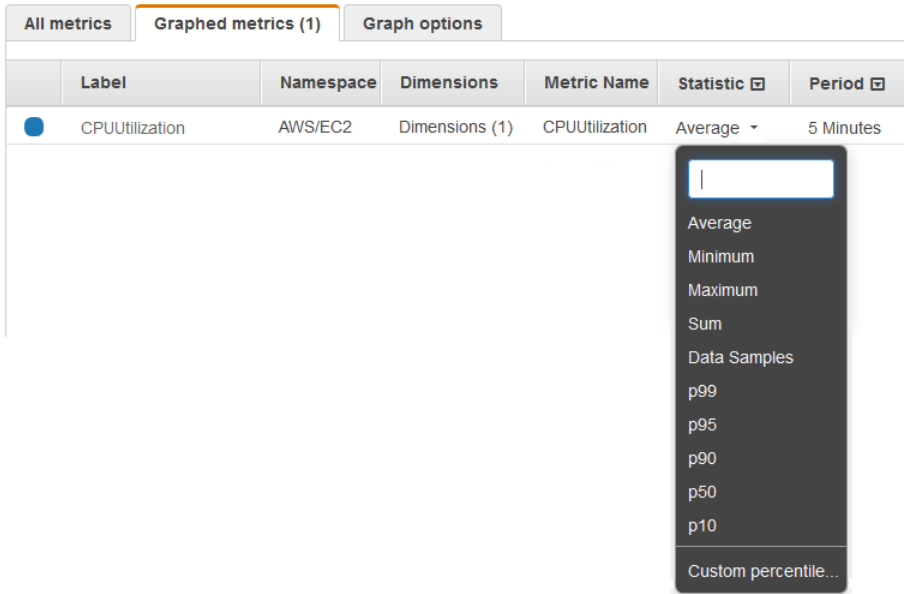
4. 选择 Per-Instance Metrics (每个实例的指标) 维度。



5. 在搜索框中，输入 **CPUUtilization** 并按 Enter。选择特定实例的行，这将显示该实例的 CPUUtilization 指标的图表。要更改图表的名称，请选择铅笔图标。要更改时间范围，请选择某个预定义的值或选择 custom。



6. 要更改统计数据，请选择 Graphed metrics (已绘制图表指标) 选项卡。选择列标题或单个值，然后选择某个统计数据或预定义百分位数，或指定自定义百分位数 (例如 **p99.999**)。



7. 要更改时间段，请选择 Graphed metrics (已绘制图表指标) 选项卡。选择列标题或单个值，然后选择其他值。

使用 Amazon CLI 获取每个 EC2 实例的 CPU 使用率

使用 `get-metric-statistics` 命令 (如下所示) 可获取指定实例的 CPUUtilization 指标：

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=InstanceId,Value=i-1234567890abcdef0 --statistics Maximum \
--start-time 2016-10-18T23:18:00 --end-time 2016-10-19T23:18:00 --period 360
```

返回的统计数据是以请求的 24 小时时间为间隔的 6 分钟数值。每个值表示特定的六分钟时间段内的指定实例的最大 CPU 使用率百分比。数据点不是按时间顺序返回的。下面显示了示例输出的开头（完整输出包括 24 小时内每 6 分钟时间段的数据点）：

```
{
  "Datapoints": [
    {
      "Timestamp": "2016-10-19T00:18:00Z",
      "Maximum": 0.33000000000000002,
      "Unit": "Percent"
    },
    {
      "Timestamp": "2016-10-19T03:18:00Z",
      "Maximum": 99.670000000000002,
      "Unit": "Percent"
    },
    {
      "Timestamp": "2016-10-19T07:18:00Z",
      "Maximum": 0.34000000000000002,
      "Unit": "Percent"
    },
    ...
  ],
  "Label": "CPUUtilization"
}
```

跨资源聚合统计数据

您可以跨多个资源聚合 Amazon 资源的指标。指标在区域之间完全独立，但您可以使用指标数学来跨区域聚合和转换类似指标。有关更多信息，请参阅[使用指标数学](#) (p. 116)。

例如，您可以聚合已启用详细监控的 EC2 实例的统计数据。不包含使用基本监控的实例。因此，您必须启用详细监控（另外收费），这将提供以 1 分钟为间隔的数据。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的[为您的实例启用或禁用详细监控](#)。

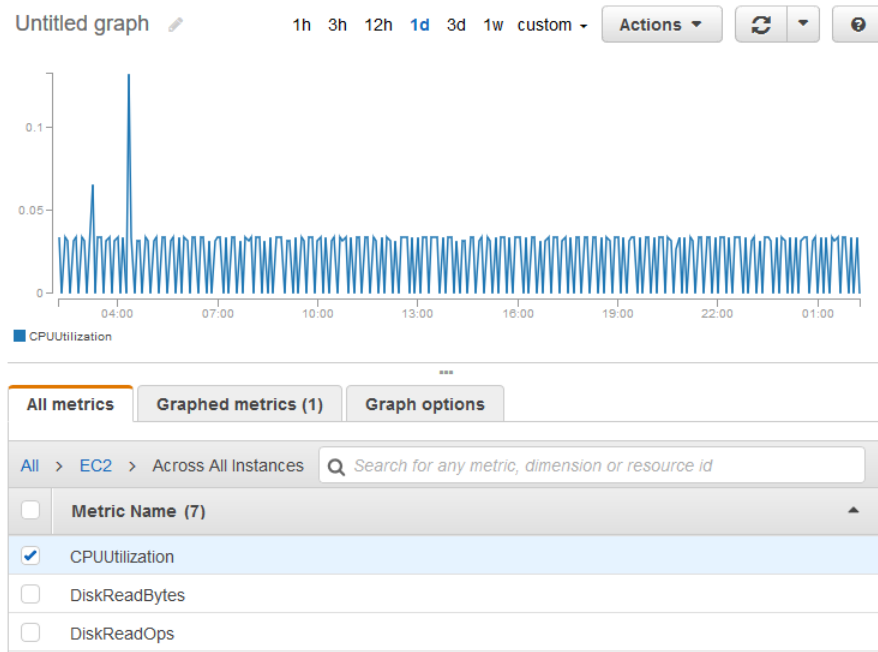
此示例说明如何获取 EC2 实例的平均 CPU 使用率。因为未指定任何维度，所以 CloudWatch 会返回 AWS/EC2 命名空间中所有维度的统计数据。要获取其他指标的统计信息，请参阅[发布 CloudWatch 指标的 Amazon 服务](#) (p. 768)。

Important

此方法可以在 Amazon 命名空间中检索所有维度，但不适用于发布到 CloudWatch 的自定义命名空间。对于自定义命名空间，必须指定与任意给定数据关联的完整的维度组，以检索包含数据点的统计数据。

显示 EC2 实例的平均 CPU 使用率

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 选择 EC2 命名空间，然后选择 Across All Instances。
4. 选择包含 CPUUtilization 的行，这将显示所有 EC2 实例的指标的图表。要更改图表的名称，请选择铅笔图标。要更改时间范围，请选择某个预定义的值或选择 custom。



5. 要更改统计数据，请选择 Graphed metrics (已绘制图表指标) 选项卡。选择列标题或单个值，然后选择某个统计数据或预定义百分位数，或指定自定义百分位数（例如 **p95.45**）。
6. 要更改时间段，请选择 Graphed metrics (已绘制图表指标) 选项卡。选择列标题或单个值，然后选择其他值。

使用 Amazon CLI 获取 EC2 实例的平均 CPU 使用率

按以下所示使用 `get-metric-statistics` 命令：

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization --
statistics "Average" "SampleCount" \
--start-time 2016-10-11T23:18:00 --end-time 2016-10-12T23:18:00 --period 3600
```

下面是示例输出：

```
{
  "Datapoints": [
    {
      "SampleCount": 238.0,
      "Timestamp": "2016-10-12T07:18:00Z",
      "Average": 0.038235294117647062,
      "Unit": "Percent"
    },
    {
      "SampleCount": 240.0,
      "Timestamp": "2016-10-12T09:18:00Z",
      "Average": 0.16670833333333332,
      "Unit": "Percent"
    },
    {
      "SampleCount": 238.0,
      "Timestamp": "2016-10-11T23:18:00Z",
      "Average": 0.041596638655462197,
      "Unit": "Percent"
    }
  ],
}
```

```
    ...  
  ],  
  "Label": "CPUUtilization"  
}
```

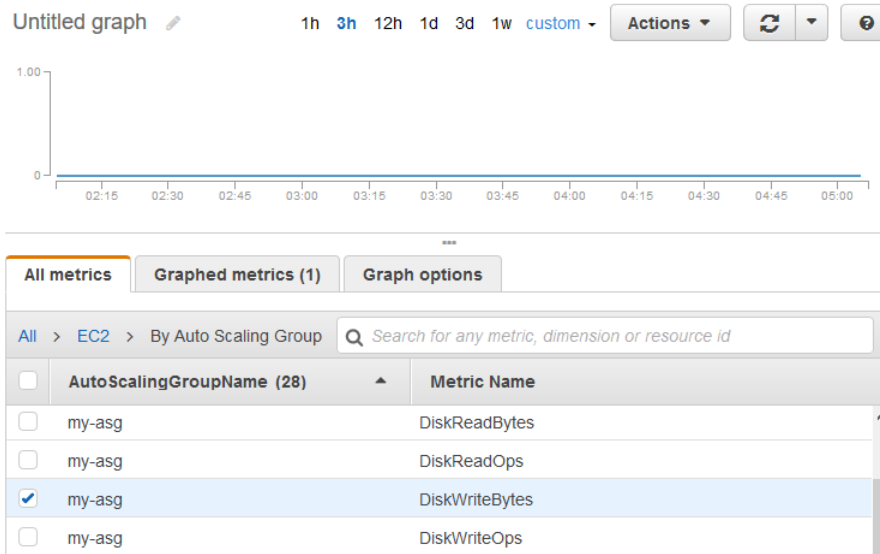
按 Auto Scaling 组聚合统计数据

您可以聚合 Auto Scaling 组中 EC2 实例的统计数据。指标在区域之间完全独立，但您可以使用 CloudWatch 指标数学来聚合和转换来自多个区域的指标。您还可以使用跨账户控制面板对来自不同账户的指标执行指标数学。

此示例说明如何获取为一个 Auto Scaling 组写入磁盘的字节总数。总数以 1 分钟为周期、24 小时为间隔针对指定 Auto Scaling 组中的所有 EC2 实例计算得出。

使用控制台显示 Auto Scaling 组中实例的 DiskWriteBytes

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 选择 EC2 命名空间，然后选择 By Auto Scaling Group。
4. 选择 DiskWriteBytes 指标和特定 Auto Scaling 组所在的行，随后将显示该 Auto Scaling 组中实例的指标的图表。要更改图表的名称，请选择铅笔图标。要更改时间范围，请选择某个预定义的值或选择 custom。



5. 要更改统计数据，请选择 Graphed metrics (已绘制图表指标) 选项卡。选择列标题或单个值，然后选择某个统计数据或预定义百分位数，或指定自定义百分位数（例如 **p95.45**）。
6. 要更改时间段，请选择 Graphed metrics (已绘制图表指标) 选项卡。选择列标题或单个值，然后选择其他值。

使用 Amazon CLI 获取 Auto Scaling 组中实例的 DiskWriteBytes

使用 `get-metric-statistics` 命令，如下所示。

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name DiskWriteBytes  
--dimensions Name=AutoScalingGroupName,Value=my-asg --statistics "Sum" "SampleCount" \  
--start-time 2016-10-16T23:18:00 --end-time 2016-10-18T23:18:00 --period 360
```

下面是示例输出。

```
{
  "Datapoints": [
    {
      "SampleCount": 18.0,
      "Timestamp": "2016-10-19T21:36:00Z",
      "Sum": 0.0,
      "Unit": "Bytes"
    },
    {
      "SampleCount": 5.0,
      "Timestamp": "2016-10-19T21:42:00Z",
      "Sum": 0.0,
      "Unit": "Bytes"
    }
  ],
  "Label": "DiskWriteBytes"
}
```

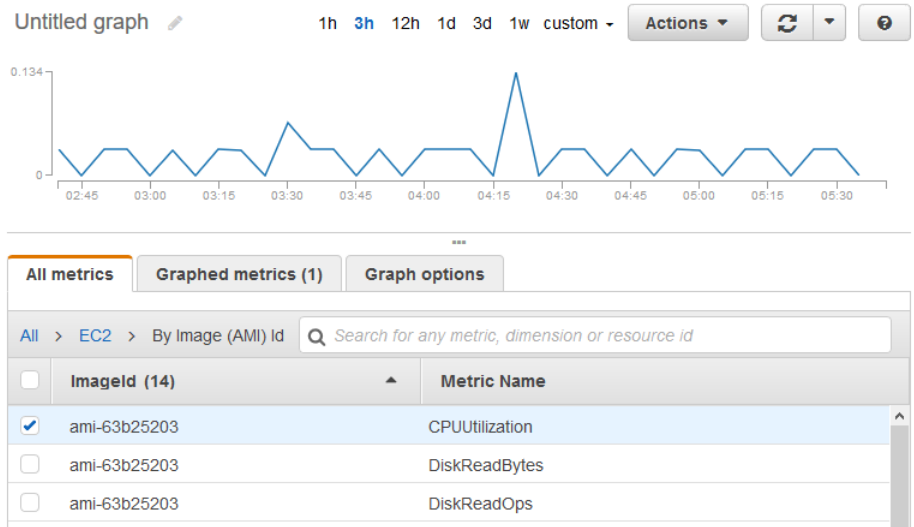
按 Amazon Machine Image (AMI) 聚合统计数据

您可以聚合已启用详细监控的 EC2 实例的统计数据。不包含使用基本监控的实例。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的[为您的实例启用或禁用详细监控](#)。

此示例说明如何确定使用指定 AMI 的所有实例的平均 CPU 使用率。平均值以 60 秒为时间间隔 1 天为周期。

使用控制台按 AMI 显示平均 CPU 使用率

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 选择 EC2 命名空间，然后选择 By Image (AMI) Id。
4. 选择 CPUUtilization 指标和特定 AMI 的行，这将显示指定 AMI 的指标的图表。要更改图表的名称，请选择铅笔图标。要更改时间范围，请选择某个预定义的值或选择 custom。



5. 要更改统计数据，请选择 Graphed metrics (已绘制图表指标) 选项卡。选择列标题或单个值，然后选择某个统计数据或预定义百分位数，或指定自定义百分位数（例如 **p95.45**）。
6. 要更改时间段，请选择 Graphed metrics (已绘制图表指标) 选项卡。选择列标题或单个值，然后选择其他值。

使用 Amazon CLI 获取按 AMI 聚合的平均 CPU 使用率

使用 `get-metric-statistics` 命令，如下所示。

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=ImageId,Value=ami-3c47a355 --statistics Average \
--start-time 2016-10-10T00:00:00 --end-time 2016-10-11T00:00:00 --period 3600
```

操作返回的统计信息是以 1 天时间为间隔的 1 分钟数值。每个数值代表运行指定 AMI 的 EC2 实例的平均 CPU 使用率百分比。下面是示例输出。

```
{
  "Datapoints": [
    {
      "Timestamp": "2016-10-10T07:00:00Z",
      "Average": 0.041000000000000009,
      "Unit": "Percent"
    },
    {
      "Timestamp": "2016-10-10T14:00:00Z",
      "Average": 0.079579831932773085,
      "Unit": "Percent"
    },
    {
      "Timestamp": "2016-10-10T06:00:00Z",
      "Average": 0.036000000000000011,
      "Unit": "Percent"
    },
    ...
  ],
  "Label": "CPUUtilization"
}
```

发布自定义指标

您可使用 Amazon CLI 或 API 将自己的指标发送到 CloudWatch。您可使用 Amazon Web Services Management Console 查看所发布指标的统计信息图表。

CloudWatch 会将关于指标的数据储存为一系列数据点。每个数据点包含一个相关联的时间戳。您甚至可以发布被称为统计数据集的聚合数据点集。

主题

- [高精度指标 \(p. 147\)](#)
- [使用维度 \(p. 148\)](#)
- [发布单一数据点 \(p. 148\)](#)
- [发布统计数据集 \(p. 149\)](#)
- [发布值零 \(p. 149\)](#)
- [停止发布指标 \(p. 150\)](#)

高精度指标

每个指标均为以下类型之一：

- 标准精度，数据粒度为一分钟
- 高精度，数据粒度为一秒

Amazon 服务生成的指标在默认情况下为标准精度。在发布自定义指标时，您可以将其定义为标准精度或高精度。发布高精度指标时，CloudWatch 使用 1 秒的精度来存储指标，您可以按照 1 秒、5 秒、10 秒、30 秒或 60 秒的任意倍数的时间段读取和检索。

高精度指标让您对应用程序的亚分钟级活动有着更详细的直观认识。请记住，每次对自定义指标的 `PutMetricData` 调用都会收取费用，因此对高精度指标频繁调用 `PutMetricData` 会导致较高的费用。有关 CloudWatch 定价的信息，请参阅 [Amazon CloudWatch 定价](#)。

如果对高精度指标设置告警，您可以指定 10 秒或 30 秒时间段的高精度告警，也可以设置 60 秒的任意倍数时间段的定期告警。10 秒或 30 秒时间段的高精度警报会产生较高的费用。

使用维度

在自定义指标中，`--dimensions` 参数很常见。维度进一步阐明了指标是什么以及指标存储什么数据。您可以将最多 30 个维度分配给一个指标，每个维度通过一个名称和值对进行定义。

当使用不同的命令时，指定维度的方式也不同。使用 `put-metric-data` 时，将每个维度指定为 `MyName=MyValue` 格式，使用 `get-metric-statistics` 或 `put-metric-alarm` 时，将使用格式 `Name=MyName`、`Value=MyValue`。例如，以下命令将发布一个包含两个维度（分别名为 `InstanceId` 和 `InstanceType`）的 `Buffers` 指标。

```
aws cloudwatch put-metric-data --metric-name Buffers --namespace MyNameSpace --unit Bytes
--value 231434333 --dimensions InstanceId=1-23456789,InstanceType=m1.small
```

此命令检索同一指标的统计数据。使用逗号分隔单一维度的名称部分和值部分，但如果有多维度，应在不同维度之间使用空格分隔。

```
aws cloudwatch get-metric-statistics --metric-name Buffers --namespace MyNameSpace --
dimensions Name=InstanceId,Value=1-23456789 Name=InstanceType,Value=m1.small --start-time
2016-10-15T04:00:00Z --end-time 2016-10-19T07:00:00Z --statistics Average --period 60
```

如果单个指标包含多个维度，则在使用 `get-metric-statistics` 时必须为每个已定义的维度指定一个值。例如，Amazon S3 指标 `BucketSizeBytes` 包含维度 `BucketName` 和 `StorageType`，因此必须使用 `get-metric-statistics` 指定这两个维度。

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --start-time
2017-01-23T14:23:00Z --end-time 2017-01-26T19:30:00Z --period 3600 --namespace
AWS/S3 --statistics Maximum --dimensions Name=BucketName,Value=MyBucketName
Name=StorageType,Value=StandardStorage --output table
```

要查看为指标定义的维度，请使用 `list-metrics` 命令。

发布单一数据点

要发布新指标或现有指标的单一数据点，请使用带有一个值和时间戳的 `put-metric-data` 命令。例如，下列每项操作会发布一个数据点。

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --value 2
--timestamp 2016-10-20T12:00:00.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --value 4
--timestamp 2016-10-20T12:00:01.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --value 5
--timestamp 2016-10-20T12:00:02.000Z
```

如果使用新的指标名称调用此命令，CloudWatch 会为您创建一个指标。否则，CloudWatch 会将您的数据与您指定的现有指标进行关联。

Note

当您创建指标时，可能需要 2 分钟时间，然后才能使用 `get-metric-statistics` 命令检索新指标的统计数据。但是，这可能需要最多 15 分钟时间，然后新指标才会出现在使用 `list-metrics` 命令检索的指标列表中。

尽管可以以千分之一秒粒度的时间戳发布数据点，但是 CloudWatch 会将数据聚合到一秒钟的最低粒度。CloudWatch 记录每个时间段内收到的值的平均值（所有项之和除以项数）以及同一个时间段内的样本数、最大值和最小值。例如，之前示例中的 `PageViewCount` 指标包含三个以若干秒为分隔的时间戳的数据点。如果将时间段设置为 1 分钟，则 CloudWatch 会聚合三个数据点，因为它们都具有 1 分钟时间段内的时间戳。

您可以使用 `get-metric-statistics` 命令，基于您发布的数据点来检索统计数据。

```
aws cloudwatch get-metric-statistics --namespace MyService --metric-name PageViewCount \
--statistics "Sum" "Maximum" "Minimum" "Average" "SampleCount" \
--start-time 2016-10-20T12:00:00.000Z --end-time 2016-10-20T12:05:00.000Z --period 60
```

下面是示例输出。

```
{
  "Datapoints": [
    {
      "SampleCount": 3.0,
      "Timestamp": "2016-10-20T12:00:00Z",
      "Average": 3.6666666666666665,
      "Maximum": 5.0,
      "Minimum": 2.0,
      "Sum": 11.0,
      "Unit": "None"
    }
  ],
  "Label": "PageViewCount"
}
```

发布统计数据集

您可在将数据发布到 CloudWatch 之前对数据进行聚合。如果您每分钟拥有多个数据点，则数据聚合会将调用 `put-metric-data` 的数目减到最少。例如，与其将彼此相隔 3 秒之内的三个数据点调用 `put-metric-data` 多次，不如使用 `--statistic-values` 参数将数据聚合到以一次调用发布的统计信息集中：

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService
--statistic-values Sum=11,Minimum=2,Maximum=5,SampleCount=3 --
timestamp 2016-10-14T12:00:00.000Z
```

CloudWatch 需要原始数据点来计算百分位数。如果您改用统计数据集发布数据，则无法检索此数据的百分位数统计数据，除非满足以下条件之一：

- 统计数据集的 `SampleCount` 为 1。
- 统计数据集的 `Minimum` 和 `Maximum` 相等

发布值零

如果数据比较分散并有一些包含无关联数据的时间段，则可以选择为该时间段发布值零 (0) 或者无任何值。如果您通过定期调用 `PutMetricData` 来监控应用程序的运行状况，则可能需要发布零值来代替无值。例

如，可以设置一个当应用程序发布指标失败时每隔五分钟向您发送通知的 CloudWatch 告警。您想让这样的应用程序为不含任何关联数据的时间段发布零值。

如果想要追踪数据点的总数或需要最小和平均等统计数据包含 0 值数据点，也可以发布零值。

停止发布指标

要停止向 CloudWatch 发布自定义指标，请更改应用程序或服务的代码以停止使用 `PutMetricData`。CloudWatch 不会从应用程序中提取指标，它只会接收推送的内容，因此要停止发布指标，您必须在源头停止这些指标。

使用 Amazon CloudWatch 告警

您可以在 Amazon CloudWatch 中创建指标和复合警报。

- 指标告警可监控单个 CloudWatch 指标，或基于 CloudWatch 指标监控数学表达式的结果。告警根据指标或表达式在多个时间段内相对于某阈值的值执行一项或多项操作。操作可以是向 Amazon SNS 主题发送通知、执行 Amazon EC2 操作或 Amazon EC2 Auto Scaling 操作，或在 Systems Manager 中创建 OpsItem 或事件。
- 复合告警包括一个规则表达式，该表达式考虑您已创建的其他告警的告警状态。只有当规则的所有条件都得到满足时，复合告警才会进入“ALARM (告警)”状态。在复合告警的规则表达式中指定的告警可以包括指标告警和其他复合告警。

使用复合告警可以减少告警噪音。您可以创建多个指标告警，还可以创建复合告警并仅为复合告警设置提示。例如，只有当所有底层指标告警都处于“ALARM (告警)”状态时，复合告警才可能进入“ALARM (告警)”状态。

复合告警可以在改变状态时发送 Amazon SNS 通知，并且可以在进入“ALARM (告警)”状态时创建 Systems Manager OpsItems 或事件，但无法执行 EC2 操作或 Auto Scaling 操作。

Note

您可以在您的 Amazon 账户中创建任意数量的警报。

您可以为控制面板添加警报，以便监控与接收跨多个区域的 Amazon 资源和应用程序的提醒。为控制面板添加警报以后，该警报会在处于 `INSUFFICIENT_DATA` 状态时变成灰色，在 `ALARM` 状态时变成红色。当处于 `OK` 状态时，警报没有颜色显示。

您还可以在 CloudWatch 控制台的导航面板中通过 Favorites and recents (收藏夹和最近记录) 选项收藏最近访问过的警报。Favorites and recents (收藏夹和最近记录) 选项中有对应的列，分别显示您收藏的警报和最近访问过的警报。

告警仅在告警状态更改时才会调用操作。使用 Auto Scaling 操作的告警除外。对于 Auto Scaling 操作，告警会在告警保持新状态时以每分钟一次的频率持续调用操作。

告警可以在同一账户中监视指标。如果您在 CloudWatch 控制台中启用了跨账户功能，则还可以创建告警，以监视其他 Amazon 账户。不支持创建跨账户复合告警。支持创建使用数学表达式的跨账户告警，但跨账户告警不支持 `ANOMALY_DETECTION_BAND`、`INSIGHT_RULE` 和 `SERVICE_QUOTA` 函数。

Note

CloudWatch 不会测试或验证您指定的操作，也不会检测因试图调用不存在的操作而导致的任何 Amazon EC2 Auto Scaling 或 Amazon SNS 错误。请确保您的告警操作存在。

指标告警状态

指标告警可能具有以下几种状态：

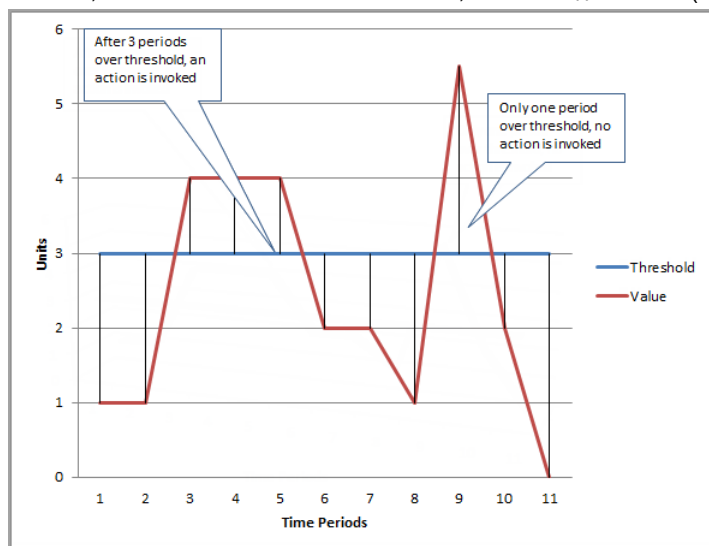
- `OK` – 指标或表达式在定义的阈值范围内。
- `ALARM` – 指标或表达式超出定义的阈值。
- `INSUFFICIENT_DATA` (数据不足) – 告警刚刚启动，指标不可用，或者指标没有足够的数据以确定告警状态。

评估告警

创建告警时，请指定三个设置，以启用 CloudWatch 评估在何时更改告警状态：

- **Period (时间段)** 是为创建告警的各个数据点而对该指标或表达式进行评估的时间长度。它以秒为单位。如果选择一分钟作为周期，则告警每分钟评估一次指标。
- **Evaluation Periods (评估期)** 是确定告警状态时要评估的最近时间段或数据点的数量。
- **Datapoints to Alarm (触发告警的数据点数)** 是评估期内必须违例才能触发告警变为 **ALARM (告警)** 状态的数据点数量。超出阈值的数据点不必是连续的，但它们必须全部在等于 **Evaluation Period (评估期)** 的最近几个数据点范围内。

在下图中，指标告警的阈值设置为三个单位。Evaluation Periods (评估期) 和 Datapoints to Alarm (触发告警的数据点数) 均为 3。也就是说，在最近三个连续评估期中的所有现有数据点都高于阈值时，告警就会变为 **ALARM (告警)** 状态。在该图中，在第三个到第五个时间段发生了这种情况。在第六个时间段，数值降至阈值以下，因此其中一个时间段被评估为未违例，且告警状态恢复为 **OK (正常)**。在第九个时间段，再次超出阈值，但仅一个时间段超出阈值。因此，告警状态保持为 **OK (正常)**。



在将 Evaluation Periods (评估期) 和 Datapoints to Alarm (触发告警的数据点数) 配置为不同的值时，您将设置“M (最大为 N)”告警。Datapoints to Alarm (触发告警的数据点数) 为 (“M”)，而 Evaluation Periods (评估期) 为 (“N”)。评估间隔等于数据点数乘以评估期时长。例如，如果为 1 分钟的评估期配置 4 个数据点 (最大为 5 个数据点)，则评估间隔为 5 分钟。如果为 10 分钟的评估期配置 3 个数据点 (最大为 3 个数据点)，则评估间隔为 30 分钟。

Note

如果创建告警后不久便有数据点缺失，并且该指标在您创建告警之前便已报告给 CloudWatch，则 CloudWatch 在评估告警时会检索从创建告警之前算起的最近数据点。

告警操作

您可以指定告警在“OK (正常)”、“ALARM (告警)”和“INSUFFICIENT_DATA (数据不足)”状态之间更改状态时所执行的操作。最常见的告警操作类型是通过向 Amazon Simple Notification Service 主题发送消息来通知一人或多人。有关 Amazon SNS 的更多信息，请参阅[什么是 Amazon SNS ?](#)

基于 EC2 指标的告警还可以执行 EC2 操作，例如停止、终止、重启或恢复 EC2 实例。有关更多信息，请参阅[创建告警以停止、终止、重启或恢复 EC2 实例 \(p. 178\)](#)。

告警还可以执行操作来扩展 Auto Scaling 组。有关更多信息，请参阅 [Amazon EC2 Auto Scaling 的步进和简单扩展策略](#)。

您还可以将告警配置为在 Systems Manager Ops Center 中创建 OpsItems，或在 Amazon Systems Manager Incident Manager 中创建事件。只有在告警进入“ALARM (告警)”状态时才能执行这些操作。有关更多信息，请参阅 [CloudWatch 配置为通过告警创建 OpsItems 和事件创建](#)。

配置 CloudWatch 告警处理缺失数据的方式

有时，并非指标的每个预期数据点都会报告到 CloudWatch。例如，当连接中断、服务器出现故障或指标设计为仅间歇性地报告数据时，可能会发生这种情况。

您可以通过 CloudWatch 指定在评估告警时如何处理缺失的数据点。这可帮助您对告警进行配置，使其仅在适合所监控的数据类型时变为 ALARM (告警) 状态。您可以避免在缺失数据没有指示问题时进行误报。

与每个告警始终处于三种状态之一类似，向 CloudWatch 报告的每个特定数据点将属于以下三个类别之一：

- 未违例 (在阈值范围内)
- 违例 (超出阈值)
- 缺失

对于每个告警，您可以指定 CloudWatch 将缺失数据点视为以下任一项：

- `notBreaching` – 将缺失数据点视为“良好”，并在阈值范围内
- `breaching` – 将缺失数据点视为“不良”，并超出阈值
- `ignore` (忽略) – 保持当前告警状态
- `missing` (缺失) – 如果告警评估范围内的所有数据点都缺失，则告警将转变为“INSUFFICIENT_DATA (数据不足)”状态。

最佳选择取决于指标的类型。对于持续报告数据的指标 (例如，实例的 `CPUUtilization`)，您可能希望将缺失数据点视为 `breaching` (超出)，因为它们可能表示出现了问题。但对于仅在错误发生时生成数据点的指标 (如 Amazon DynamoDB 中的 `ThrottledRequests`)，应将缺失数据视为 `notBreaching`。默认行为是 `missing`。

为您的告警选择最佳选项可防止不必要和误导性的告警条件更改，还可以更准确地指示您的系统的运行状况。

Important

评估 AWS/DynamoDB 命名空间中的指标的告警，始终会忽略缺失的数据，即便您为告警应该如何处理缺失的数据选择了不同的选项也是如此。当 AWS/DynamoDB 指标包含缺失数据时，评估该指标的告警仍将保持其当前状态。

在数据缺失时如何评估告警状态

每当告警评估是否更改状态时，CloudWatch 都会尝试检索高于 Evaluation Periods (评估期) 指定的数量的数据点数。它尝试检索的数据点的确切数量取决于告警期限长度，以及它是基于标准分辨率指标，还是基于高分辨率指标。它尝试检索的数据点的时间范围是评估范围。

一旦 CloudWatch 检索这些数据点，便会发生以下情况：

- 如果评估范围内的数据点没有缺失，CloudWatch 将根据最近收集的数据点来评估告警。评估的数据点数等于告警的 Evaluation Periods (评估期)。在评估范围内，不需要额外的数据点，因此它们将被忽略。
- 如果评估范围内的一些数据点缺失，但是从评估范围内成功检索的现有数据点的总数等于或超过告警的 Evaluation Periods (评估期)，则 CloudWatch 将根据已成功检索的最近现有数据点来评估告警状态，包

括从更远的评估范围内获得的必要的额外数据点。在此情况下，您针对如何处理缺失数据而设置的值便没有必要，并且将被忽略。

- 如果评估范围内的一些数据点缺失，并且检索的实际数据点数量少于 Evaluation Periods (评估期) 的告警数量，则 CloudWatch 将在缺失数据点中填写您针对如何处理缺失数据而指定的结果，然后评估该告警。但是，评估范围内的所有实时数据点都包含在评估中。CloudWatch 尽可能少地使用缺失数据点。

Note

该行为的一种特殊情况是，在指标流停止后的一段时间内，CloudWatch 告警可能会反复重新评估最后一组数据点。如果告警在指标流即将停止之前更改了状态，这种重新评估可能会导致告警更改状态并重新执行操作。要缓解此行为，请使用较短时间段。

下表阐明了告警评估行为的示例。在第一个表中，Datapoints to Alarm (触发告警的数据点数) 和 Evaluation Periods (评估期) 均为 3。CloudWatch 在评估告警时会检索最近的 5 个数据点，以防最近的 3 个数据点中的某些数据点丢失。5 为告警的评估范围。

由于评估范围为 5，因此第 1 列显示最近的 5 个数据点。数据点显示为：右侧为最近的数据点，0 是未违例数据点，X 是违例数据点，而 - 是缺失数据点。

第 2 列显示 3 个必需数据点中缺失的个数。即使评估了最近的 5 个数据点，也只需要 3 个 (Evaluation Periods (评估期) 的设置) 来评估告警状态。第 2 列中的数据点数是必须“填写”的数据点数 (使用有关处理丢失数据的方式的设置)。

在第 3–6 列中，列标题是如何处理缺少数据的可能值。这些列中显示了为处理缺失数据的每种可能方法设置的告警状态。

数据点	必须填写的数据点数	缺失	忽略	违例	未违例
0 - X - X	0	OK	OK	OK	OK
---- 0	2	OK	OK	OK	OK
-----	3	INSUFFICIENT_DATA	保留当前状态	ALARM	OK
0 X X - X	0	ALARM	ALARM	ALARM	ALARM
--- X -	2	ALARM	Retain current state	ALARM	OK

在上表的第二行中，即使将缺失数据视为违例，告警也会保持 OK (正常) 状态，因为一个现有的数据点未违例，并且该数据点与两个被视为违例的缺失数据点一起进行评估。下次评估该告警时，如果数据仍缺失，它将变为 ALARM (告警) 状态，因为未违例数据点不再处于评估范围内。

第三行 (所有五个最新数据点都缺失) 说明了缺失数据处理方式的设置对告警状态的影响。如果缺失的数据点被视为违例，告警将进入“ALARM (告警)”状态，而如果它们被视为未违例，则告警进入“OK (正常)”状态。如果忽略缺少的数据点，告警将保留缺失数据点之前的当前状态。如果缺少的数据点被认为已缺失，则告警没有足够的最新真实数据来进行评估，并变为“INSUFFICIENT_DATA (数据不足)”状态。

在第四行，告警转到 ALARM (告警) 状态，因为三个最近的数据点违例，而告警的 Evaluation Periods (评估期) 和 Datapoints to Alarm (触发告警的数据点数) 都设为 3。在这种情况下，缺失的数据点将被忽略，并且不需要缺失数据评估方式的设置，因为有 3 个实际数据点需要评估。

第 5 行表示告警评估的特殊情况，称为提前告警状态。有关更多信息，请参阅[避免提前转换到告警状态 \(p. 155\)](#)。

在下一个表中，Period（时间段）再次设置为 5 分钟，并且 Datapoints to Alarm（触发告警的数据点数）为 2，而 Evaluation Periods（评估期）为 3。这是 2（最大为 3），M（最大为 N）告警。

评估范围为 5。这是最近检索到的数据点的最大数目，可以在某些数据点丢失的情况下使用这些数据点。

数据点	缺失数据点数	缺失	忽略	违例	未违例
0 - X - X	0	ALARM	ALARM	ALARM	ALARM
0 0 X 0 X	0	ALARM	ALARM	ALARM	ALARM
0 - X - -	1	OK	OK	ALARM	OK
- - - - 0	2	OK	OK	ALARM	OK
- - - X -	2	ALARM	保留当前状态	ALARM	OK

在第 1 行和第 2 行中，告警始终处于“ALARM（告警）”状态，因为 3 个最近的数据点中有 2 个违例。在第 2 行中，不需要评估范围内的两个最早的数据点，因为 3 个最近的数据点中无一缺失，因此这两个较旧的数据点将被忽略。

在第 3 行和第 4 行中，只有当缺失的数据被视为违例时，告警才会进入“ALARM（告警）”状态，在这种情况下，两个最近丢失的数据点都被视为违例。在第 4 行中，这两个被视为违例的缺失数据点提供了两个触发“ALARM（告警）”状态所必要的违例数据点。

第 5 行表示告警评估的特殊情况，称为提前告警状态。有关更多信息，请参阅以下部分。

避免提前转换到告警状态

CloudWatch 告警评估包括用于尝试避免误报的逻辑，当数据出现间歇性时，告警会提前进入告警状态。上一节表中的第 5 行中显示的示例说明了这种逻辑。在这些行中，以及在以下示例中，Evaluation Periods（评估期）为 3，评估范围为 5 个数据点。Datapoints to Alarm（触发告警的数据点数）为 3，M（最大为 N）示例除外，其中 Datapoints to Alarm（触发告警的数据点数）为 2。

假设告警的最新数据是 - - - - x，其中有四个缺失的数据点，然后一个违例数据点作为最新的数据点。由于下一个数据点可能未违例，因此当数据处于 - - - - x 或者 - - - x -，且 Datapoints to Alarm（触发告警的数据点数）为 3 时，告警不会立即进入“ALARM（告警）”状态。这样，当下一个数据点未违例并导致数据为 - - - x o 或者 - - x - o 时，误报得以避免。

但是，如果最后几个数据点是 - - x - -，即使缺失的数据点被视为缺失，告警也会进入“ALARM（告警）”状态。这是因为根据告警的设计，当在数据点数量 Evaluation Periods（评估期）间最早的可用的违例数据点至少与 Datapoints to Alarm（触发告警的数据点数）的值同样早，并且所有其他较新的数据点都违例或缺失时，告警都会进入“ALARM（告警）”状态。在这种情况下，即使可用数据点的总数小于 M（Datapoints to Alarm（触发告警的数据点数）），告警也会进入“ALARM（告警）”状态。

此告警逻辑也适用于 M（最大为 N）告警。如果评估范围内最早的违例数据点至少与 Datapoints to Alarm（触发告警的数据点数）的值同样早，并且所有最新的数据点均违例或缺失，则无论 M 值（Datapoints to Alarm（触发告警的数据点数））如何，告警都会进入“ALARM（告警）”状态。

高精度告警

如果对高精度指标设置告警，您可以指定 10 秒或 30 秒时间段的高精度告警，也可以设置 60 秒的任意倍数时间段的定期告警。高精度告警的费用较高。有关高精度指标的更多信息，请参阅 [发布自定义指标 \(p. 147\)](#)。

针对数学表达式的告警

您可以针对基于一个或多个 CloudWatch 指标的数学表达式的结果设置告警。用于告警的数学表达式可以包含多达 10 个指标。每个指标都必须使用相同的时间段。

对于基于数学表达式的告警，您可以指定您希望 CloudWatch 在评估告警时，如何对待基础指标的缺失数据点。

基于数学表达式的告警无法执行 Amazon EC2 操作。

有关指标数学表达式和语法的更多信息，请参阅 [使用指标数学 \(p. 116\)](#)。

基于百分位数的 CloudWatch 告警和小数据样本

当您将其百分位数设置为某个告警的统计数据时，您可以指定在没有数量足以进行有效的统计评估的数据时所执行的操作。您仍然可以选择让告警评估统计数据，并可以更改告警状态。或者，您也可以让告警在样本大小过小时忽略指标，并等到有足够的数据来实现统计显著性时对样本进行评估。

对于介于 0.5 (含) 和 1.00 (不含) 之间的百分位数，将在评估期内数据点的数量少于 10/ (1 个百分位) 时使用此设置。例如，如果 p99 百分位上某个告警的样本数少于 1000 个，则会使用此设置。对于 0 和 0.5 (不含) 之间的百分位数，当数据点的数量少于 10/百分位数时，将使用此设置。

CloudWatch 告警和 Amazon EventBridge

每当 CloudWatch 告警更改告警状态时，CloudWatch 都会将事件发送到 Amazon EventBridge。您可以使用这些告警状态更改事件在 EventBridge 中触发事件目标。有关更多信息，请参阅 [告警事件和 EventBridge \(p. 185\)](#)。

CloudWatch 告警的常见功能

以下功能适用于所有 CloudWatch 告警：

- 您可以创建的告警数量没有限制。要创建或更新告警，请使用 CloudWatch 控制台、[PutMetricAlarm](#) API 操作，或 Amazon CLI 中的 [put-metric-alarm](#) 命令。
- 警报名称必须仅包含 ASCII 字符。
- 您可以使用 CloudWatch 控制台、[DescribeAlarms](#) API 操作，或 Amazon CLI 中的 [describe-alarms](#) 命令来列出任何或所有当前已配置的告警和列出任意特定状态的告警。
- 您可以使用 [DisableAlarmActions](#) 和 [EnableAlarmActions](#) API 操作或 Amazon CLI 中的 [disable-alarm-actions](#) 和 [enable-alarm-actions](#) 命令来禁用或启用告警。
- 您可以使用 [SetAlarmState](#) API 操作或 Amazon CLI 中的 [set-alarm-state](#) 命令将告警设为任意状态，以对其进行测试。此短暂状态变更只会持续到下一个告警比较发生时。
- 您可以在创建自定义指标之前为该自定义指标创建告警。为了使告警有效，必须在告警定义中包含自定义指标的所有维度以及指标命名空间和指标名称。要执行此操作，您可以使用 [PutMetricAlarm](#) API 操作，或 Amazon CLI 中的 [put-metric-alarm](#) 命令。
- 您可以使用 CloudWatch 控制台、[DescribeAlarmHistory](#) API 操作，或 Amazon CLI 中的 [describe-alarm-history](#) 命令查看告警的历史记录。CloudWatch 可将告警历史记录保存两个星期。每个状态转换都标有一个唯一的时间戳。个别情况下，一个状态变更的历史记录可能显示多个通知。通过时间戳可以确认独特的状态变更。
- 您可以在 CloudWatch 控制台的导航面板中通过 Favorites and recents (收藏夹和最近记录) 选项收藏最近访问过的警报，具体方法为：将鼠标悬停在您想要收藏的警报上方，然后选择它旁边的星号。
- 告警的评估期数乘以每个评估期的长度不能超过一天。

Note

在特定情况下，某些 Amazon 资源不会向 CloudWatch 发送指标数据。例如，Amazon EBS 可能不会发送未附加到 Amazon EC2 实例的可用卷的指标数据，因为该卷没有要监控的指标活动。如果为此类指标设置了一个告警，您可能会注意到其状态变为 `INSUFFICIENT_DATA`（数据不足）。这可能表示资源处于不活动状态，并不一定表示出现了问题。您可以指定每个告警如何处理丢失数据。有关更多信息，请参阅[配置 CloudWatch 告警处理丢失数据的方式](#) (p. 153)。

设置 Amazon SNS 通知

Amazon CloudWatch 使用 Amazon SNS 发送电子邮件。首先，创建一个 SNS 主题并订阅此主题。创建 CloudWatch 告警后，可以添加此 SNS 主题，以在告警改变状态时发送电子邮件通知。有关更多信息，请参阅[Amazon Simple Notification Service 入门指南](#)。

或者，如果您计划使用 Amazon Web Services Management Console 创建 CloudWatch 告警，则可跳过此过程，因为您可在创建告警时创建主题。

Note

在创建 Amazon SNS 主题时，您可以选择将其设为标准主题或 FIFO 主题。CloudWatch 定会将所有告警通知发布到这两种类型的主题。但是，即使您使用 FIFO 主题，在极少数情况下，CloudWatch 也会不按顺序将通知发送到该主题。如果您使用 FIFO 主题，告警会将告警通知的消息组 ID 设置为告警的 ARN 的哈希值。

防止混淆代理问题

为防止跨服务混淆代理安全问题，我们建议您使用 Amazon SNS 资源策略中的 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件键，向 CloudWatch 授予访问您的 Amazon SNS 资源的权限。

下面的资源策略示例使用 `aws:SourceArn` 条件键将 `SNS:Publish` 权限缩小为只能由指定账户中的 CloudWatch 警报使用。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "cloudwatch.amazonaws.com"
    },
    "Action": "SNS:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:cloudwatch:us-east-2:111122223333:alarm:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "111122223333"
      }
    }
  }]
}
```

如果警报 ARN 包含任何非 ASCII 字符，请仅使用 `aws:SourceAccount` 全局条件键限制权限。

使用 Amazon Web Services Management Console 设置 Amazon SNS 主题

首先，创建一个主题，然后订阅此主题。您可以选择将测试消息发布到此主题。

创建 SNS 主题

1. 通过以下网址打开 Amazon SNS 控制台：<https://console.aws.amazon.com/sns/v3/home>。
2. 在 Amazon SNS 控制面板上的 Common actions (常用操作) 下，选择 Create Topic (创建主题)。
3. 在 Create new topic (创建新主题) 对话框中，为 Topic name (主题名称) 输入主题的名称 (例如 **my-topic**)。
4. 选择 Create topic (创建主题)。
5. 复制下一个任务的 Topic ARN (主题 ARN) (例如 `arn:aws:sns:us-east-1:111122223333:my-topic`)。

订阅 SNS 主题

1. 通过以下网址打开 Amazon SNS 控制台：<https://console.aws.amazon.com/sns/v3/home>。
2. 在导航窗格中，依次选择 Subscriptions (订阅)、Create subscription (创建订阅)。
3. 在 Create subscription (创建订阅) 对话框中，为 Topic ARN (主题 ARN) 粘贴在上一任务中创建的主题 ARN。
4. 对于协议，选择电子邮件。
5. 对于 Endpoint (端点)，输入一个可用于接收通知的电子邮件地址，然后选择 Create subscription (创建订阅)。
6. 从您的电子邮件应用程序中，打开来自 Amazon 通知的消息并确认您的订阅。

您的 Web 浏览器将显示来自 Amazon SNS 的确认响应。

向 SNS 主题发布测试消息

1. 通过以下网址打开 Amazon SNS 控制台：<https://console.aws.amazon.com/sns/v3/home>。
2. 在导航窗格中，选择 Topics (主题)。
3. 在 Topics (主题) 页面上，选择一个主题，然后选择 Publish to topic (发布到主题)。
4. 在 Publish (发布消息) 页面中，为 Subject (主题) 输入消息的主题行，并为 Message (消息) 输入简短的消息。
5. 选择 Publish Message (发布消息)。
6. 查看电子邮件，确认您已收到消息。

使用 Amazon CLI 设置 SNS 主题

首先，您创建一个 SNS 主题，然后将一条消息直接发布到该主题，以测试您是否正确配置了该主题。

设置 SNS 主题

1. 使用 `create-topic` 命令创建主题，如下所示。

```
aws sns create-topic --name my-topic
```

Amazon SNS 返回具有以下格式的主题 ARN：

```
{
  "TopicArn": "arn:aws:sns:us-east-1:111122223333:my-topic"
}
```

2. 使用 `subscribe` 命令以通过您的电子邮件地址订阅该主题。如果订阅请求成功，您将收到一封确认电子邮件。

```
aws sns subscribe --topic-arn arn:aws:sns:us-east-1:111122223333:my-topic --protocol
email --notification-endpoint my-email-address
```

Amazon SNS 将返回以下内容：

```
{
  "SubscriptionArn": "pending confirmation"
}
```

3. 从您的电子邮件应用程序中，打开来自 Amazon 通知的消息并确认您的订阅。

您的 Web 浏览器将显示来自 Amazon Simple Notification Service 的确认响应。

4. 使用 `list-subscriptions-by-topic` 命令检查订阅。

```
aws sns list-subscriptions-by-topic --topic-arn arn:aws:sns:us-east-1:111122223333:my-
topic
```

Amazon SNS 将返回以下内容：

```
{
  "Subscriptions": [
    {
      "Owner": "111122223333",
      "Endpoint": "me@mycompany.com",
      "Protocol": "email",
      "TopicArn": "arn:aws:sns:us-east-1:111122223333:my-topic",
      "SubscriptionArn": "arn:aws:sns:us-east-1:111122223333:my-topic:64886986-
bf10-48fb-a2f1-dab033aa67a3"
    }
  ]
}
```

5. （可选）使用 `publish` 命令向主题发布测试消息。

```
aws sns publish --message "Verification" --topic arn:aws:sns:us-east-1:111122223333:my-
topic
```

Amazon SNS 将返回以下内容。

```
{
  "MessageId": "42f189a0-3094-5cf6-8fd7-c2dde61a4d7d"
}
```

6. 查看电子邮件，确认您已收到消息。

根据静态阈值创建 CloudWatch 告警

您可以为告警选择一个要监视的 CloudWatch 指标，并为该指标选择阈值。如果该指标在指定数量的评估期内超出阈值，告警将变为 ALARM（告警）状态。

根据单个指标创建告警

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Alarms（警报）和 All alarms（所有警报）。

3. 选择 Create alarm (创建警报)。
 4. 选择 Select Metric (选择指标)。
 5. (可选) 如果您在 CloudWatch 控制台中启用了跨账户功能, 并且当前账户是监控账户, 请在 Search Metrics (搜索指标) 下选择一个不同的 Amazon 账户, 该账户包含您希望告警监视的指标。有关更多信息, 请参阅[跨账户跨区域的 CloudWatch 控制台 \(p. 654\)](#)。
 6. 请执行下列操作之一:
 - 选择包含所需指标的服务命名空间。继续选择所显示的选项, 以缩小选择范围。在显示指标列表时, 选中所需的指标旁边的复选框。
 - 在搜索框中, 输入指标名称、维度或资源 ID, 然后按 **Enter**。接下来, 选择其中的一个结果并继续, 直到显示指标列表。选中所需的指标旁边的复选框。
 7. 选择 Graphed metrics (绘制的指标) 选项卡。
 - a. 在 Statistic (统计数据) 下, 选择其中的一个统计数据或预定义百分比值, 或者指定一个自定义百分比值 (例如 **p95.45**)。
 - b. 在 Period (时间段) 下, 选择告警的评估期。评估告警时, 每个时间段都聚合到一个数据点。

在创建告警时, 您还可以选择是在左侧还是右侧显示 Y 轴图例。该首选项仅在创建告警时使用。
 - c. 选择选择指标。

将显示 Specify metric and conditions (指定指标和条件) 页面, 其中显示一个图表以及有关您选择的指标和统计数据的其他信息。
 8. 在条件下面, 指定以下内容:
 - a. 对于 Whenever **metric** is (每当指标), 指定指标是必须大于、小于还是等于阈值。在... 下面, 指定阈值。
 - b. 选择其他配置。对于触发警报的数据点数, 指定必须有多少个评估期 (数据点) 处于 ALARM 状态才能触发警报。如果此处的两个值匹配, 则会创建一个告警; 如果多个连续评估期违例, 该告警将变为 ALARM (告警) 状态。

要创建“M (最大为 N)”告警, 为第一个值指定的数字应小于为第二个值指定的数字。有关更多信息, 请参阅[评估告警 \(p. 152\)](#)。
 - c. 对于缺失数据处理, 选择在缺失某些数据点时的警报行为。有关更多信息, 请参阅[配置 CloudWatch 告警处理缺失数据的方式 \(p. 153\)](#)。
 - d. 如果警报将百分比值作为监控的统计数据, 将显示样本数少的百分比框。使用它来选择是评估还是忽略采样率低的案例。如果选择忽略 (保持警报状态), 在样本大小太小时, 将始终保持当前警报状态。有关更多信息, 请参阅[基于百分位数的 CloudWatch 告警和小数据样本 \(p. 156\)](#)。
 9. 选择 Next (下一步)。
 10. 在通知下面, 选择一个在警报处于 ALARM、OK 或 INSUFFICIENT_DATA 状态时通知的 SNS 主题。

要使告警为相同告警状态或不同告警状态发送多个通知, 请选择 Add notification (添加通知)。

要让警报不发送通知, 请选择删除。
 11. 要让告警执行 Auto Scaling、EC2 或 Systems Manager 操作, 请选择相应的按钮, 然后选择告警状态和要执行的操作。告警只有在进入“ALARM (告警)”状态时才能执行 Systems Manager 操作。有关 Systems Manager 操作的更多信息, 请参阅[将 CloudWatch 配置为通过告警创建 OpsItems 和事件创建](#)。
- Note
- 要创建执行 SSM Incident Manager 操作的告警, 您必须具有特定的权限。有关更多信息, 请参阅[Amazon Systems Manager Incident Manager 的基于身份的策略示例](#)。
12. 在完成后, 选择下一步。
 13. 输入警报的名称和说明。名称只能包含 ASCII 字符。然后选择下一步。
 14. 在 Preview and create 下面, 确认具有所需的信息和条件, 然后选择 Create alarm。

您还可以将警报添加到控制面板。有关更多信息，请参阅 [在 CloudWatch 控制面板上添加或删除警报小组件 \(p. 52\)](#)。

根据异常检测创建 CloudWatch 告警

您可以根据 CloudWatch 异常检测来创建告警，该异常检测可以分析过去的指标数据并创建预期值模型。预期值会考虑指标中的典型每小时、每日和每周模式。

您需要为异常检测阈值设置一个值，然后 CloudWatch 在模型中使用该阈值来确定指标值的“正常”范围。阈值越高，所产生的“正常”值的范围越大。

您可以选择当指标值高于预期值范围、低于预期值范围，或出现二者情况之一时是否触发告警。

您还可以针对单个指标和指标数学表达式的输出创建异常检测告警。您可以使用这些表达式来创建能可视化异常检测范围的图表。

不支持基于异常检测的跨账户或跨区域告警。

有关更多信息，请参阅 [使用 CloudWatch 异常检测 \(p. 113\)](#)。

Note

如果您出于可视化目的已在 Metrics（指标）控制台对某个指标使用了异常检测，并针对该指标创建了异常检测告警，则您为该告警设置的阈值不会更改您为可视化设置的阈值。有关更多信息，请参阅 [创建图表 \(p. 105\)](#)。

创建基于异常检测的告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Alarms（告警），然后选择 All alarms（所有告警）。
3. 选择 Create alarm（创建警报）。
4. 请执行下列操作之一：
 - 选择包含您的指标的服务命名空间，然后继续选择相应选项，因为这些选项看起来会缩小您的选项范围。在显示指标列表时，选中您的指标旁边的复选框。
 - 在搜索框中，输入指标名称、维度或资源 ID，然后按键盘上的 Enter 键。选择其中的一个结果，然后继续在选项显示时选择它们，直到显示指标列表为止。选中您的指标旁边的复选框。
5. 选择 Graphed metrics（绘制的指标）。
 - a. （可选）在 Statistic（统计数据）列下，选择下拉列表，然后选择一个预定义的统计数据或百分位数。您可以使用下拉列表中的搜索框指定自定义百分位数，如 **p95.45**。
 - b. （可选）在 Period（周期）列下，选择下拉列表，然后选择一个预定义的评估周期。

Note

当 CloudWatch 评估您的告警时，它会将该周期汇总为单个数据点。对于异常检测告警，评估周期必须是一分钟或更长时间。

- c. （可选）在 Y axis（Y 轴）列下，您可以指定 Y 轴图例显示在图形的左侧还是右侧。

Note

此选项只能在创建告警时使用。

6. 选择选择指标。
7. 在条件下面，指定以下内容：
 - a. 选择 Anomaly detection（异常检测）。

如果针对此指标和统计数据的模型已存在，则 CloudWatch 在屏幕顶部的图中显示异常检测范围的预览。创建警报后，实际异常检测范围出现在图中可能需要 15 分钟。在此之前，您看到的范围是异常检测范围的近似值。

Tip

要在更长的时间范围内查看屏幕顶部的图表，请选择屏幕右上方的 Edit (编辑)。

如果此指标和统计的模型不存在，CloudWatch 在您完成创建警报后生成异常检测范围。对于新模型，实际异常检测范围出现在图表中可能需要 3 个小时。新模型的训练可能需要两周时间，因此异常检测范围可以显示更准确的预期值。

- b. 对于 Whenever **metric** is (每当指标为)，指定何时触发告警。例如，每当指标大于、小于或超出范围 (在任一方向)。
- c. 对于 Anomaly detection threshold (异常检测阈值)，选择用于异常检测阈值的数字。数字越大，创建的“正常”值的范围就越大，也更能容忍指标变化。数字越小，创建的范围就越小，它将以较小的指标偏差进入 ALARM 状态。数字不一定是整数。
- d. 选择其他配置。对于触发警报的数据点数，指定必须有多少个评估期 (数据点) 处于 ALARM 状态才能触发警报。如果此处的两个值匹配，则会创建一个告警；如果多个连续评估期违例，该告警将变为 ALARM (告警) 状态。

要创建“M (最大为 N)”告警，为第一个值指定的数字应小于为第二个值指定的数字。有关更多信息，请参阅[评估告警 \(p. 152\)](#)。

- e. 对于 Missing data treatment (缺失数据处理)，选择在缺失某些数据点时的告警行为。有关更多信息，请参阅[配置 CloudWatch 告警处理缺失数据的方式 \(p. 153\)](#)。
 - f. 如果警报将百分比值作为监控的统计数据，将显示样本数少的百分比框。使用它来选择是评估还是忽略采样率低的案例。如果选择 Ignore (maintain alarm state) (忽略 (保持警报状态))，在样本本大小太小时，将始终保持当前警报状态。有关更多信息，请参阅[基于百分位数的 CloudWatch 告警和小数据样本 \(p. 156\)](#)。
8. 选择 Next (下一步)。
 9. 在通知下面，选择一个在警报处于 ALARM、OK 或 INSUFFICIENT_DATA 状态时通知的 SNS 主题。

要为相同告警状态或不同告警状态发送多个通知，请选择 Add notification (添加通知)。

如果不想告警发送通知，请选择 Remove (移除)。

10. 您可以将告警设置为在发生状态更改时执行 EC2 操作，或在进入告警状态时创建 Systems Manager OpsItem 或事件。为此，请选择相应的按钮，然后选择警报状态和要执行的操作。

有关 Systems Manager 操作的更多信息，请参阅[将 CloudWatch 配置为通过告警创建 OpsItems 和事件创建](#)。

Note

要创建执行 Amazon Systems Manager Incident Manager 操作的警报，您必须具有特定的权限。有关更多信息，请参阅[Amazon Systems Manager Incident Manager 的基于身份的策略示例](#)。

11. 选择 Next (下一步)。
12. 在 Name and description (名称和描述) 下，输入告警的名称和描述，然后选择 Next (下一步)。

Tip

告警名称只能包含 ASCII 字符。

13. 在 Preview and create (预览和创建) 下，确认告警的信息和条件正确，然后选择 Create alarm (创建告警)。

修改异常检测模型

创建告警后，可以调整异常检测模型。您可以排除某些时间段，不在创建模型时使用。从训练数据中排除系统中断、部署和假日等异常事件至关重要。您还可以指定是否针对夏令时更改调整模型。

针对告警调整异常检测模型

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Alarms (警报) 和 All alarms (所有警报)。
3. 选择警报的名称。如有必要，请使用搜索框来查找警报。
4. 选择 View in metrics (查看指标)。
5. 在屏幕的下部分中，选择 Edit model (编辑模型)。
6. 要排除用于生成模型的时间段，请选择 Add another time range to exclude from training (添加要从训练中排除的另一个时间范围)。然后，选择或输入要从训练中排除的天数和时间，并选择 Apply (应用)。
7. 如果指标需要区分夏令时更改，请在 Metric timezone (指标时区) 框中选择相应的时区。
8. 选择 Update (更新)。

删除异常检测模型

对告警使用异常检测会产生 费用。作为最佳实践，如果警报不再需要异常检测模型，请先删除警报，然后删除模型。评估异常检测警报时，系统将代表您创建任何缺失的异常检测器。如果您删除模型而不删除告警，则告警会自动重新创建模型。

删除警报

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Alarms (告警) 和 All alarms (所有告警)。
3. 选择警报的名称。
4. 依次选择 Actions (操作) 和 Delete (删除)。

要删除曾用于告警的异常检测模型

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Metrics (指标)，然后选择 All metrics (所有指标)。
3. 选择 Browse (浏览)，然后选择包含异常检测模型的指标。您可以在搜索框中搜索您的指标，也可以通过选择选项来选择您的指标。
 - (可选) 如果要使用原始界面，请选择 All metrics (所有指标)，然后选择包含异常检测模型的指标。您可以在搜索框中搜索您的指标，也可以通过选择选项来选择您的指标。
4. 选择 Graphed metrics (绘制的指标)。
5. 在 Graphed metrics (已绘制图表的指标) 选项卡中，选择要删除的异常检测模型的名称，然后选择 Delete anomaly detection model (删除异常检测模型)。
 - (可选) 如果要使用原始界面，请选择 Edit model (编辑模型)。系统会将您定向到新屏幕。在新屏幕上，选择 Delete model (删除模型)，然后选择 Delete (删除)。

根据日志组指标筛选器创建 CloudWatch 告警

本节中的过程介绍如何基于日志组指标筛选条件创建告警。使用指标筛选条件，您可以在日志数据发送到 CloudWatch 时查找其中的字词和模式。有关更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[使用筛选条件从日志事件创建指标](#)。在根据日志组指标筛选条件创建告警之前，您必须完成以下操作：

- 创建 日志组。有关更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[使用日志组和日志流](#)。
- 创建指标筛选条件。有关更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[为日志组创建指标筛选条件](#)。

要根据日志组指标筛选条件创建告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 从左侧导航窗格中，选择 Logs（日志），然后选择 Log groups（日志组）。
3. 选择包含您的指标筛选条件的日志组。
4. 选择 Metric filters（指标筛选条件）。
5. 在指标筛选条件选项卡中，选中要作为告警基础的指标筛选条件的复选框。
6. 选择 Create alarm（创建警报）。
7. （可选）在 Metric（指标）下，编辑 Metric name（指标名称）、Statistic（统计数据）和 Period（项目）。
8. 在条件下面，指定以下内容：
 - a. 对于 Threshold type（阈值类型），选择 Static（静态）或 Anomaly detection（异常检测）。
 - b. 对于 Whenever **your-metric-name** is ...（当您的 your-metric-name ...），选择 Greater（大于）、Greater/Equal（大于/等于）、Lower/Equal（小于/等于）或 Lower（小于）。
 - c. 对于 than ...（比...），为您的阈值指定一个数值。
9. 选择其他配置。
 - a. 对于 Data points to alarm（触发告警的数据点数），指定有多少数据点触发您的告警进入 ALARM 状态。如果指定匹配的值，则如果多个连续评估期违例，该告警将变为 ALARM 状态。要创建 M-out-of-N（M（最大为 N））告警，为第一个值指定的数字应小于为第二个值指定的数字。有关更多信息，请参阅[使用 Amazon CloudWatch 告警](#)。
 - b. 对于 Missing data treatment（缺失数据处理），选择一个选项以指定在评估告警时如何应对缺失数据。
10. 选择 Next（下一步）。
11. 对于 Notification（通知），选择当您的告警处于 ALARM、OK 或 INSUFFICIENT_DATA 状态时要通知的 Amazon SNS 主题。
 - a. （可选）要为相同告警状态或不同告警状态发送多个通知，请选择 Add notification（添加通知）。
 - b. （可选）要想不发送通知，请选择 Remove（删除）。
12. （可选）如果您希望告警为 Amazon EC2 Auto Scaling、Amazon EC2、票证或 Amazon Systems Manager 执行操作，请选择相应的按钮，并指定告警状态和操作。

Note

您的告警只有在处于 ALARM 状态时才能执行 Systems Manager 操作。有关 Systems Manager 操作的信息，请参阅[将 CloudWatch 配置为创建 OpsItems 和事件创建](#)。

13. 选择 Next（下一步）。
14. 对于名称和描述，输入告警的名称和描述。
15. 对于 Preview and create（预览并创建），确保您的配置正确，然后选择 Create alarm（创建警告）。

根据指标数学表达式创建 CloudWatch 告警

要基于指标数学表达式创建告警，请选择表达式中要使用的一个或多个 CloudWatch 指标。然后，指定表达式、阈值和评估期。

无法创建基于 SEARCH (搜索) 表达式的告警。这是因为搜索表达式可返回多个时间序列，而基于数学表达式的告警只能监视一个时间序列。

根据指标数学表达式创建告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Alarms (告警)，然后选择 All alarms (所有告警)。
3. 选择 Create alarm (创建警报)。
4. 选择 Select Metric (选择指标)，然后执行以下操作之一：
 - 从 Amazon namespaces (命名空间) 下拉列表或 Custom namespaces (自定义命名空间) 下拉列表中，选择命名空间。选择命名空间后，您可以继续选择选项，直到显示指标列表，您可以在其中选择正确指标旁边的复选框。
 - 使用搜索框查找指标、维度或资源 ID。输入指标、维度或资源 ID 后，您可以继续选择选项，直到显示指标列表，您可以在其中选择正确指标旁边的复选框。
5. (可选) 如果您当前的 Amazon 账户是监控账户，并且您将跨账户功能添加到 CloudWatch 控制台中，您可以使用搜索框查找包含您想要告警监控的指标的 Amazon 账户。有关跨账户功能的更多信息，请参阅 [跨账户跨区域的 CloudWatch 控制台 \(p. 654\)](#)。

Note

目前，您可以创建跨账户告警，但不能创建跨区域告警。

6. (可选) 如果要向指标数学表达式添加另一个指标，您可以使用搜索框查找特定指标。您可以向指标数学表达式添加多达 10 个指标。
7. 选择 Graphed metrics (绘制的指标) 选项卡。对于您之前添加的每个指标，请执行以下操作：
 - a. 在 Statistic (统计数据) 列中，选择下拉菜单。在下拉菜单中，选择一个预定义的统计数据或百分位数。使用下拉菜单中的搜索框指定自定义百分位数。
 - b. 在 Period (时间段) 列下，选择下拉菜单。在下拉菜单中，选择一个预定义的评估期。

在创建告警时，您可以指定 Y 轴图例是在左侧还是右侧显示。

Note

在 CloudWatch 评估告警时，时间段会聚合到单个数据点中。

8. 选择 Add math (添加数学) 下拉菜单，然后从预定义的指标数学表达式列表中选择 Start with an empty expression (从空表达式开始)。

选择 Start with an empty expression (从空表达式开始) 之后，系统将显示一个数学表达式框，您可以在其中应用或编辑数学表达式。

9. 在数学表达式框中，输入数学表达式，然后选择 Apply (应用)。

选择 Apply (应用) 之后，Label (标注) 列旁边会显示一个 ID 列。

要将一个指标或另一个指标数学表达式的结果用作当前数学表达式公式的一部分，您可以使用 ID 列下显示的值。要更改 ID 值，您可以选择当前值旁边的笔和纸图标。新值必须以小写字母开头，并且可以包含数字、字母和下划线符号。将 ID 值更改为更明显的名称也可以使告警图表更易于理解。

有关可用于指标数学的函数的信息，请参阅 [指标数学语法和函数 \(p. 117\)](#)。

10. (可选) 在新数学表达式的公式中，使用指标和其他数学表达式的结果添加更多数学表达式。
11. 当您具有要用于告警的表达式时，请清除页面上每个其他表达式和每个指标左侧的复选框。只应选中要用于告警的表达式旁边的复选框。您为告警选择的表达式必须生成单个时间序列，并且仅在图表上显示一行。然后选择 Select metric (选择指标)。

将显示 Specify metric and conditions (指定指标和条件) 页面，其中显示一个图表以及有关您选择的数学表达式的其他信息。

12. 对于 Whenever **expression** is (每当表达式) , 指定表达式是必须大于、小于还是等于阈值。在...下面, 指定阈值。
13. 选择其他配置。对于触发警报的数据点数, 指定必须有多少个评估期 (数据点) 处于 ALARM 状态才能触发警报。如果此处的两个值匹配, 则会创建一个告警; 如果多个连续评估期违例, 该告警将变为 ALARM (告警) 状态。

要创建“M (最大为 N)”告警, 为第一个值指定的数字应小于为第二个值指定的数字。有关更多信息, 请参阅[评估告警 \(p. 152\)](#)。

14. 对于缺失数据处理, 选择在缺失某些数据点时的警报行为。有关更多信息, 请参阅[配置 CloudWatch 告警处理缺失数据的方式 \(p. 153\)](#)。
15. 选择 Next (下一步)。
16. 在通知下面, 选择一个在警报处于 ALARM、OK 或 INSUFFICIENT_DATA 状态时通知的 SNS 主题。

要使告警为相同告警状态或不同告警状态发送多个通知, 请选择 Add notification (添加通知) 。

要让警报不发送通知, 请选择删除。

17. 要让告警执行 Auto Scaling、EC2 或 Systems Manager 操作, 请选择相应的按钮, 然后选择告警状态和要执行的操作。告警只有在进入“ALARM (告警)”状态时才能执行 Systems Manager 操作。有关 Systems Manager 操作的更多信息, 请参阅[将 CloudWatch 配置为通过告警创建 OpsItems 和事件创建](#)。

Note

要创建执行 SSM Incident Manager 操作的告警, 您必须具有特定的权限。有关更多信息, 请参阅[Amazon Systems Manager Incident Manager 的基于身份的策略示例](#)。

18. 在完成后, 选择下一步。
19. 输入警报的名称和说明。名称只能包含 ASCII 字符。然后选择下一步。
20. 在 Preview and create 下面, 确认具有所需的信息和条件, 然后选择 Create alarm。

您还可以将警报添加到控制面板。有关更多信息, 请参阅[在 CloudWatch 控制面板上添加或删除警报小组件 \(p. 52\)](#)。

创建复合告警

复合告警通过监控其他告警的状态来确定其状态。您可以使用复合告警来降低告警噪音。例如, 您可以创建一个复合告警, 在该告警中, 基础指标告警在满足特定条件时会进入 ALARM 状态。然后, 您可以将复合告警设置为进入 ALARM 状态, 并在基础指标告警进入 ALARM 状态时向您发送通知, 方法是将基础指标告警配置为永不采取行动。目前, 复合告警可执行以下操作:

- 通知 SNS 主题
- 在 Amazon Systems Manager Ops Center 中创建 OpsItems
- 在 Amazon Systems Manager Incident Manager 中创建事件

Note

复合告警中的所有基础告警都必须与您的复合告警位于相同的 Amazon 账户和区域中。单个复合告警可以监控 100 个基础告警, 150 个复合告警可以监控单个基础告警。

规则表达式

所有复合告警都包含规则表达式。规则表达式告诉复合告警要监控哪些其他告警并确定其状态。规则表达式可以同时引用指标告警和复合告警。当您在规则表达式中引用告警时, 您可以为告警指定一个函数, 用于确定告警将处于以下三种状态中的哪一种:

- 告警

如果告警处于 ALARM 状态，则 ALARM (“alarm-name 或 alarm-ARN”) 为 TRUE。

- 确定

如果告警处于 ALARM 状态，则 OK (“alarm-name 或 alarm-ARN”) 为 TRUE。

- INSUFFICIENT_DATA

如果指定的告警处于 INSUFFICIENT_DATA 状态，则 INSUFFICIENT_DATA (“alarm-name or alarm-ARN”) 为 TRUE。

Note

TRUE 始终计算为 TRUE，FALSE 始终计算为 FALSE。

表达式示例

请求参数 AlarmRule 支持使用逻辑运算符 AND、OR 和 NOT，这样您就可以将多个函数组合成一个表达式。以下示例表达式显示了如何在复合告警中配置基础告警：

- ALARM(CPUUtilizationTooHigh) AND ALARM(DiskReadOpsTooHigh)

该表达式指定复合告警仅在 CPUUtilizationTooHigh 和 DiskReadOpsTooHigh 处于 ALARM 状态时进入 ALARM 状态。

- ALARM(CPUUtilizationTooHigh) AND NOT ALARM(DeploymentInProgress)

该表达式指定复合告警在 CPUUtilizationTooHigh 处于 ALARM 且 DeploymentInProgress 处于 ALARM 状态时进入 ALARM 状态。这是一个复合告警的示例，它降低了部署时段内的告警噪音。

- (ALARM(CPUUtilizationTooHigh) OR ALARM(DiskReadOpsTooHigh)) AND OK(NetworkOutTooHigh)

该表达式指定复合告警在 (ALARM(CPUUtilizationTooHigh) 或 (DiskReadOpsTooHigh) 处于 ALARM 且 (NetworkOutTooHigh) 处于 OK 状态时进入 ALARM 状态。这是一个复合告警的示例，如果任何一个基础告警在发生网络问题时不处于 ALARM 状态，则它不会向您发送通知，从而降低告警噪音。

创建复合告警

创建复合告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Alarms (告警)，然后选择 In alarm (告警中)。
3. 在告警列表中，选中要在规则表达式中引用的每个现有告警旁边的复选框，然后选择 Create composite alarm (创建复合告警)。
4. 在 Specify composite alarm conditions (指定复合告警条件) 中，指定新复合告警的规则表达式。

Note

您从告警列表中选择的告警将自动列在 Conditions (条件) 框中。默认情况下，ALARM 函数已指定给您的每个告警，并且每个告警都由逻辑运算符 OR 连接。

您可以使用以下子步骤修改规则表达式：

- a. 您可以将每个告警的所需状态从 ALARM 更改为 OK 或 INSUFFICIENT_DATA。
- b. 您可以将规则表达式中的逻辑运算符从 OR 更改为 AND 或 NOT，并且您可以添加括号来对函数进行分组。

- c. 您可以在规则表达式中包含其他告警，也可以从规则表达式中删除告警。

Example: Rule expression with conditions (示例：带条件的规则表达式)

```
(ALARM("CPUUtilizationTooHigh") OR  
ALARM("DiskReadOpsTooHigh")) AND  
OK("NetworkOutTooHigh")
```

复合告警在 ALARM ("CPUUtilizationTooHigh") 或 ALARM ("DiskReadOpsTooHigh") 处于 ALARM 状态且同时 OK ("NetworkOutTooHigh") 处于 OK 状态时进入 ALARM 的情况下的示例规则表达式。

5. 在完成后，选择下一步。
6. 在 Configure actions (配置操作) 下，您可以从以下选项中进行选择：

对于 Notification (通知)

- Select an existing SNS topic (选择一个现有的 SNS 主题)、Create a new SNS topic (创建新 SNS 主题)，或者 Use a topic ARN (使用主题 ARN) 定义将接收通知的 SNS 主题。
- Add notification (添加通知)，以便为相同告警状态或不同告警状态发送多个通知。
- Remove (删除) 以阻止您的告警发送通知或采取措施。

对于 Systems Manager action (Systems Manager 操作)

- Add Systems Manager action (添加 Systems Manager 操作)，以便您的告警可以在进入 ALARM 状态时执行 SSM 操作。

要了解有关 Systems Manager 操作的更多信息，请参阅《Amazon Systems Manager 用户指南》中的[配置 CloudWatch 以从告警中创建 OpsItems](#)和《Incident Manager 用户指南》中的[事件创建](#)。要创建执行 SSM Incident Manager 操作的告警，您必须具有正确的权限。有关更多信息，请参阅《Incident Manager 用户指南》中的[Amazon Systems Manager Incident Manager 的基于身份的策略示例](#)。

7. 在完成后，选择下一步。
8. 在 Add name and description (添加名称和描述) 下，为您的新复合告警输入告警名称和可选描述。告警名称只能包含 ASCII 字符。
9. 在完成后，选择下一步。
10. 在 Preview and create (预览和创建) 下，确认您的信息，然后选择 Create composite alarm (创建复合告警)。

Note

您可以创建一个复合告警循环，其中一个复合告警和另一个复合告警相互依赖。如果处于这种情况，则复合告警将停止被评估，并且您无法删除复合告警，因为它们相互依赖。要打破复合告警之间的依赖循环，最简单的方法是将您的其中一个复合告警中的函数 AlarmRule 更改为 False。

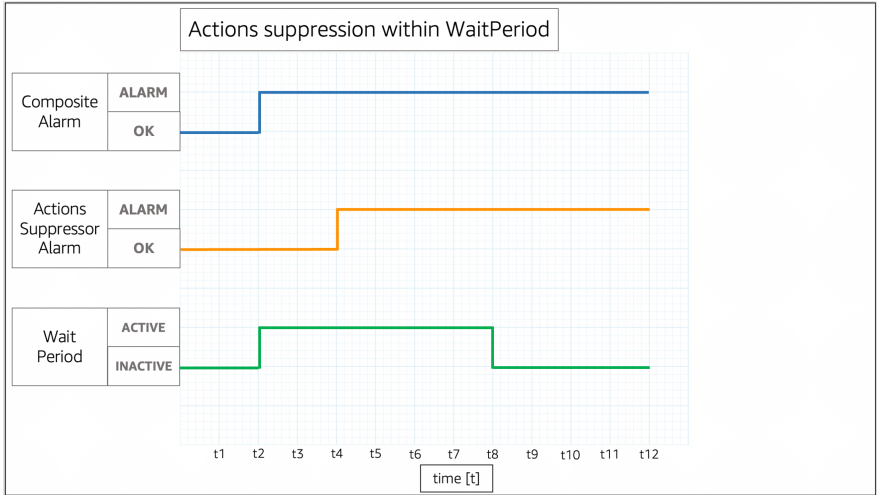
复合告警操作抑制

通过复合告警操作抑制，您可以将告警定义为抑制器告警。抑制器告警可防止复合告警采取行动。例如，您可以指定表示支持资源状态的抑制器告警。如果支持资源关闭，则抑制器告警会阻止复合告警发送通知。复合告警操作抑制功能可帮助您降低警报噪音，从而减少管理告警的时间，将更多的时间集中在操作上。

您可以在配置复合告警时指定抑制器告警。任何告警都可以用作抑制器告警。当抑制器告警的状态从 OK 变为 ALARM 时，其复合告警停止执行操作。当抑制器告警的状态从 ALARM 变为 OK 时，其复合告警恢复执行操作。

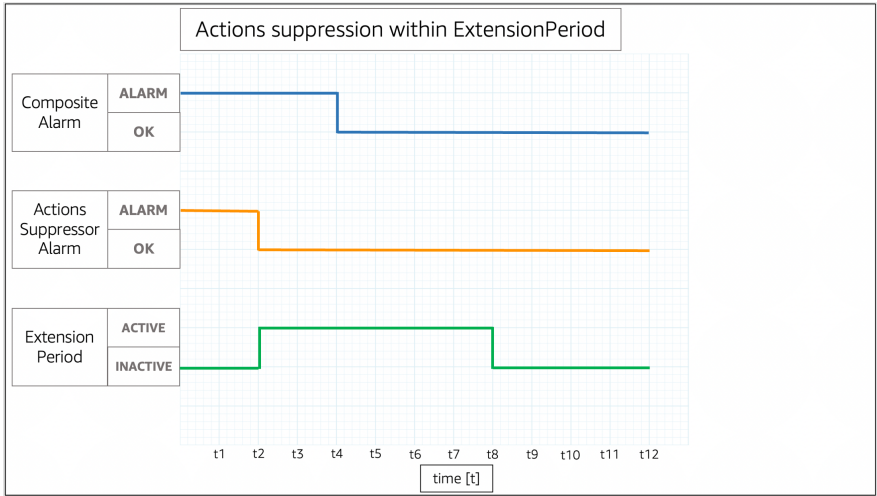
WaitPeriod 和 ExtensionPeriod

指定抑制器告警时，需要设置参数 `WaitPeriod` 和 `ExtensionPeriod`。这些参数可防止复合告警在抑制器告警改变状态时意外执行操作。使用 `WaitPeriod` 补偿当抑制器告警从 `OK` 变为 `ALARM` 时发生的任何延迟。例如，如果抑制器告警在 60 秒内从 `OK` 变为 `ALARM`，则将 `WaitPeriod` 设置为 60 秒。



在图像中，复合告警在 t_2 时从 `OK` 变为 `ALARM`。`WaitPeriod` 在 t_2 时开始并在 t_8 时结束。这使抑制器告警有时间在 t_4 时从状态 `OK` 变为 `ALARM`，然后在 `WaitPeriod` 在 t_8 时到期时抑制复合告警的操作。

使用 `ExtensionPeriod` 补偿当抑制器告警变为 `OK` 后复合告警变为 `OK` 时发生的任何延迟。例如，如果复合告警在抑制器告警变为 `OK` 的 60 秒内变为 `OK`，则将 `ExtensionPeriod` 设置为 60 秒。



在图像中，抑制器告警在 t_2 时从 `ALARM` 变为 `OK`。`ExtensionPeriod` 在 t_2 时开始并在 t_8 时结束。这使得复合告警有从 `ALARM` 变为 `OK` 的时间，然后 `ExtensionPeriod` 在 t_8 时到期。

复合告警在 `WaitPeriod` 和 `ExtensionPeriod` 变为活动状态时不执行操作。当 `ExtensionPeriod` 和 `WaitPeriod` 变为非活动状态时，复合告警基于其当前状态执行操作。我们建议您将每个参数的值设置为 60 秒，因为 CloudWatch 每分钟评估一次指标告警。您可以将参数设置为任意整数（以秒为单位）。

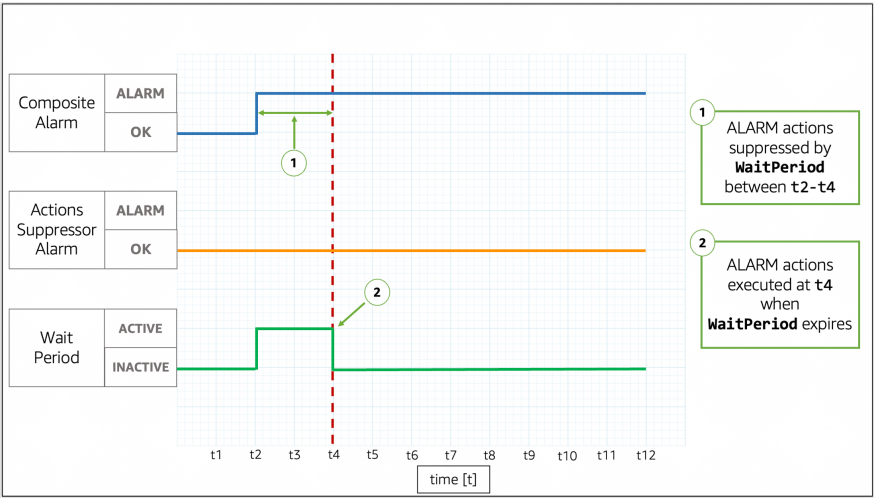
以下示例更详细地描述了 `WaitPeriod` 和 `ExtensionPeriod` 如何防止复合告警意外执行操作。

Note

在以下示例中，`WaitPeriod` 配置为 2 个时间单位，`ExtensionPeriod` 配置为 3 个时间单位。

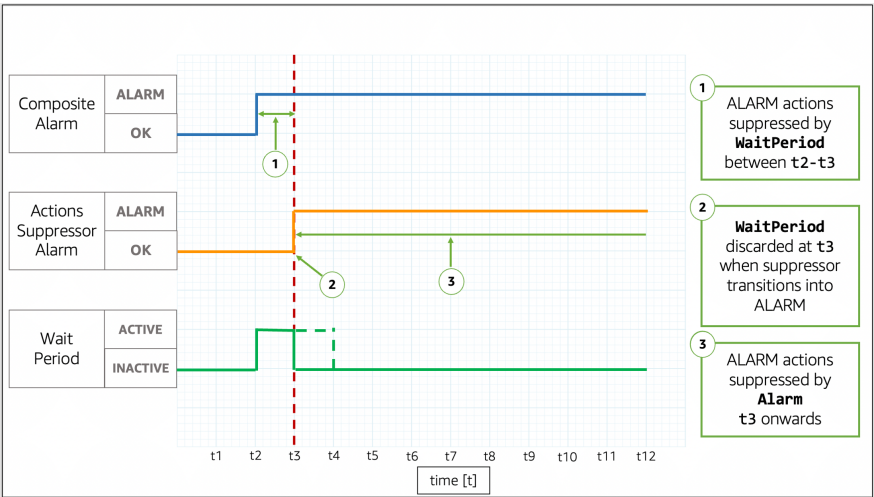
示例

Example 1: Actions are not suppressed after **WaitPeriod** (示例 1 : 操作在 **WaitPeriod** 后未被抑制)



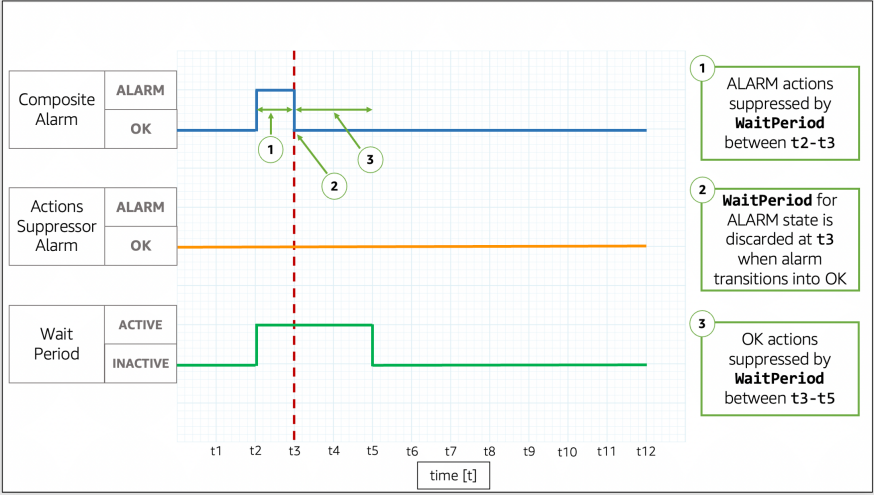
在图像中，复合告警在 t2 时状态从 OK 变为 ALARM。WaitPeriod 在 t2 时开始并在 t4 时结束，因此它可以阻止复合告警执行操作。WaitPeriod 在 t4 时到期后，复合告警会执行操作，因为抑制器告警仍处于 OK 状态。

Example 2: Actions are suppressed by alarm before **WaitPeriod** expires (示例 2 : 操作在 **WaitPeriod** 到期前被告警抑制)



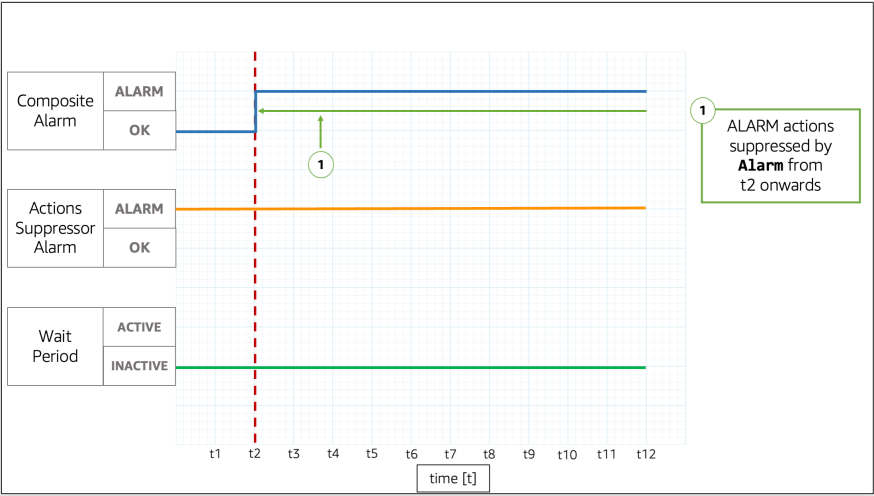
在图像中，复合告警在 t2 时状态从 OK 变为 ALARM。WaitPeriod 在 t2 时开始并在 t4 时结束。这使抑制器告警有时间在 t3 时从状态 OK 变为 ALARM。因为抑制器告警的状态在 t3 时从 OK 变为 ALARM，从 t2 开始的 WaitPeriod 将被丢弃，抑制器告警现在会阻止复合告警执行操作。

Example 3: State transition when actions are suppressed by **WaitPeriod** (示例 3 : 操作被 **waitPeriod** 抑制时的状态转换)



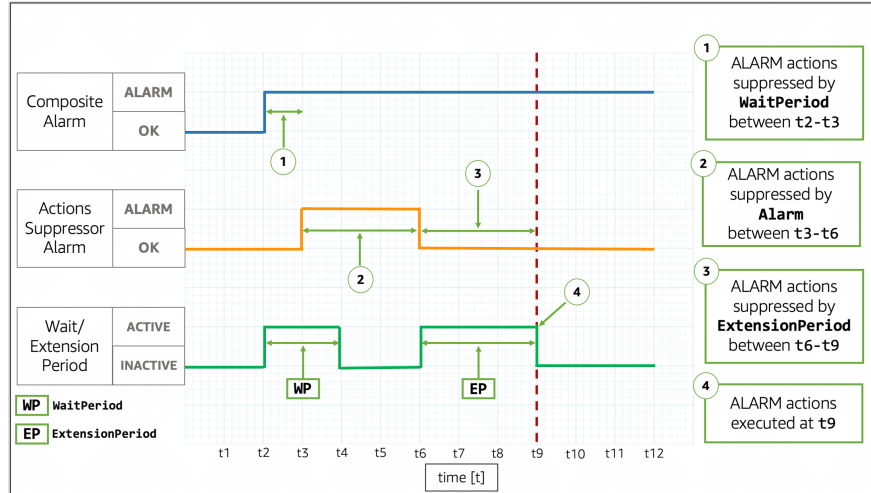
在图像中，复合告警在 t2 时状态从 OK 变为 ALARM。WaitPeriod 在 t2 时开始并在 t4 时结束。这使抑制器告警有时间改变状态。复合告警在 t3 时变回 OK，所以在 t2 时开始的 WaitPeriod 被丢弃。新的 WaitPeriod 在 t3 时开始并在 t5 时结束。新的 WaitPeriod 在 t5 时到期后，复合告警将执行操作。

Example 4: State transition when actions are suppressed by alarm (示例 4：操作被告警抑制时的状态转换)



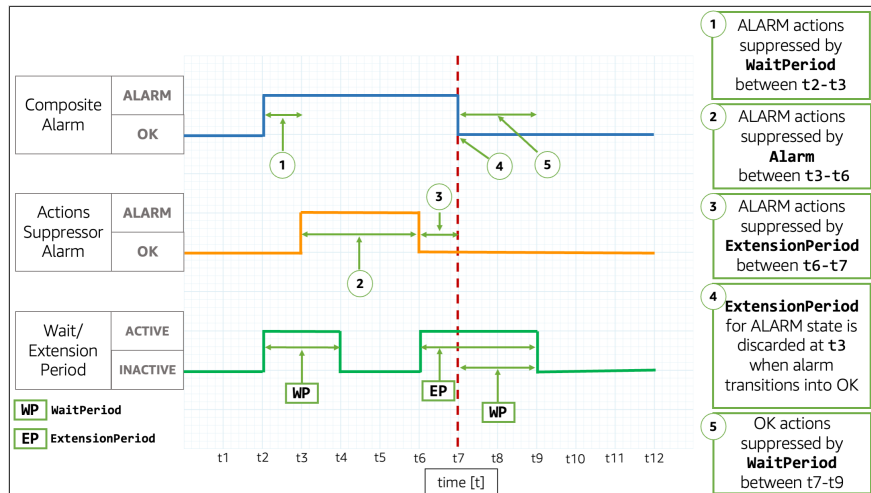
在图像中，复合告警在 t2 时状态从 OK 变为 ALARM。抑制器告警已经处于 ALARM 状态。抑制器告警可防止复合告警执行操作。

Example 5: Actions are not suppressed after **ExtensionPeriod** (示例 5：操作在 ExtensionPeriod 后未被抑制)



在图像中，复合告警在 t2 时状态从 OK 变为 ALARM。WaitPeriod 在 t2 时开始并在 t4 时结束。这使抑制器告警有时间在 t3 时从状态 OK 变为 ALARM，然后在 t6 前抑制复合告警的操作。因为抑制器告警的状态在 t3 时从 OK 变为 ALARM，从 t2 开始的 WaitPeriod 将被丢弃。在 t6 时，抑制器告警变为 OK。ExtensionPeriod 在 t6 时开始并在 t9 时结束。ExtensionPeriod 到期后，复合告警将执行操作。

Example 6: State transition when actions are suppressed by **ExtensionPeriod** (示例 6：操作被 **ExtensionPeriod** 抑制时的状态转换)



在图像中，复合告警在 t2 时状态从 OK 变为 ALARM。WaitPeriod 在 t2 时开始并在 t4 时结束。这使抑制器告警有时间在 t3 时从状态 OK 变为 ALARM，然后在 t6 前抑制复合告警的操作。因为抑制器告警的状态在 t3 时从 OK 变为 ALARM，从 t2 开始的 WaitPeriod 将被丢弃。在 t6 时，抑制器告警变回 OK。ExtensionPeriod 在 t6 时开始并在 t8 时结束。当复合告警在 t7 时变回 OK 时，ExtensionPeriod 被丢弃，并且新的 WaitPeriod 在 t7 时开始并在 t9 时结束。新的 WaitPeriod 到期后，复合告警可以执行其操作。

Tip

如果您更换操作抑制器告警，则任何活动的 WaitPeriod 或 ExtensionPeriod 将被丢弃。

编辑或删除 CloudWatch 告警

您可以编辑或删除现有告警。

您无法更改现有告警的名称。您可以复制一个告警，并为新告警指定不同的名称。要复制一个告警，请在告警列表中选中该告警名称旁边的复选框，然后选择 Action (操作) 和 Copy (复制)。

编辑告警

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Alarms (告警) 和 All alarms (所有告警)。
3. 选择警报的名称。
4. 选择 Edit (编辑)。

将显示 Specify metric and conditions (指定指标和条件) 页面，其中显示一个图表以及有关您选择的指标和统计数据的其他信息。

5. 要更改指标，请选择 Edit (编辑)，选择 All metrics (全部指标) 选项卡，然后执行以下操作之一：
 - 选择包含所需指标的服务命名空间。继续选择所显示的选项，以缩小选择范围。在显示指标列表时，选中所需的指标旁边的复选框。
 - 在搜索框中，输入指标名称、维度或资源 ID，然后按 Enter。接下来，选择其中的一个结果并继续，直到显示指标列表。选中所需的指标旁边的复选框。

选择选择指标。

6. 要更改告警的其他内容，请选择相应的选项。要更改必须有多少个数据点违例以使告警变为 ALARM (告警) 状态或更改缺失数据的处理方式，请选择 Additional configuration (其他配置)。
7. 选择 Next (下一步)。
8. 在 Notification (通知)、Auto Scaling action (Auto Scaling 操作) 和 EC2 action (EC2 操作) 下，编辑在触发告警时执行的操作 (可选)。然后选择下一步。
9. (可选) 更改告警描述。

您无法更改现有告警的名称。您可以复制一个告警，并为新告警指定不同的名称。要复制一个告警，请在告警列表中选中该告警名称旁边的复选框，然后选择 Action (操作) 和 Copy (复制)。

10. 选择 Next (下一步)。
11. 在 Preview and create (预览和创建) 下，确认具有所需的信息和条件，然后选择 Update alarm (更新告警)。

更新使用 Amazon SNS 控制台创建的电子邮件通知列表

1. 通过以下网址打开 Amazon SNS 控制台：<https://console.aws.amazon.com/sns/v3/home>。
2. 在导航窗格中，选择 Topics (主题)，然后选择通知列表 (主题) 的 ARN。
3. 请执行下列操作之一：
 - 要添加电子邮件地址，请选择 Create subscription (创建订阅)。对于协议，选择电子邮件。对于 Endpoint (端点)，输入新收件人的电子邮件地址。选择 Create subscription (创建订阅)。
 - 要删除电子邮件地址，请选择 Subscription ID (订阅 ID)。选择 Other subscription actions (其他订阅操作)、Delete subscriptions (删除订阅)。
4. 选择 Publish to topic (发布到主题)。

删除警报

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Alarms (警报)。
3. 选中告警名称左侧的复选框，然后依次选择 Actions (操作) 和 Delete (删除)。
4. 选择 Delete (删除)。

创建 CPU 使用率告警

您可以创建在告警状态从 OK (正常) 变为 ALARM (告警) 时使用 Amazon SNS 发送通知的 CloudWatch 告警。

在 EC2 实例的平均 CPU 使用率在指定的连续评估期内超出指定的阈值时，告警将变为 ALARM (告警) 状态。

使用 Amazon Web Services Management Console 设置 CPU 使用率告警

可以执行以下步骤以使用 Amazon Web Services Management Console 创建 CPU 使用率警报。

根据 CPU 利用率创建警报

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Alarms (告警) 和 All alarms (所有告警)。
3. 选择 Create alarm (创建警报)。
4. 选择选择指标。
5. 在 All metrics (所有指标) 选项卡中，选择 EC2 metrics (EC2 指标)。
6. 选择一个指标类别 (例如，Per-Instance Metrics (每个实例的指标))。
7. 找到您希望在 InstanceId (实例 ID) 列和 Metric Name (指标名称) 中的 CPUUtilization (CPU 使用率) 中列出的实例所在的行。选中此行旁边的复选框，然后选择 Select metric (选择指标)。
8. 在 Specify metric and conditions (指定指标和条件) 下，对于 Statistic (统计数据)，选择 Average (平均值)，然后选择一个预定义百分位数，或指定自定义百分位数 (例如 **p95.45**)。
9. 选择时间段 (例如 **5 minutes**)。
10. 在条件下面，指定以下内容：
 - a. 对于 Threshold type (阈值类型)，选择 Static (静态)。
 - b. 对于 Whenever CPUUtilization is (每当 CPU 利用率)，指定 Greater (大)。在 than... (于...) 下，指定在 CPU 使用率超过此百分比时触发告警进入“ALARM (告警)”状态的阈值。例如，70。
 - c. 选择其他配置。对于触发警报的数据点数，指定必须有多少个评估期 (数据点) 处于 ALARM 状态才能触发警报。如果此处的两个值匹配，则会创建一个告警；如果多个连续评估期违例，该告警将变为 ALARM (告警) 状态。

要创建“M (最大为 N)”告警，为第一个值指定的数字应小于为第二个值指定的数字。有关更多信息，请参阅 [评估告警 \(p. 152\)](#)。
 - d. 对于缺失数据处理，选择在缺失某些数据点时的警报行为。有关更多信息，请参阅 [配置 CloudWatch 告警处理缺失数据的方式 \(p. 153\)](#)。
 - e. 如果警报将百分比值作为监控的统计数据，将显示样本数少的百分比框。使用它来选择是评估还是忽略采样率低的案例。如果选择忽略 (保持警报状态)，在样本大小太小时，将始终保持当前警报状态。有关更多信息，请参阅 [基于百分位数的 CloudWatch 告警和小数据样本 \(p. 156\)](#)。
11. 选择 Next (下一步)。
12. 在 Notification (通知) 下，选择 In alarm (处于告警中)，并选择当告警处于 ALARM 状态时要通知的 SNS 主题

要使告警为相同告警状态或不同告警状态发送多个通知，请选择 Add notification (添加通知)。

要让警报不发送通知，请选择删除。

13. 在完成后，选择下一步。
14. 输入警报的名称和说明。名称只能包含 ASCII 字符。然后选择下一步。
15. 在 Preview and create 下面，确认具有所需的信息和条件，然后选择 Create alarm。

使用 Amazon CLI 设置 CPU 使用率告警

可以执行以下步骤以使用 Amazon CLI 创建 CPU 使用率警报。

根据 CPU 利用率创建警报

1. 设置 SNS 主题。有关更多信息，请参阅[设置 Amazon SNS 通知 \(p. 157\)](#)。
2. 使用 `put-metric-alarm` 命令创建告警，如下所示。

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
when CPU exceeds 70%" --metric-name CPUUtilization --namespace AWS/EC2 --statistic
Average --period 300 --threshold 70 --comparison-operator GreaterThanThreshold --
dimensions Name=InstanceId,Value=i-12345678 --evaluation-periods 2 --alarm-actions
arn:aws:sns:us-east-1:111122223333:my-topic --unit Percent
```

3. 通过使用 `set-alarm-state` 命令强制更改警报状态来测试警报。
 - a. 将告警状态从 `INSUFFICIENT_DATA` (数据不足) 更改为 `OK` (正常)。

```
aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason "initializing"
--state-value OK
```

- b. 将告警状态从 `OK` (正常) 更改为 `ALARM` (告警)。

```
aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason "initializing"
--state-value ALARM
```

- c. 检查您是否收到有关告警的电子邮件通知。

创建发送电子邮件的负载均衡器延迟告警

您可以设置 Amazon SNS 通知并配置告警，此告警监控经典负载均衡器超过 100 毫秒的延迟。

使用 Amazon Web Services Management Console 设置延迟告警

可以执行以下步骤以使用 Amazon Web Services Management Console 创建负载均衡器延迟警报。

创建负载均衡器延迟告警

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Alarms (告警) 和 All alarms (所有告警)。
3. 选择 Create alarm (创建警报)。
4. 在 CloudWatch Metrics by Category (按类别显示的 CloudWatch 指标) 下，选择 ELB Metrics (ELB 指标) 类别。
5. 选择包含经典负载均衡器和 Latency (延迟) 指标的行。

6. 对于统计数据，选择 Average (平均值)，然后选择其中的一个预定义百分比值，或者指定一个自定义百分比值 (例如 **p95.45**)。
7. 对于时间段，选择 1 Minute (1 分钟)。
8. 选择 Next (下一步)。
9. 在警报阈值下面，输入警报的唯一名称 (例如，**myHighCpuAlarm**) 和警报描述 (例如，**Alarm when Latency exceeds 100s**)。警报名称必须仅包含 ASCII 字符。
10. 在 Whenever (每当) 下，对于 is (是)，选择 > 并输入 **0.1**。对于 for (持续时间)，输入 **3**。
11. 在 Additional settings (附加设置) 下，对于 Treat missing data as (将缺失的数据作为以下内容处理) 选择 ignore (maintain alarm) (忽略 (保持告警状态))，以使缺失数据点不会触发告警状态更改。

对于 Percentiles with low samples (样本数少的百分比)，选择 ignore (maintain the alarm state) (忽略 (保持告警状态))，使告警只评估具有充足数量的数据样本的情况。

12. 在 Actions (操作) 下，为 Whenever this alarm (每当此告警) 选择 State is ALARM (状态为“告警”)。对于 Send notification to (发送通知到)，选择一个现有 SNS 主题或创建一个新 SNS 主题。

要创建 SNS 主题，请选择 New list (新列表)。对于 Send notification to (发送通知到) 输入 SNS 主题的名称 (例如 **myHighCpuAlarm**)，对于 Email list (电子邮件列表) 输入在告警状态变为 ALARM (告警) 时通知的电子邮件地址列表 (以逗号分隔)。将向每个电子邮件地址发送一封主题订阅确认电子邮件。您必须先确认订阅，然后才会发送通知。

13. 选择 Create Alarm (创建告警)。

使用 Amazon CLI 设置延迟告警

可以执行以下步骤以使用 Amazon CLI 创建负载均衡器延迟警报。

创建负载均衡器延迟告警

1. 设置 SNS 主题。有关更多信息，请参阅[设置 Amazon SNS 通知 \(p. 157\)](#)。
2. 使用 `put-metric-alarm` 命令创建告警，如下所示：

```
aws cloudwatch put-metric-alarm --alarm-name lb-mon --alarm-description "Alarm when Latency exceeds 100s" --metric-name Latency --namespace AWS/ELB --statistic Average --period 60 --threshold 100 --comparison-operator GreaterThanThreshold --dimensions Name=LoadBalancerName,Value=my-server --evaluation-periods 3 --alarm-actions arn:aws:sns:us-east-1:111122223333:my-topic --unit Seconds
```

3. 通过使用 `set-alarm-state` 命令强制更改警报状态来测试警报。
 - a. 将告警状态从 INSUFFICIENT_DATA (数据不足) 更改为 OK (正常)。

```
aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --state-value OK
```

- b. 将告警状态从 OK (正常) 更改为 ALARM (告警)。

```
aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --state-value ALARM
```

- c. 检查您是否收到有关告警的电子邮件通知。

创建发送电子邮件的存储吞吐量告警

您可以设置 SNS 通知并配置告警，该告警在 Amazon EBS 超过 100MB 吞吐量时触发。

使用 Amazon Web Services Management Console 设置存储吞吐量告警

可以执行以下步骤，使用 Amazon Web Services Management Console 创建基于 Amazon EBS 吞吐量的告警。

创建存储吞吐量告警

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Alarms (告警) 和 All alarms (所有告警)。
3. 选择 Create alarm (创建警报)。
4. 在 EBS Metrics (EBS 指标) 下，选择一个指标类别。
5. 选择包含卷和 VolumeWriteBytes 指标的行。
6. 对于统计数据，选择 Average (平均值)。对于时间段，选择 5 Minutes (5 分钟)。选择 Next (下一步)。
7. 在警报阈值下面，输入警报的唯一名称 (例如，**myHighWriteAlarm**) 和警报描述 (例如，**VolumeWriteBytes exceeds 100,000 KiB/s**)。警报名称必须仅包含 ASCII 字符。
8. 在 Whenever (每当) 下，对于 is (是)，选择 > 并输入 **100000**。对于 for (持续时间)，输入 **15** 个连续评估期。

Alarm Preview (告警预览) 下会显示阈值的图形表示。

9. 在 Additional settings (附加设置) 下，对于 Treat missing data as (将缺失的数据作为以下内容处理) 选择 ignore (maintain alarm) (忽略 (保持告警状态))，以使缺失数据点不会触发告警状态更改。
10. 在 Actions (操作) 下，为 Whenever this alarm (每当此告警) 选择 State is ALARM (状态为“告警”)。对于 Send notification to (发送通知到)，选择现有的 SNS 主题或创建一个主题。

要创建 SNS 主题，请选择 New list (新列表)。对于 Send notification to (发送通知到) 输入 SNS 主题的名称 (例如 **myHighCpuAlarm**)，对于 Email list (电子邮件列表) 输入在告警状态变为 ALARM (告警) 时通知的电子邮件地址列表 (以逗号分隔)。将向每个电子邮件地址发送一封主题订阅确认电子邮件。您必须先确认订阅，然后才能将通知发送到电子邮件地址。

11. 选择 Create Alarm (创建告警)。

使用 Amazon CLI 设置存储吞吐量告警

可以执行以下步骤，使用 Amazon CLI 创建基于 Amazon EBS 吞吐量的告警。

创建存储吞吐量告警

1. 创建 SNS 主题。有关更多信息，请参阅[设置 Amazon SNS 通知 \(p. 157\)](#)。
2. 创建告警。

```
aws cloudwatch put-metric-alarm --alarm-name ebs-mon --alarm-description "Alarm when EBS volume exceeds 100MB throughput" --metric-name VolumeReadBytes --namespace AWS/EBS --statistic Average --period 300 --threshold 100000000 --comparison-operator GreaterThanThreshold --dimensions Name=VolumeId,Value=my-volume-id --evaluation-periods 3 --alarm-actions arn:aws:sns:us-east-1:111122223333:my-alarm-topic --insufficient-data-actions arn:aws:sns:us-east-1:111122223333:my-insufficient-data-topic
```

3. 通过使用 [set-alarm-state](#) 命令强制更改警报状态来测试警报。
 - a. 将告警状态从 INSUFFICIENT_DATA (数据不足) 更改为 OK (正常)。

```
aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason "initializing"
--state-value OK
```

- b. 将告警状态从 OK (正常) 更改为 ALARM (告警)。

```
aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason "initializing"
--state-value ALARM
```

- c. 将告警状态从 ALARM (正常) 更改为 INSUFFICIENT_DATA (告警)。

```
aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason "initializing"
--state-value INSUFFICIENT_DATA
```

- d. 检查您是否收到有关告警的电子邮件通知。

创建告警以停止、终止、重启或恢复 EC2 实例

利用 Amazon CloudWatch 告警操作，您可创建自动停止、终止、重启或恢复 EC2 实例的告警。当不再需要某个实例运行时，您可使用停止或终止操作来帮助您节省资金。如果发生了系统损害，您可使用重启和恢复操作自动重启这些实例或将它们恢复到新硬件上。

在许多情况下，您可能需要自动终止或停止实例。例如，您可能拥有专用于批工资单处理作业或科学计算任务的实例，这些实例在运行一段时间后就完成了其工作。与其让这些实例空闲（并产生费用），不如将其停止或终止以帮助节省开支。使用停止警报操作和终止警报操作的主要区别是，停止的警报可以在以后需要运行时轻松重启。您还可以保留相同的实例 ID 和根卷。而终止的实例则无法重新启动。如此就必须启动一个新的实例。

您可以向为 Amazon EC2 每个实例指标设置的任何告警添加停止、终止或重启操作，这些指标包括 Amazon CloudWatch 提供的基本和详细监控指标（在亚马逊云科技/EC2 命名空间中），以及包含“InstanceId=”维度的任何自定义指标，只要 InstanceId 值引用有效运行的 Amazon EC2 实例。您还可以将恢复操作添加到针对任何 Amazon EC2 每个实例指标设置的告警中，但以下情况除外：StatusCheckFailed_Instance。

要设置可重启、停止或终止实例的 CloudWatch 告警操作，您必须使用服务相关的 IAM 角色 `AWSServiceRoleForCloudWatchEvents`。AWSServiceRoleForCloudWatchEvents IAM 角色允许 Amazon 代表您执行告警操作。

要为 CloudWatch Events 创建服务相关角色，请使用以下命令：

```
aws iam create-service-linked-role --aws-service-name events.amazonaws.com
```

控制台支持

您可使用 CloudWatch 控制台或 Amazon EC2 控制台创建告警。本文档中的程序使用 CloudWatch 控制台。有关使用 CloudWatch 控制台的程序，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的[创建停止、终止、重启或恢复实例的告警](#)。

权限

如果您使用 Amazon Identity and Access Management (IAM) 账户来创建或修改执行 EC2 操作或 Systems Manager OpsItem 操作的告警，您必须拥有 `iam:CreateServiceLinkedRole` 权限。

目录

- 在 Amazon CloudWatch 告警中添加停止操作 (p. 179)
- 在 Amazon CloudWatch 告警中添加终止操作 (p. 179)
- 在 Amazon CloudWatch 告警中添加重启操作 (p. 180)
- 在 Amazon CloudWatch 告警中添加恢复操作 (p. 181)

- [查看已触发的告警和操作的历史记录 \(p. 182\)](#)

在 Amazon CloudWatch 告警中添加停止操作

可以创建当达到一定阈值后停止 Amazon EC2 实例的警报。例如，您可能运行了开发或测试实例而偶尔忘记将其关闭。可以创建当平均 CPU 使用率低于 10% 达 24 小时触发的警报，同时告知其为空闲并不再使用。可以根据需要调整阈值、时长和时间段，还可以添加 SNS 通知，以便您在触发警报后能够收到电子邮件。

可以停止或终止将 Amazon Elastic Block Store 卷用作根设备的 Amazon EC2 实例，但只能终止将实例存储用作根设备的实例。

使用 Amazon CloudWatch 控制台创建停止空闲实例的告警

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Alarms (警报) 和 All alarms (所有警报)。
3. 选择创建警报。
4. 选择 Select Metric (选择指标)。
5. 对于 Amazon namespaces (Amazon 命名空间)，请选择 EC2。
6. 执行以下操作：
 - a. 选择 Per-Instance Metrics (每个实例的指标)。
 - b. 选择包含正确实例和 CPUUtilization 指标的行中的复选框。
 - c. 选择 Graphed metrics (绘制的指标)。
 - d. 对于统计数据，选择 Average (平均值)。
 - e. 选择时间段 (例如 1 Hour)。
 - f. 选择选择指标。
7. 对于 Define Alarm 步骤，执行以下操作：
 - a. 在 Conditions (条件) 下，选择 Static (静态)。
 - b. 在 Whenever CPUUtilization is (每当 CPUUtilization) 下，选择 Lower (降低)。
 - c. 对于 than (比较)，键入 10。
 - d. 选择 Next (下一步)。
 - e. 在 Notification 下，为 Send notification to 选择一个现有 SNS 主题或创建一个新 SNS 主题。

要创建 SNS 主题，请选择 New list (新列表)。对于发送通知到，键入 SNS 主题的名称 (例如 "Stop_EC2_Instance")。对于 Email list (电子邮件列表)，请键入警报变为 ALARM 状态时将通知发送到的电子邮件地址列表 (以逗号分隔)。将向每个电子邮件地址发送一封主题订阅确认电子邮件。您必须先确认订阅，然后才能将通知发送到电子邮件地址。
 - f. 选择 Add EC2 Action (添加 EC2 操作)。
 - g. 对于 Alarm state trigger (告警状态触发器)，选择 In alarm (在告警中)。对于 Take the following action (请执行以下操作)，选择 Stop this instance (停止此实例)。
 - h. 选择 Next (下一步)。
 - i. 输入警报的名称和说明。名称只能包含 ASCII 字符。然后选择下一步。
 - j. 在 Preview and create 下面，确认具有所需的信息和条件，然后选择 Create alarm。

在 Amazon CloudWatch 告警中添加终止操作

可以创建当达到一定阈值时自动终止 EC2 实例的警报 (只要该实例未启用终止保护)。例如，某个实例已经完成工作，您不再需要此实例而希望将其终止。如果可能在之后使用该实例，则应该选择停止而不是终止。有

关于实例启用和禁用终止保护的信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的[为实例启用终止保护](#)。

使用 Amazon CloudWatch 控制台创建终止空闲实例的告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Alarms 和 Create Alarm。
3. 对于 Select Metric 步骤，执行以下操作：
 - a. 在 EC2 Metrics 下，选择 Per-Instance Metrics。
 - b. 选择包含实例和 CPUUtilization 指标的行。
 - c. 对于统计数据，选择 Average (平均值)。
 - d. 选择时间段 (例如 1 Hour)。
 - e. 选择 Next (下一步)。
4. 对于 Define Alarm 步骤，执行以下操作：
 - a. 在 Alarm Threshold 下，键入警报的唯一名称 (例如“Terminate EC2 instance”) 和警报的描述 (例如“Terminate EC2 instance when CPU is idle for too long”)。警报名称必须仅包含 ASCII 字符。
 - b. 在每当下，为是选择 < 并键入 10。对于对于，键入 24 作为连续时段数。

Alarm Preview (告警预览) 下会显示阈值的图形表示。
 - c. 在 Notification 下，为 Send notification to 选择一个现有 SNS 主题或创建一个新 SNS 主题。

要创建 SNS 主题，请选择 New list (新列表)。对于发送通知到，键入 SNS 主题的名称 (例如“Terminate_EC2_Instance”)。对于 Email list (电子邮件列表)，请键入警报变为 ALARM 状态时将通知发送到的电子邮件地址列表 (以逗号分隔)。将向每个电子邮件地址发送一封主题订阅确认电子邮件。您必须先确认订阅，然后才能将通知发送到电子邮件地址。
 - d. 选择 EC2 Action。
 - e. 对于每当此警报，请选择状态为“警报”。对于 Take this action，选择 Terminate this instance。
 - f. 选择 Create Alarm (创建告警)。

在 Amazon CloudWatch 告警中添加重启操作

您可创建监控 Amazon EC2 实例并自动重启此实例的 Amazon CloudWatch 警报。在实例运行状况检查失败时，推荐重启警报操作 (与恢复警报操作相反，该操作适合系统运行状况检查失败的情况)。实例重启相当于操作系统重启。在许多情况下，只需要几分钟时间即可重启您的实例。重启实例时，其仍驻留在相同的物理主机上，因此您的实例将保留其公有 DNS 名称、私有 IP 地址及其实例存储卷上的任何数据。

重启实例不会启动新的实例计费时间，这与停止并重新启动您的实例不同。有关重新启动实例的更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的[重启实例](#)。

Important

为了避免重启操作与恢复操作之间的竞争情况，请避免为重启警报和恢复警报设置相同的评估期。我们建议您将重启警报设置为 3 个 1 分钟的评估期。

使用 Amazon CloudWatch 控制台创建重启实例的告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Alarms 和 Create Alarm。
3. 对于 Select Metric 步骤，执行以下操作：
 - a. 在 EC2 Metrics 下，选择 Per-Instance Metrics。
 - b. 选择包含实例和 StatusCheckFailed_Instance 指标的行。

- c. 对于统计数据，选择 Minimum。
- d. 选择一个时间段（例如，1 Minute），然后选择下一步。
4. 对于 Define Alarm 步骤，执行以下操作：
 - a. 在 Alarm Threshold 下，键入警报的唯一名称（例如“Reboot EC2 instance”）和警报的描述（例如“Reboot EC2 instance when health checks fail”）。警报名称必须仅包含 ASCII 字符。
 - b. 在每当下，为是选择 > 并键入 0。对于对于，键入 3 作为连续时段数。

Alarm Preview（告警预览）下会显示阈值的图形表示。
 - c. 在 Notification 下，为 Send notification to 选择一个现有 SNS 主题或创建一个新 SNS 主题。

要创建 SNS 主题，请选择 New list（新列表）。对于发送通知到，键入 SNS 主题的名称（例如“Reboot_EC2_Instance”）。对于 Email list（电子邮件列表），请键入警报变为 ALARM 状态时将通知发送到的电子邮件地址列表（以逗号分隔）。将向每个电子邮件地址发送一封主题订阅确认电子邮件。您必须先确认订阅，然后才能将通知发送到电子邮件地址。
 - d. 选择 EC2 Action。
 - e. 对于每当此警报，请选择状态为“警报”。对于 Take this action，选择 Reboot this instance。
 - f. 选择 Create Alarm（创建告警）。

在 Amazon CloudWatch 告警中添加恢复操作

您可以创建 Amazon CloudWatch 警报用于监控 Amazon EC2 实例，并且在实例受损（由于发生底层硬件故障或需要 Amazon 参与才能修复的问题）时自动恢复实例。无法恢复终止的实例。恢复的实例与原始实例相同，包括实例 ID、私有 IP 地址、弹性 IP 地址以及所有实例元数据。

当 `StatusCheckFailed_System` 告警触发且恢复操作启动时，您在创建告警及相关恢复操作时所选择的 Amazon SNS 主题将向您发出通知。在实例恢复过程中，实例将在重启时迁移，并且内存中的所有数据都将丢失。当该过程完成后，会向您已配置警报的 SNS 主题发布信息。任何订阅此 SNS 主题的用户都将收到一封电子邮件通知，其中包括恢复尝试的状态以及任何进一步的指示。您会注意到，实例在已恢复的实例上重启。

恢复操作仅适用于 `StatusCheckFailed_System`，而不能用于 `StatusCheckFailed_Instance`。

导致系统状态检查出现故障的问题示例包括：

- 网络连接丢失
- 系统电源损耗
- 物理主机上的软件问题
- 物理主机上影响到网络连接状态的硬件问题

恢复操作仅受到以下项支持：

- A1、C3、C4、C5、C5a、C5n、C6g、Inf1、M3、M4、M5、M5a、M5n、M5zn、M6g、P3、P4、R3、R4、R5、R5a 和 X2iedn 实例类型
- VPC 中的实例
- 具有 `default` 或 `dedicated` 实例租赁的实例
- 仅使用 Amazon EBS 卷的实例（不配置实例存储卷）

如果您的实例具有公有 IPv4 地址，它会在恢复后保留公有 IPv4 地址。

Important

如果实例包含附加的存储卷，则数据将在恢复过程中丢失。

Important

为了避免重启操作与恢复操作之间的竞争情况，请避免为重启警报和恢复警报设置相同的评估期。我们建议您将恢复警报设置为 2 个 1 分钟的评估期，并将重启警报设置为 3 个 1 分钟的评估期。

使用 Amazon CloudWatch 控制台创建恢复实例的告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Alarms 和 Create Alarm。
3. 对于 Select Metric 步骤，执行以下操作：
 - a. 在 EC2 Metrics 下，选择 Per-Instance Metrics。
 - b. 选择包含实例和 StatusCheckFailed_System 指标的行。
 - c. 对于统计数据，选择 Minimum。
 - d. 选择时间段（例如 **1 Minute**）。

Important

为了避免重启操作与恢复操作之间的竞争情况，请避免为重启警报和恢复警报设置相同的评估期。我们建议您将恢复警报设置为 2 个 1 分钟的评估期。

- e. 选择 Next (下一步)。
4. 对于 Define Alarm 步骤，执行以下操作：
 - a. 在 Alarm Threshold 下，键入警报的唯一名称（例如“Recover EC2 instance”）和警报的描述（例如“Recover EC2 instance when health checks fail”）。警报名称必须仅包含 ASCII 字符。
 - b. 在每当下，为是选择 > 并键入 0。对于对于，键入 2 作为连续时段数。
 - c. 在 Notification 下，为 Send notification to 选择一个现有 SNS 主题或创建一个新 SNS 主题。

要创建 SNS 主题，请选择 New list（新列表）。タイプ: 整数。对于 Email list（电子邮件列表），请键入警报变为 ALARM 状态时将通知发送到的电子邮件地址列表（以逗号分隔）。将向每个电子邮件地址发送一封主题订阅确认电子邮件。您必须先确认订阅，然后才能将通知发送到电子邮件地址。
 - d. 选择 EC2 Action。
 - e. 对于每当此警报，请选择状态为“警报”。对于 Take this action，选择 Recover this instance。
 - f. 选择 Create Alarm（创建告警）。

查看已触发的告警和操作的历史记录

您可以在 Amazon CloudWatch 控制台中查看警报和操作历史记录。Amazon CloudWatch 会保留最近两周的警报和操作历史记录。

要查看已触发的警报和操作的历史记录

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Alarms（警报），然后选择一个警报。
3. 要查看最近的状态转换以及时间和指标值，请选择详细信息。
4. 要查看最近的历史记录条目，请选择 History（历史记录）。

创建账单告警以监控 Amazon 估算费用

您可以使用 Amazon CloudWatch 监控 Amazon 估算费用。在为您的 Amazon 账户启用估算费用监控时，将每天计算几次估算费用并作为指标数据发送到 CloudWatch。

当您的账户账单超过阈值时，将触发告警。它仅在当前账单超出阈值时触发。它不会根据您的本月迄今为止的使用情况进行预测。

如果您每当费用超出阈值时创建一个账单告警，则告警将立即变为 ALARM (告警) 状态。

Note

有关分析已计费的 CloudWatch 费用的信息，请参阅 [CloudWatch 账单和成本 \(p. 7\)](#)。

任务

- [启用账单提醒 \(p. 183\)](#)
- [创建账单告警 \(p. 183\)](#)
- [删除账单告警 \(p. 184\)](#)

启用账单提醒

在为估算费用创建告警之前，您必须启用账单提醒，以便监控 Amazon 估算费用并使用账单指标数据创建告警。在启用账单提醒后，您无法禁用数据收集，但可以删除创建的任何账单告警。

首次启用账单提醒后，大约需要 15 分钟时间，您就可以查看账单数据和设置账单告警。

要求

- 您必须使用账户根用户凭证或作为被授予权限的 IAM 用户登录，才能查看账单信息。
- 对于整合账单账户，每个关联账户的账单数据可以在付款账户登录后找到。您可以查看每个关联账户以及整合账户的估计费用总和，和各项服务的估计费用。
- 在整合账单账户中，仅当付款人账户启用 Receive Billing Alerts (接收账单提醒) 首选项时，才会捕获成员关联账户的指标。如果您更改了您的管理账户/付款人账户，则必须新的管理账户/付款人账户中启用账单提醒。
- 该账户不能属于 Amazon 合作伙伴网络 (APN)，因为对于 APN 账户，账单指标不会发布到 CloudWatch。有关更多信息，请参阅 [Amazon 合作伙伴网络](#)。

要启用预估收费监控

1. 访问 <https://console.aws.amazon.com/billing/>，打开 Amazon Billing 控制台。
2. 在导航窗格中，选择 Billing Preferences (账单首选项)。
3. 选择 Receive Billing Alerts (接收账单提醒)。
4. 选择 Save preferences (保存首选项)。

创建账单告警

Important

创建账单告警之前，您必须将区域设置为美国东部 (弗吉尼亚州北部)。账单指标数据存储在该区域中，并表示全球费用。您还必须为您的账户启用账单提醒；或者，如果您使用的是整合账单，则必须在管理账户/付款人账户中启用账单提醒。有关更多信息，请参阅 [启用账单提醒 \(p. 183\)](#)。

在该过程中，您可以创建一个告警，以便在 Amazon 的估算费用超出定义的阈值时发送通知。

使用 CloudWatch 控制台创建告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。

2. 在导航窗格中，选择 Alarms (告警)，然后选择 All alarms (所有告警)。
3. 选择 Create alarm (创建警报)。
4. 选择选择指标。在 Browse (浏览) 中，选择 Billing (账单)，然后选择 Total Estimated Charge (估计费用总额)。

Note

如果您没看到账单/估计费用总额指标，则启用账单提醒，并将您的区域更改为美国东部 (弗吉尼亚州北部)。有关更多信息，请参阅[启用账单提醒 \(p. 183\)](#)。

5. 选中 EstimatedCharges (估算费用) 指标的复选框，然后选择 Select metric (选择指标)。
6. 对于 Threshold type (阈值类型)，选择 Static (静态)。
7. 对于 Whenever EstimatedCharges is ... (当估算费用...)，选择 Greater (大)。
8. 对于 than ... (于...)，定义触发告警的阈值 (例如 **200 USD**)。

Note

定义阈值后，预览图显示您当月的预估费用。

9. 选择 Next (下一步)。
10. 在 Notification (通知) 下，选择当您的告警处于 ALARM 状态时要通知的 Amazon SNS 主题。您可以选择现有的 Amazon SNS 主题，创建一个新 Amazon SNS 主题，或使用主题 ARN 通知其他账户。如果您希望您的告警为相同告警状态或不同告警状态发送多个通知，请选择 Add notification (添加通知)。
11. 选择 Next (下一步)。
12. 在 Name and description (名称和描述) 下，为您的告警输入名称。
 - (可选) 输入告警的描述。
13. 在 Preview and create (预览并创建) 下，确保您的配置正确，然后选择 Create alarm (创建警告)。

删除账单告警

当您不再需要账单告警时，可将其删除。

删除账单告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Alarms (警报) 和 All alarms (所有警报)。
3. 选中告警旁的复选框，然后依次选择 Actions (操作) 和 Delete (删除)。
4. 当系统提示进行确认时，选择 Yes, Delete (是，删除)。

隐藏 Auto Scaling 警报

在 Amazon Web Services Management Console 中查看您的警报时，您可以隐藏与 Amazon EC2 Auto Scaling 相关的警报。仅在 Amazon Web Services Management Console 中提供了该功能。

若要临时隐藏 Auto Scaling 警报

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Alarms (告警) 和 All alarms (所有告警)，然后选择 Hide Auto Scaling alarms (隐藏弹性伸缩告警)。

告警事件和 EventBridge

每当 CloudWatch 告警被创建、更新、删除或更改告警状态时，CloudWatch 都会将事件发送到 Amazon EventBridge。您可以使用 EventBridge 和这些事件来编写规则，这些规则会在告警更改状态时采取措施，例如通知您。有关更多信息，请参阅[什么是 Amazon EventBridge？](#)

CloudWatch 保证向 EventBridge 传送告警状态更改事件。

来自 CloudWatch 的示例事件

本部分包括来自 CloudWatch 的示例事件。

单指标告警的状态更改

```
{
  "version": "0",
  "id": "c4c1c1c9-6542-e61b-6ef0-8c4d36933a92",
  "detail-type": "CloudWatch Alarm State Change",
  "source": "aws.cloudwatch",
  "account": "123456789012",
  "time": "2019-10-02T17:04:40Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh"
  ],
  "detail": {
    "alarmName": "ServerCpuTooHigh",
    "configuration": {
      "description": "Goes into alarm when server CPU utilization is too high!",
      "metrics": [
        {
          "id": "30b6c6b2-a864-43a2-4877-c09a1afc3b87",
          "metricStat": {
            "metric": {
              "dimensions": {
                "InstanceId": "i-12345678901234567"
              },
              "name": "CPUUtilization",
              "namespace": "AWS/EC2"
            },
            "period": 300,
            "stat": "Average"
          },
          "returnData": true
        }
      ]
    },
    "previousState": {
      "reason": "Threshold Crossed: 1 out of the last 1 datapoints [0.0666851903306472 (01/10/19 13:46:00)] was not greater than the threshold (50.0) (minimum 1 datapoint for ALARM -> OK transition).",
      "reasonData": "{\n\"version\": \"1.0\", \"queryDate\": \"2019-10-01T13:56:40.985+0000\", \"startDate\": \"2019-10-01T13:46:00.000+0000\", \"statistic\": \"Average\", \"period\": 300, \"recentDatapoints\": [0.0666851903306472], \"threshold\": 50.0}",
      "timestamp": "2019-10-01T13:56:40.987+0000",
      "value": "OK"
    },
    "state": {
      "reason": "Threshold Crossed: 1 out of the last 1 datapoints [99.50160229693434 (02/10/19 16:59:00)] was greater than the threshold (50.0) (minimum 1 datapoint for OK -> ALARM transition).",

```

```
        "reasonData": "{\\\"version\\\":\\\"1.0\\\",\\\"queryDate\\\":\\\"2019-10-02T17:04:40.985+0000\\\",\\\"startDate\\\":\\\"2019-10-02T16:59:00.000+0000\\\",\\\"statistic\\\":\\\"Average\\\",\\\"period\\\":300,\\\"recentDatapoints\\\":[99.50160229693434],\\\"threshold\\\":50.0}\",
        \"timestamp\": \"2019-10-02T17:04:40.989+0000\",
        \"value\": \"ALARM\"
    }
}
```

数学指标告警的状态更改

```
{
  \"version\": \"0\",
  \"id\": \"2dde0eb1-528b-d2d5-9ca6-6d590caf2329\",
  \"detail-type\": \"CloudWatch Alarm State Change\",
  \"source\": \"aws.cloudwatch\",
  \"account\": \"123456789012\",
  \"time\": \"2019-10-02T17:20:48Z\",
  \"region\": \"us-east-1\",
  \"resources\": [
    \"arn:aws:cloudwatch:us-east-1:123456789012:alarm:TotalNetworkTrafficTooHigh\"
  ],
  \"detail\": {
    \"alarmName\": \"TotalNetworkTrafficTooHigh\",
    \"configuration\": {
      \"description\": \"Goes into alarm if total network traffic exceeds 10Kb\",
      \"metrics\": [
        {
          \"expression\": \"SUM(METRICS())\",
          \"id\": \"e1\",
          \"label\": \"Total Network Traffic\",
          \"returnData\": true
        },
        {
          \"id\": \"m1\",
          \"metricStat\": {
            \"metric\": {
              \"dimensions\": {
                \"InstanceId\": \"i-12345678901234567\"
              },
              \"name\": \"NetworkIn\",
              \"namespace\": \"AWS/EC2\"
            },
            \"period\": 300,
            \"stat\": \"Maximum\"
          },
          \"returnData\": false
        },
        {
          \"id\": \"m2\",
          \"metricStat\": {
            \"metric\": {
              \"dimensions\": {
                \"InstanceId\": \"i-12345678901234567\"
              },
              \"name\": \"NetworkOut\",
              \"namespace\": \"AWS/EC2\"
            },
            \"period\": 300,
            \"stat\": \"Maximum\"
          },
          \"returnData\": false
        }
      ]
    }
  }
}
```

```

    },
    "previousState": {
      "reason": "Unchecked: Initial alarm creation",
      "timestamp": "2019-10-02T17:20:03.642+0000",
      "value": "INSUFFICIENT_DATA"
    },
    "state": {
      "reason": "Threshold Crossed: 1 out of the last 1 datapoints [45628.0 (02/10/19 17:10:00)] was greater than the threshold (10000.0) (minimum 1 datapoint for OK -> ALARM transition).",
      "reasonData": "{\n\"version\": \"1.0\", \"queryDate\": \"2019-10-02T17:20:48.551+0000\", \"startDate\": \"2019-10-02T17:10:00.000+0000\", \"period\": 300, \"recentDatapoints\": [45628.0], \"threshold\": 10000.0}",
      "timestamp": "2019-10-02T17:20:48.554+0000",
      "value": "ALARM"
    }
  }
}

```

异常检测告警的状态更改

```

{
  "version": "0",
  "id": "daafc9f1-bddd-c6c9-83af-74971fcfc4ef",
  "detail-type": "CloudWatch Alarm State Change",
  "source": "aws.cloudwatch",
  "account": "123456789012",
  "time": "2019-10-03T16:00:04Z",
  "region": "us-east-1",
  "resources": ["arn:aws:cloudwatch:us-east-1:123456789012:alarm:EC2 CPU Utilization Anomaly"],
  "detail": {
    "alarmName": "EC2 CPU Utilization Anomaly",
    "state": {
      "value": "ALARM",
      "reason": "Thresholds Crossed: 1 out of the last 1 datapoints [0.0 (03/10/19 15:58:00)] was less than the lower thresholds [0.020599444741798756] or greater than the upper thresholds [0.3006915352732461] (minimum 1 datapoint for OK -> ALARM transition).",
      "reasonData": "{\n\"version\": \"1.0\", \"queryDate\": \"2019-10-03T16:00:04.650+0000\", \"startDate\": \"2019-10-03T15:58:00.000+0000\", \"period\": 60, \"recentDatapoints\": [0.0], \"recentLowerThresholds\": [0.020599444741798756], \"recentUpperThresholds\": [0.3006915352732461]}",
      "timestamp": "2019-10-03T16:00:04.653+0000"
    },
    "previousState": {
      "value": "OK",
      "reason": "Thresholds Crossed: 1 out of the last 1 datapoints [0.1666666666664241 (03/10/19 15:57:00)] was not less than the lower thresholds [0.0206719426210418] or not greater than the upper thresholds [0.30076870222143803] (minimum 1 datapoint for ALARM -> OK transition).",
      "reasonData": "{\n\"version\": \"1.0\", \"queryDate\": \"2019-10-03T15:59:04.670+0000\", \"startDate\": \"2019-10-03T15:57:00.000+0000\", \"period\": 60, \"recentDatapoints\": [0.1666666666664241], \"recentLowerThresholds\": [0.0206719426210418], \"recentUpperThresholds\": [0.30076870222143803]}",
      "timestamp": "2019-10-03T15:59:04.672+0000"
    },
    "configuration": {
      "description": "Goes into alarm if CPU Utilization is out of band",
      "metrics": [{
        "id": "m1",
        "metricStat": {
          "metric": {
            "namespace": "AWS/EC2",
            "name": "CPUUtilization",
            "dimensions": {

```

```
        "InstanceId": "i-12345678901234567"
      },
      {
        "period": 60,
        "stat": "Average"
      },
      {
        "returnData": true
      },
      {
        "id": "ad1",
        "expression": "ANOMALY_DETECTION_BAND(m1, 0.8)",
        "label": "CPUUtilization (expected)",
        "returnData": true
      }
    ]
  }
}
```

State change for a composite alarm with a suppressor alarm (带有抑制器告警的复合告警的状态变化)

```
{
  "version": "0",
  "id": "d3dfc86d-384d-24c8-0345-9f7986db0b80",
  "detail-type": "CloudWatch Alarm State Change",
  "source": "aws.cloudwatch",
  "account": "123456789012",
  "time": "2022-07-22T15:57:45Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
  ],
  "detail": {
    "alarmName": "ServiceAggregatedAlarm",
    "state": {
      "actionsSuppressedBy": "WaitPeriod",
      "actionsSuppressedReason": "Actions suppressed by WaitPeriod",
      "value": "ALARM",
      "reason": "arn:aws:cloudwatch:us-east-1:123456789012:alarm:SuppressionDemo.EventBridge.FirstChild transitioned to ALARM at Friday 22 July, 2022 15:57:45 UTC",
      "reasonData": "{\n  \"triggeringAlarms\": [\n    {\n      \"arn\": \"arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh\",\n      \"state\": {\n        \"value\": \"ALARM\",\n        \"timestamp\": \"2022-07-22T15:57:45.394+0000\"\n      }\n    }\n  ]\n}",
      "timestamp": "2022-07-22T15:57:45.394+0000"
    },
    "previousState": {
      "value": "OK",
      "reason": "arn:aws:cloudwatch:us-east-1:123456789012:alarm:SuppressionDemo.EventBridge.Main was created and its alarm rule evaluates to OK",
      "reasonData": "{\n  \"triggeringAlarms\": [\n    {\n      \"arn\": \"arn:aws:cloudwatch:us-east-1:123456789012:alarm:TotalNetworkTrafficTooHigh\",\n      \"state\": {\n        \"value\": \"OK\",\n        \"timestamp\": \"2022-07-14T16:28:57.770+0000\"\n      }\n    },\n    {\n      \"arn\": \"arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh\",\n      \"state\": {\n        \"value\": \"OK\",\n        \"timestamp\": \"2022-07-14T16:28:54.191+0000\"\n      }\n    }\n  ]\n}",
      "timestamp": "2022-07-22T15:56:14.552+0000"
    },
    "configuration": {
      "alarmRule": "ALARM(ServerCpuTooHigh) OR ALARM(TotalNetworkTrafficTooHigh)",
      "actionsSuppressor": "ServiceMaintenanceAlarm",
      "actionsSuppressorWaitPeriod": 120,
      "actionsSuppressorExtensionPeriod": 180
    }
  }
}
```

复合告警的创建

```
{
  "version": "0",
  "id": "91535fdd-1e9c-849d-624b-9a9f2b1d09d0",
  "detail-type": "CloudWatch Alarm Configuration Change",
  "source": "aws.cloudwatch",
  "account": "123456789012",
  "time": "2022-03-03T17:06:22Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
  ],
  "detail": {
    "alarmName": "ServiceAggregatedAlarm",
    "operation": "create",
    "state": {
      "value": "INSUFFICIENT_DATA",
      "timestamp": "2022-03-03T17:06:22.289+0000"
    },
    "configuration": {
      "alarmRule": "ALARM(ServerCpuTooHigh) OR ALARM(TotalNetworkTrafficTooHigh)",
      "alarmName": "ServiceAggregatedAlarm",
      "description": "Aggregated monitor for instance",
      "actionsEnabled": true,
      "timestamp": "2022-03-03T17:06:22.289+0000",
      "okActions": [],
      "alarmActions": [],
      "insufficientDataActions": []
    }
  }
}
```

Creation of a composite alarm with a suppressor alarm (带有抑制器告警的复合告警的创建)

```
{
  "version": "0",
  "id": "454773e1-09f7-945b-aa2c-590af1c3f8e0",
  "detail-type": "CloudWatch Alarm Configuration Change",
  "source": "aws.cloudwatch",
  "account": "123456789012",
  "time": "2022-07-14T13:59:46Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
  ],
  "detail": {
    "alarmName": "ServiceAggregatedAlarm",
    "operation": "create",
    "state": {
      "value": "INSUFFICIENT_DATA",
      "timestamp": "2022-07-14T13:59:46.425+0000"
    },
    "configuration": {
      "alarmRule": "ALARM(ServerCpuTooHigh) OR ALARM(TotalNetworkTrafficTooHigh)",
      "actionsSuppressor": "ServiceMaintenanceAlarm",
      "actionsSuppressorWaitPeriod": 120,
      "actionsSuppressorExtensionPeriod": 180,
      "alarmName": "ServiceAggregatedAlarm",
      "actionsEnabled": true,
      "timestamp": "2022-07-14T13:59:46.425+0000",
      "okActions": [],
      "alarmActions": [],
    }
  }
}
```

```

        "insufficientDataActions": []
    }
}

```

指标告警的更新

```

{
  "version": "0",
  "id": "bc7d3391-47f8-ae47-f457-1b4d06118d50",
  "detail-type": "CloudWatch Alarm Configuration Change",
  "source": "aws.cloudwatch",
  "account": "123456789012",
  "time": "2022-03-03T17:06:34Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh"
  ],
  "detail": {
    "alarmName": "ServerCpuTooHigh",
    "operation": "update",
    "state": {
      "value": "INSUFFICIENT_DATA",
      "timestamp": "2022-03-03T17:06:13.757+0000"
    },
    "configuration": {
      "evaluationPeriods": 1,
      "threshold": 80,
      "comparisonOperator": "GreaterThanOrEqualToThreshold",
      "treatMissingData": "ignore",
      "metrics": [
        {
          "id": "86bfa85f-b14c-ebf7-8916-7da014ce23c0",
          "metricStat": {
            "metric": {
              "namespace": "AWS/EC2",
              "name": "CPUUtilization",
              "dimensions": {
                "InstanceId": "i-12345678901234567"
              }
            },
            "period": 300,
            "stat": "Average"
          },
          "returnData": true
        }
      ],
      "alarmName": "ServerCpuTooHigh",
      "description": "Goes into alarm when server CPU utilization is too high!",
      "actionsEnabled": true,
      "timestamp": "2022-03-03T17:06:34.267+0000",
      "okActions": [],
      "alarmActions": [],
      "insufficientDataActions": []
    },
    "previousConfiguration": {
      "evaluationPeriods": 1,
      "threshold": 70,
      "comparisonOperator": "GreaterThanOrEqualToThreshold",
      "treatMissingData": "ignore",
      "metrics": [
        {
          "id": "d6bfa85f-893e-b052-a58b-4f9295c9111a",
          "metricStat": {

```



```
        "metric": {
            "namespace": "AWS/EC2",
            "name": "CPUUtilization",
            "dimensions": {
                "InstanceId": "i-12345678901234567"
            }
        },
        "period": 300,
        "stat": "Average"
    },
    "returnData": true
}
},
"alarmName": "ServerCpuTooHigh",
"description": "Goes into alarm when server CPU utilization is too high!",
"actionsEnabled": true,
"timestamp": "2022-03-03T17:06:13.757+0000",
"okActions": [],
"alarmActions": [],
"insufficientDataActions": []
}
}
}
```

Update of a composite alarm with a suppressor alarm (带有抑制器告警的复合告警的更新)

```
{
    "version": "0",
    "id": "4c6f4177-6bd5-c0ca-9f05-b4151c54568b",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-14T13:59:56Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "update",
        "state": {
            "actionsSuppressedBy": "WaitPeriod",
            "value": "ALARM",
            "timestamp": "2022-07-14T13:59:46.425+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 360,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:56.290+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        },
        "previousConfiguration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 180,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:46.425+0000",

```

```
        "okActions": [],
        "alarmActions": [],
        "insufficientDataActions": []
    }
}
```

指标数学告警的删除

```
{
  "version": "0",
  "id": "f171d220-9e1c-c252-5042-2677347a83ed",
  "detail-type": "CloudWatch Alarm Configuration Change",
  "source": "aws.cloudwatch",
  "account": "123456789012",
  "time": "2022-03-03T17:07:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudwatch:us-east-1:123456789012:alarm:TotalNetworkTrafficTooHigh"
  ],
  "detail": {
    "alarmName": "TotalNetworkTrafficTooHigh",
    "operation": "delete",
    "state": {
      "value": "INSUFFICIENT_DATA",
      "timestamp": "2022-03-03T17:06:17.672+0000"
    },
    "configuration": {
      "evaluationPeriods": 1,
      "threshold": 10000,
      "comparisonOperator": "GreaterThanOrEqualToThreshold",
      "treatMissingData": "ignore",
      "metrics": [
        {
          "id": "m1",
          "metricStat": {
            "metric": {
              "namespace": "AWS/EC2",
              "name": "NetworkIn",
              "dimensions": {
                "InstanceId": "i-12345678901234567"
              }
            },
            "period": 300,
            "stat": "Maximum"
          },
          "returnData": false
        },
        {
          "id": "m2",
          "metricStat": {
            "metric": {
              "namespace": "AWS/EC2",
              "name": "NetworkOut",
              "dimensions": {
                "InstanceId": "i-12345678901234567"
              }
            },
            "period": 300,
            "stat": "Maximum"
          },
          "returnData": false
        }
      ]
    }
  }
}
```

```
        "id": "e1",
        "expression": "SUM(METRICS())",
        "label": "Total Network Traffic",
        "returnData": true
    }
],
"alarmName": "TotalNetworkTrafficTooHigh",
"description": "Goes into alarm if total network traffic exceeds 10Kb",
"actionsEnabled": true,
"timestamp": "2022-03-03T17:06:17.672+0000",
"okActions": [],
"alarmActions": [],
"insufficientDataActions": []
}
}
```

Deletion of a composite alarm with a suppressor alarm (带有抑制器告警的复合告警的删除)

```
{
  "version": "0",
  "id": "e34592a1-46c0-b316-f614-1b17a87be9dc",
  "detail-type": "CloudWatch Alarm Configuration Change",
  "source": "aws.cloudwatch",
  "account": "123456789012",
  "time": "2022-07-14T14:00:01Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
  ],
  "detail": {
    "alarmName": "ServiceAggregatedAlarm",
    "operation": "delete",
    "state": {
      "actionsSuppressedBy": "WaitPeriod",
      "value": "ALARM",
      "timestamp": "2022-07-14T13:59:46.425+0000"
    },
    "configuration": {
      "alarmRule": "ALARM(ServerCpuTooHigh) OR ALARM(TotalNetworkTrafficTooHigh)",
      "actionsSuppressor": "ServiceMaintenanceAlarm",
      "actionsSuppressorWaitPeriod": 120,
      "actionsSuppressorExtensionPeriod": 360,
      "alarmName": "ServiceAggregatedAlarm",
      "actionsEnabled": true,
      "timestamp": "2022-07-14T13:59:56.290+0000",
      "okActions": [],
      "alarmActions": [],
      "insufficientDataActions": []
    }
  }
}
```

应用程序监控

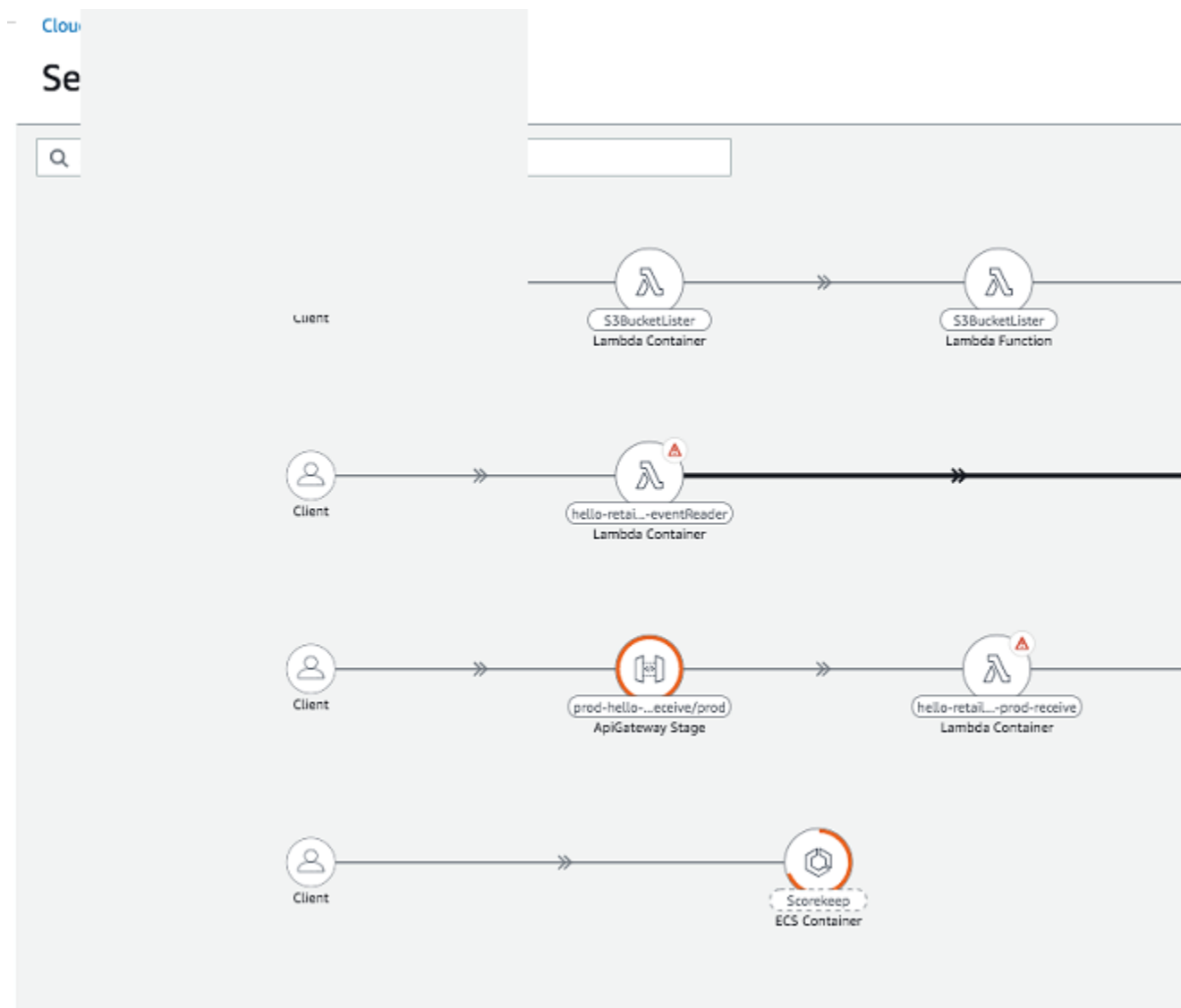
本部分中的主题介绍了 CloudWatch 的功能，这些功能可以帮助您获得对 Amazon 资源的操作了解。

主题

- [使用 ServiceLens 监控应用程序的运行状况 \(p. 194\)](#)
- [使用 Synthetics 监控 \(p. 208\)](#)
- [使用 CloudWatch Evidently 执行启动和 A/B 实验 \(p. 292\)](#)
- [使用 CloudWatch Rum \(p. 323\)](#)

使用 ServiceLens 监控应用程序的运行状况

CloudWatch ServiceLens 使您能够将跟踪、指标、日志、告警和其他资源运行状况信息整合到一个位置，从而提高服务和应用程序的可观察性。ServiceLens 将 CloudWatch 与 Amazon X-Ray 集成，提供应用程序的端到端视图，帮助您更高效地查明性能瓶颈并确定受影响的用户。服务地图将服务终端节点和资源显示为“节点”，并突出显示每个节点及其连接的流量、延迟和错误。您可以选择一个节点来查看与该服务部分相关的指标、日志和跟踪的详细见解。这使您能够调查问题及其对应用程序产生的影响。



要充分利用 ServiceLens 和相关的指标、日志和跟踪，您必须更新 X-Ray 开发工具包和应用程序分析。ServiceLens 支持 Lambda 函数、API Gateway、运行于 Amazon EC2 上的基于 Java 的应用程序以及运行于部署了 Container Insights 的 Amazon EKS 或 Kubernetes 上的基于 Java 的应用程序的日志关联。

ServiceLens 与 Amazon CloudWatch Synthetics 集成，后者是一项完全托管式服务，使您能够创建 Canary 以从外到内监控端点和 API。您创建的 Canary 将显示在 ServiceLens 服务地图上。有关更多信息，请参阅[使用 Synthetics 监控 \(p. 208\)](#)。

ServiceLens 在 X-Ray 可用的每个区域都可用。

主题

- [部署 ServiceLens \(p. 196\)](#)
- [在 ServiceLens 中使用服务地图 \(p. 203\)](#)
- [使用 ServiceLens 中的跟踪视图 \(p. 205\)](#)
- [使用 ServiceLens 中的资源运行状况视图 \(p. 205\)](#)
- [ServiceLens 问题排查 \(p. 207\)](#)

部署 ServiceLens

部署 ServiceLens 需要执行两个步骤：

- 部署 Amazon X-Ray 以便查看服务地图。
- 部署 CloudWatch 代理和 X-Ray 守护进程，以支持服务地图与 CloudWatch 指标和 CloudWatch Logs 集成。

主题

- [部署 Amazon X-Ray \(p. 196\)](#)
- [部署 CloudWatch 代理和 X-Ray 守护进程 \(p. 197\)](#)

部署 Amazon X-Ray

您可以使用任何 Amazon X-Ray 开发工具包启用 X-Ray。但仅在您使用 Java 软件开发工具包时支持日志和指标与跟踪的关联。

请遵循标准 X-Ray 设置部署 X-Ray。有关更多信息，请参阅下列内容：

- [适用于 Java 的 Amazon X-Ray 开发工具包](#) (支持日志关联)
- [适用于 Node.js 的 X-Ray 软件开发工具包](#)
- [适用于 .NET 的 Amazon X-Ray 开发工具包](#)
- [适用于 Go 的 Amazon X-Ray 开发工具包](#)
- [适用于 Python 的 Amazon X-Ray 开发工具包](#)
- [Amazon X-Ray SDK for Ruby](#)

完成 X-Ray 的设置后，请执行以下部分中的步骤来将 X-Ray 与 CloudWatch Logs 集成并启用段指标。

主题

- [与 CloudWatch Logs 集成 \(p. 196\)](#)
- [启用 X-Ray 中的段指标 \(p. 197\)](#)

与 CloudWatch Logs 集成

要启用与 CloudWatch Logs 的集成，需要执行两个步骤：

- 启用跟踪以记录关联。这仅支持使用适用于 Java 的开发工具包。
- 配置跟踪 ID 注入。

启用跟踪以记录日志关联

适用于 Java 的开发工具包支持一组标准应用程序日志记录框架和 CloudWatch Logs 本机支持。在完成以下步骤之前，您必须已完成 Amazon X-Ray SDK for Java 的标准设置。

支持的运行时为 Amazon EC2、已启用 CloudWatch Container Insights 的 Amazon ECS 和 Lambda。

- 要启用跟踪以记录 Amazon EC2 上的日志关联，请启用 X-Ray EC2 插件。有关更多信息，请参阅[服务插件](#)
- 要启用跟踪以记录 Amazon EKS 上的日志关联，请先启用 Container Insights (如果尚未执行此操作)。有关更多信息，请参阅[使用 Container Insights \(p. 353\)](#)。

然后，启用 X-Ray 开发工具包 EKS 插件。有关更多信息，请参阅[服务插件](#)。

- 要启用跟踪以记录 Lambda 上的日志关联，您必须在 Lambda 上启用 X-Ray。有关更多信息，请参阅[Amazon Lambda](#) 和 [Amazon X-Ray](#)。

启用跟踪 ID 注入

有关如何启用跟踪 ID 注入的信息，请参阅[日志记录](#)。

启用 X-Ray 中的段指标

Amazon X-Ray SDK for Java 可以将有关段的多个指标发送到 CloudWatch，以提供延迟、限制、错误和故障率的未采样视图。它使用 CloudWatch 代理发出这些指标，以最大程度地减少对应用程序性能的影响。有关分段的更多信息，请参阅[分段](#)。

如果启用段指标，则会创建一个名为 XRayApplicationMetrics 的日志组，指标 ErrorRate、FaultRate、ThrottleRate 和 Latency 也会发布到名为 Observability 的自定义 CloudWatch 指标命名空间中。

Lambda 当前不支持段指标。

要启用 Amazon X-Ray SDK for Java 以发布段指标，请使用以下示例。

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withSegmentListener(new MetricsSegmentListener());
```

如果您将 ServiceLens 与 Amazon EKS 和 Container Insights 结合使用，请将 AWS_XRAY_METRICS_DAEMON_ADDRESS 环境变量添加到 HOST_IP，如下示例所示。

```
env:
- name: HOST_IP
  valueFrom:
    fieldRef:
      apiVersion: v1
      fieldPath: status.hostIP
- name: AWS_XRAY_METRICS_DAEMON_ADDRESS
  value: ${HOST_IP}:25888
```

有关更多信息，请参阅[启用 X-Ray CloudWatch 指标](#)。

部署 CloudWatch 代理和 X-Ray 守护进程

本部分介绍了如何部署 CloudWatch 代理和 X-Ray 守护进程。您可以在以下环境中部署代理和守护程序：

- Amazon ECS 或 Fargate
- Amazon EKS 或托管在 Amazon EC2 上的 Kubernetes
- Amazon EC2

以下部分介绍了这些环境的部署步骤。

主题

- [在 Amazon ECS 上部署 CloudWatch 代理和 X-Ray 守护进程 \(p. 198\)](#)
- [在 Amazon EKS 或 Kubernetes 上部署 CloudWatch 代理和 X-Ray 守护进程 \(p. 201\)](#)

- 在 Amazon EC2 上部署 CloudWatch 代理和 X-Ray 守护进程 (p. 203)

在 Amazon ECS 上部署 CloudWatch 代理和 X-Ray 守护进程

在 Amazon ECS 上，您可以将 CloudWatch 代理作为附加项部署到应用程序容器来收集指标。您可以通过 SSM 参数仓库来配置 CloudWatch 代理。

创建 IAM 角色

您必须创建两个 IAM 角色。如果您已创建这些角色，则可能需要向其添加权限。

- ECS 任务角色 – 容器使用此角色运行。权限应是您的应用程序所需的任何权限，以及 CloudWatchAgentServerPolicy 和 AWSXRayDaemonWriteAccess。
- ECS 任务执行角色— Amazon ECS 使用此角色启动和执行您的容器。如果您已创建此角色，请将 AmazonSSMReadOnlyAccess、AmazonECSTaskExecutionRolePolicy 和 CloudWatchAgentServerPolicy 策略附加到此角色。

如果您需要存储更敏感的数据以供 Amazon ECS 使用，请参阅[指定敏感数据](#)以了解更多信息。

有关创建 IAM 角色的更多信息，请参阅[创建 IAM 角色](#)。

将代理配置存储在 SSM 参数仓库中

您需要确保您的代理配置文件具有以下部分，然后将它上载到 SSM 参数仓库。

```
{
  "logs": {
    "metrics_collected": {
      "emf": {}
    }
  }
}
```

将代理配置上载到 SSM 参数仓库中

1. 将代理配置内容放入本地文件 /tmp/ecs-cwagent.json 中
2. 输入以下命令。将 **region** 替换为集群的区域。

```
aws ssm put-parameter \
--name "ecs-cwagent" \
--type "String" \
--value "`cat /tmp/ecs-cwagent.json`" \
--region "region"
```

创建任务定义及启动任务

此任务的步骤取决于您要使用的是 EC2 启动类型还是 Fargate 启动类型。

EC2 启动类型

首先，创建任务定义。在此示例中，容器“demo-app”向 CloudWatch 代理发送 X-Ray 开发工具包指标，并向 X-Ray 守护进程发送跟踪信息。

将以下任务定义复制到本地 JSON 文件，例如 /tmp/ecs-cwagent-ec2.json。替换以下占位符：

- 将 `{{ecs-task-role}}` 替换为 ECS 任务角色的 ARN。

- 将 `{{ecs-task-execution-role}}` 替换为 ECS 任务执行角色的 ARN。
- 将 `{{demo-app-image}}` 替换为已启用 X-Ray 开发工具包集成的应用程序镜像。将名称从 demo-app 更改为您自己的应用程序名称。
- 将 `{{region}}` 替换为要发送容器日志的 Amazon 区域的名称。例如：us-west-2。

```
{
  "family": "ecs-cwagent-ec2",
  "taskRoleArn": "{{ecs-task-role}}",
  "executionRoleArn": "{{ecs-task-execution-role}}",
  "networkMode": "bridge",
  "containerDefinitions": [
    {
      "name": "demo-app",
      "image": "{{demo-app-image}}",
      "links": [
        "cloudwatch-agent",
        "xray-daemon"
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "True",
          "awslogs-group": "/ecs/ecs-cwagent-ec2",
          "awslogs-region": "{{region}}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    },
    {
      "name": "xray-daemon",
      "image": "public.ecr.aws/xray/aws-xray-daemon:latest",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "True",
          "awslogs-group": "/ecs/ecs-cwagent-ec2",
          "awslogs-region": "{{region}}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    },
    {
      "name": "cloudwatch-agent",
      "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
      "secrets": [
        {
          "name": "CW_CONFIG_CONTENT",
          "valueFrom": "ecs-cwagent"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "True",
          "awslogs-group": "/ecs/ecs-cwagent-ec2",
          "awslogs-region": "{{region}}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "requiresCompatibilities": [
    "EC2"
  ]
}
```

```
],  
  "cpu": "256",  
  "memory": "256"  
}
```

输入以下命令以创建任务定义。将 `{{region}}` 替换为集群的区域。

```
aws ecs register-task-definition \  
  --cli-input-json file:///tmp/ecs-cwagent-ec2.json \  
  --region {{region}}
```

输入以下命令以启动任务。将 `{{cluster-name}}` 和 `{{region}}` 分别替换为集群的名称和区域。

```
aws ecs run-task \  
  --cluster {{cluster-name}} \  
  --task-definition ecs-cwagent-ec2 \  
  --region {{region}} \  
  --launch-type EC2
```

Fargate 启动类型

首先，创建任务定义。在此示例中，容器“demo-app”向 CloudWatch 代理发送 X-Ray 开发工具包指标，并向 X-Ray 守护进程发送跟踪信息。

将以下任务定义复制到本地 JSON 文件，例如 `/tmp/ecs-cwagent-ec2.json`。替换以下占位符：

- 将 `{{ecs-task-role}}` 替换为 ECS 任务角色的 Amazon Resource Name (ARN)。
- 将 `{{ecs-task-execution-role}}` 替换为 ECS 任务执行角色的 ARN。
- 将 `{{demo-app-image}}` 替换为已启用 X-Ray 开发工具包集成的应用程序镜像。将名称从 `demo-app` 更改为您自己的应用程序名称。
- 将 `{{region}}` 替换为要发送容器日志的 Amazon 区域的名称。例如：`us-west-2`。

```
{  
  "family": "ecs-cwagent-fargate",  
  "taskRoleArn": "{{ecs-task-role}}",  
  "executionRoleArn": "{{ecs-task-execution-role}}",  
  "networkMode": "awsvpc",  
  "containerDefinitions": [  
    {  
      "name": "demo-app",  
      "image": "{{demo-app-image}}",  
      "logConfiguration": {  
        "logDriver": "awslogs",  
        "options": {  
          "awslogs-create-group": "True",  
          "awslogs-group": "/ecs/ecs-cwagent-fargate",  
          "awslogs-region": "{{region}}",  
          "awslogs-stream-prefix": "ecs"  
        }  
      }  
    },  
    {  
      "name": "xray-daemon",  
      "image": "public.ecr.aws/xray/aws-xray-daemon:latest",  
      "logConfiguration": {  
        "logDriver": "awslogs",  
        "options": {  
          "awslogs-create-group": "True",
```

```
        "awslogs-group": "/ecs/ecs-cwagent-fargate",
        "awslogs-region": "{{region}}",
        "awslogs-stream-prefix": "ecs"
      }
    },
    {
      "name": "cloudwatch-agent",
      "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
      "secrets": [
        {
          "name": "CW_CONFIG_CONTENT",
          "valueFrom": "ecs-cwagent"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "True",
          "awslogs-group": "/ecs/ecs-cwagent-fargate",
          "awslogs-region": "{{region}}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "cpu": "512",
  "memory": "1024"
}
```

输入以下命令以创建任务定义。将 `{{region}}` 替换为集群的区域。

```
aws ecs register-task-definition \
  --cli-input-json file:///tmp/ecs-cwagent-fargate.json \
  --region {{region}}
```

如果您已设置 Fargate 集群，则可使用刚刚创建的任务定义来启动任务。如果您尚不具有任何 Fargate 集群，请参阅[配置服务](#)以了解有关设置 Fargate 的其余步骤的更多信息。

在 Amazon EKS 或 Kubernetes 上部署 CloudWatch 代理和 X-Ray 守护进程

这些主题介绍如何在 Amazon EKS 或 Kubernetes 上安装 X-Ray 守护进程和 CloudWatch 代理。

在 Amazon EKS 或 Kubernetes 上部署 X-Ray 守护进程

要在 Amazon EKS 或 Kubernetes 上安装 CloudWatch 代理和 X-Ray 守护进程，您可以使用快速设置。

在 Amazon EKS 或 Kubernetes 上安装 CloudWatch 代理和 X-Ray 守护进程

1. 确保附加到 EC2 实例或 Kubernetes Worker 节点的 IAM 角色已附加了 `CloudWatchAgentServerPolicy` 和 `AWSXRayDaemonWriteAccess` 策略。
2. 输入以下命令：

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/master/k8s-deployment-manifest-templates/deployment-mode/daemonset/cwagent-fluentd-xray/cwagent-fluentd-xray-quickstart.yaml | sed "s/{{cluster_name}}/cluster-name/;s/{{region_name}}/region/" | kubectl apply -f -
```

快速入门的用途

本部分介绍 CloudWatch 代理和 X-Ray 守护进程的快速设置。

- 快速设置指定入站端口和协议。不必显式打开出站连接。
- 快速设置通过 `kubectl apply -f` 命令使用以下文件内容安装 X-Ray 守护进程。

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: xray-daemon
  namespace: amazon-cloudwatch
spec:
  selector:
    matchLabels:
      name: xray-daemon
  template:
    metadata:
      labels:
        name: xray-daemon
    spec:
      containers:
        - name: xray-daemon
          image: amazon/aws-xray-daemon:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 2000
              hostPort: 2000
              protocol: UDP
      resources:
        limits:
          cpu: 100m
          memory: 256Mi
        requests:
          cpu: 50m
          memory: 50Mi
      terminationGracePeriodSeconds: 60
```

- 快速设置使用 Docker 镜像版本/标签 1.231221.0 或更高版本或最新版本更新 CloudWatch 代理。您可以在 <https://hub.docker.com/r/amazon/cloudwatch-agent> 上查找映像。
- 为了使 X-Ray 开发工具包能够读取集群名称和区域信息，快速设置将使用 Docker 镜像版本/标签 1.231221.0 或更高版本或最新版本更新 CloudWatch 代理。您可以在 <https://hub.docker.com/r/amazon/cloudwatch-agent> 上查找映像。
- 为了使 X-Ray 开发工具包能够读取集群名称和区域信息，快速设置创建了一个包含以下内容的文件，然后使用 `kubectl apply -f` 命令应用该文件。

```
---
# create role binding for XRay SDK to read config map
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: container-insights-discovery-role
  namespace: amazon-cloudwatch
rules:
- apiGroups:
  - ""
  resourceNames:
  - cluster-info
  resources:
  - configmaps
  verbs:
  - get
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: service-users-cloudwatch-discovery-role-binding
  namespace: amazon-cloudwatch
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: container-insights-discovery-role
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
```

- 快速设置会显示接收 X-Ray 开发工具包指标的 CloudWatch 代理端口。默认端口为 UDP 25888。

```
ports:
- containerPort: 25888
  hostPort: 25888
  protocol: UDP
```

- 快速设置将代理配置 JSON 与带有以下 JSON 的 X-Ray 开发工具包指标配置合并。

```
{
  "logs": {
    "metrics_collected": {
      "emf": {}
    }
  }
}
```

在 Amazon EC2 上部署 CloudWatch 代理和 X-Ray 守护进程

CloudWatch 代理和 X-Ray 守护进程的标准安装足以在 Amazon EC2 上启用 ServiceLens，并添加了以下 CloudWatch 代理配置部分示例。有关安装代理的更多信息，请参阅 [安装 CloudWatch 代理 \(p. 660\)](#)。有关 CloudWatch 代理配置文件的更多信息，请参阅 [手动创建或编辑 CloudWatch 代理配置文件 \(p. 686\)](#)。

在配置 CloudWatch 代理时，请在配置文件中包含此部分：

```
{
  "logs": {
    "metrics_collected": {
      "emf": {}
    }
  }
}
```

有关安装 X-Ray 守护进程的更多信息，请参阅 [X-Ray 守护进程配置](#)。

在 ServiceLens 中使用服务地图

此部分介绍了服务地图，并帮助您了解如何导航服务地图。

要查看服务地图，您必须已安装 Amazon X-Ray 并已完成其他 ServiceLens 部署步骤。有关更多信息，请参阅 [部署 ServiceLens \(p. 196\)](#)。

您还必须登录到具有 `AWSXrayReadOnlyAccess` 托管策略以及允许您查看 CloudWatch 控制台的权限的账户。有关更多信息，请参阅 [Amazon X-Ray 如何与 IAM 结合使用](#) 和 [使用 Amazon CloudWatch 控制面板 \(p. 29\)](#)。

开始使用服务地图

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 ServiceLens，然后选择 Service Map (服务地图)。

这将显示服务地图。它具有以下部分：

- 已为其启用跟踪的 Amazon 服务和自定义应用程序将显示为圆圈或“节点”。每个节点的大小指示要发送到该服务的已跟踪请求的相对数量。
- 边缘或节点之间的连接将显示为连接节点的线。默认情况下，线的粗细指示这些节点之间的已跟踪请求的相对数量。

您可以使用右上角的下拉菜单来选择是将已跟踪请求数还是平均延迟用于节点和边缘大小调整。您还可以选择为所有节点和边缘使用固定大小。

- 节点的入口点将在左侧显示为“客户端”。“客户端”表示 Web 服务器流量和已跟踪的 API 操作请求。
 - 带有部分红色、橙色或紫色轮廓的节点存在问题。对这些节点的某些已跟踪请求存在故障、错误或限制。颜色轮廓的百分比指示存在问题的已跟踪请求的百分比。
 - 如果节点旁边有一个带感叹号的三角形，则至少一个与该节点相关的 CloudWatch 告警处于告警状态。
3. 默认情况下，地图中的数据针对的是最近 6 小时的时段。要更改时段的时间范围，请使用屏幕右上角的控件。要显示的时间范围可以多达 6 小时，也可以是过去的 30 天。
 4. 如果已启用 X-Ray 组，则可通过在筛选器中选择一个 X-ray 组来筛选地图。
 5. 要查看节点的指标，请选择该节点。要查看有关该节点的详细信息，请选择 View logs (查看日志)、View traces (查看跟踪)，或 View dashboard (查看控制面板)。
 6. 要关注节点的传入和传出连接，请选择节点，然后选择服务地图顶部附近的 View connections (查看连接)。
 7. 要查看显示节点的延迟、错误、请求和警报摘要统计数据的弹出窗口，请在该节点上暂停。
 8. 要查看边缘连接的延迟统计数据，请在表示该边缘的线上暂停。
 9. 要显示服务的警报状态以及延迟、错误和跟踪计数的折线图，请在地图上选择该服务节点。

有关此视图的更多信息，请参阅以下过程。

10. 要以表形式查看服务地图，请选择屏幕顶部附近的 List view (列表视图)。在此视图中，您可以对地图上显示的节点和警报进行筛选和排序。
11. 要查看具有特定节点的指标的控制面板，请选择该节点，然后选择屏幕底部附近的 View dashboard (查看控制面板)。

在服务地图上查看服务或应用程序的跟踪

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 ServiceLens，然后选择 Service Map (服务地图)。
3. 选择表示要调查的服务或应用程序的节点。

CloudWatch 显示该服务的延迟、错误和跟踪计数的折线图以及告警状态的摘要。

这些图表上方是用于深入了解该服务的日志和跟踪的选项。

4. 要查看与服务相关的跟踪，请选择 View traces (查看跟踪)。

控制台将切换到 Traces (跟踪) 视图，并侧重于您正在调查的服务。有关更多信息，请参阅[使用 ServiceLens 中的跟踪视图 \(p. 205\)](#)。

使用 ServiceLens 中的跟踪视图

利用跟踪视图，您可以查看应用程序中最近的 X-Ray 跟踪。

查看跟踪并深入了解更多信息

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 ServiceLens，然后选择 Traces（跟踪）。
3. 在 Filter type（筛选条件类型）下，您可以选择用于对跟踪进行排序的其他条件。在您执行此操作时，CloudWatch 将根据您的选择显示跟踪的成功率和响应时间。页面底部显示的跟踪列表的范围将缩小到与筛选条件匹配的跟踪。将检索与筛选条件匹配的前 1000 个跟踪。

筛选条件表中的值由筛选条件返回的跟踪填充，并在筛选条件中优化选择时进行更新。

使用 Filter type（筛选条件类型）下的 Custom query（自定义查询）选项，将自定义表达式添加为筛选条件。自定义表达式可以包含关键词、一元或二元运算符以及关键词的比较值。关键词可对应于跟踪的各个部分，例如 `responsetime`、`duration` 和状态代码。有关更多信息，请参阅[筛选表达式语法](#)。

可以通过选中 Traces by（跟踪依据）下的行旁边的复选框并选择 Add to filter（添加到筛选条件）来进一步专注于筛选条件。选定行中的筛选值将添加到筛选条件中。

4. 要查看表示由当前筛选条件返回的跟踪的响应时间分布的直方图，以及显示了每个跟踪的表，请滚动到屏幕底部。

选择表中的某个跟踪可查看有关该跟踪的详细信息。

跟踪详细信息页面显示一个时间线，其中包括构成跟踪的每个段。选择一个段可查看其他关联的元数据，包括错误（如果适用）。对于可关联日志的跟踪，与跟踪关联的日志行将显示在时间线下方。

然后，您可以选择在 CloudWatch Logs Insights 中查看这些日志条目。为此，请选择 View in CloudWatch Logs Insights（在 CloudWatch Logs Insights 中查看），然后选择 Run query（运行查询）。

某些跟踪没有关联的日志条目。

- 如果您未看到 Amazon EC2 或 Amazon EKS 的日志条目，则需要更新到最新版本的 X-Ray 开发工具包。有关更多信息，请参阅[部署 Amazon X-Ray \(p. 196\)](#)。
- 只有在名称以 `/ecs/` 开头的日志组数量为 20 个或以下时，ServiceLens 才会显示 Amazon ECS 的日志条目。
- 目前，Fargate 跟踪的日志条目暂不可用。

使用 ServiceLens 中的资源运行状况视图

您可以使用资源运行状况视图在单个视图中自动发现、管理和可视化主机在其所有应用程序中的运行状况和性能。您可以按性能维度（如 CPU 或内存）可视化显示其主机的运行状况，并使用筛选条件在单个视图对数百台主机进行切片和切块。您可以按标签或使用案例进行筛选，例如同一个 Auto Scaling 组中的主机或使用同一负载均衡器的主机，

先决条件

要确保充分享受资源运行状况视图的益处，请检查您是否满足以下先决条件。

- 要查看主机的内存利用率并将其用作筛选条件，您必须在主机上安装 CloudWatch 代理，并将其设置为在默认 `CWAgent` 命名空间中向 CloudWatch 发送内存指标。在 Linux 和 macOS 实例上，CloudWatch 代理必须发送 `mem_used_percent` 指标。在 Windows 实例上，此代理必须发送 `Memory % Committed Bytes in Use` 指标。如果您使用向导创建 CloudWatch 代理配置文件并选择任意预定义的指标集，则这些指标会包含在其中。CloudWatch 代理收集的指标按自定义指标计费。有关更多信息，请参阅[安装 CloudWatch 代理 \(p. 660\)](#)。

当您使用 CloudWatch 代理收集这些内存指标以用于资源运行状况视图时，您必须将以下部分添加到 CloudWatch 代理配置文件中。此部分包含默认维度设置，并且是预设情况下的创建内容，因此请勿更改此部分的任何内容，保持其与以下示例所示内容一致。

```
"append_dimensions": {
  "ImageId": "${aws:ImageId}",
  "InstanceId": "${aws:InstanceId}",
  "InstanceType": "${aws:InstanceType}",
  "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
},
```

- 要查看资源运行状况视图中的所有可用信息，您必须登录具有以下权限的账户。如果您登录的账户的权限较少，您仍然可以使用资源运行状况视图，但某些性能数据将不可见。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:Describe*",
        "cloudwatch:Describe*",
        "cloudwatch:Describe*",
        "cloudwatch:Get*",
        "cloudwatch:List*",
        "logs:Get*",
        "logs:Describe*",
        "sns:Get*",
        "sns:List*",
        "ec2:DescribeInstances",
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeRegions"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

在账户中查看资源运行状况

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 ServiceLens，然后选择 Resource Health（资源运行状况）。

此时将显示资源运行状况页面，页面上显示账户中每台主机对应的一个方块。根据 Color by（颜色依据）设置，每个方块的颜色都基于该主机的当前状态。主机方块带有告警符号，表示该主机当前有一个或多个处于 ALARM 状态的告警。

您可以在单个视图中查看多达 500 台主机的运行状况。如果您账户中主机数量超过此数量，请使用此程序步骤 6 中的筛选条件设置。

3. 若要更改显示每个主机运行状况所基于的条件，请选择 Color by（颜色依据）。您可以选择 CPU Utilization（CPU 利用率）、Memory Utilization（内存利用率），或 Status check（状态检查）。内存利用率指标仅适用于运行 CloudWatch 代理且将其配置为收集内存指标并将指标发送到默认 CWAgent 命名空间的主机。有关更多信息，请参阅[使用 CloudWatch 代理从 Amazon EC2 实例和本地部署服务器中收集指标和日志](#) (p. 659)。
4. 若要更改用于网格中运行状况指标的阈值和颜色，请选择网格上方的齿轮图标。
5. 若要切换是否在主机网格中显示告警，请选择或清除 Show alarms across all metrics（显示所有指标的告警）。

6. 若要将地图中的主机拆分为组，请选择 Group by (分组依据)。
7. 若要将视图所涵盖的主机范围缩小，请选择 Filter by (筛选条件) 中的筛选条件。您可以按标签和资源分组 (如 Auto Scaling 组、实例类型、安全组等) 进行筛选。
8. 若要对主机进行排序，请选择 Sort by (排序方式)。您可以按状态检查结果、实例状态、CPU 或内存利用率以及处于 ALARM 状态的告警数量进行排序。
9. 要查看有关主机的详细信息，请选择表示该主机的方块。随后会显示一个弹出窗格。然后要深入了解有关该主机的信息，请选择 View dashboard (查看控制面板) 或 View on list (在列表中查看)。

ServiceLens 问题排查

如果您在使用 CloudWatch ServiceLens 时遇到问题，以下部分可为您提供帮助。

我未看到我的所有日志

如何将日志配置为根据服务显示在 ServiceLens 中。

- 如果已在 API Gateway 中启用了日志记录，则会显示 API Gateway 日志。
- 如果您使用的是最新版本 of X-Ray 开发工具包和 CloudWatch 代理，则会显示 Amazon ECS 和 Amazon EKS 日志。有关更多信息，请参阅[部署 ServiceLens \(p. 196\)](#)。
- 如果请求 ID 位于日志条目中，则会显示 Lambda 日志。对于下表中列出的情况，此操作会自动发生。对于其他情况，如果运行时未自动包含跟踪 ID，您可以手动包含跟踪 ID。

运行时	方法	请求 ID 是否会自动显示在日志条目中？
Java	context.getLogger.log aws-lambda-java-log4j2	是
Java	System.out.println	否
Python	context.log logging.info/error/log/etc...	是
Python	print	否
Node.js	context.log console.log/info/error/etc...	是
dotnet	context.Logger.log Console.WriteLine()	否
Go	fmt.Printf log.Print	否
Ruby	puts	否

我未在服务地图上看到我的所有告警

如果与节点关联的任何告警都处于 ALARM 状态，则 ServiceLens 仅显示该节点的提示图标。

ServiceLens 使用以下逻辑将告警与节点关联：

- 如果节点代表 Amazon 服务，则包含与该服务关联的命名空间的所有告警都与该节点关联。例如，AWS::Kinesis 类型的节点与基于 CloudWatch 命名空间 AWS/Kinesis 中指标的所有告警关联。
- 如果节点代表一个 Amazon 资源，则将链接该特定资源上的告警。例如，名为“MyTable”的 AWS::DynamoDB::Table 类型的节点将链接到所有基于具有命名空间 AWS/DynamoDB 的指标且 TableName 维度设置为 MyTable 的告警。
- 如果节点的类型未知（由名称周围的虚线边框标识），则任何告警均不会与该节点关联。

我未看到服务地图上的某些 Amazon 资源

对于服务地图上要跟踪的 Amazon 资源，必须使用 X-Ray 开发工具包捕获 Amazon 开发工具包。有关 X-Ray 的更多信息，请参阅[什么是 Amazon X-Ray](#)。

并非每个 Amazon 资源都由一个专用节点表示。对于对服务的所有请求，某些 Amazon 服务由单个节点表示。将显示以下资源类型，并且每个资源对应一个节点：

- AWS::DynamoDB::Table
- AWS::Lambda::Function

Lambda 函数由两个节点表示：一个表示 Lambda 容器，另一个表示函数。这有助于识别 Lambda 函数的冷启动问题。Lambda 容器节点与告警和控制面板的关联方式与 Lambda 函数节点与告警和控制面板的关联方式相同。

- AWS::ApiGateway::Stage
- AWS::SQS::Queue
- AWS::SNS::Topic

我的服务地图包含太多节点

使用 X-Ray 组将您的地图分成多个地图。有关更多信息，请参阅[对组使用筛选表达式](#)。

使用 Synthetics 监控

您可以使用 Amazon CloudWatch Synthetics 创建金丝雀。金丝雀是按计划运行的可配置脚本，用于监控端点和 API。金丝雀遵循相同的路线并执行与客户相同的操作，这使您能够持续验证您的客户体验，即使您的应用程序中没有任何客户流量也可以。使用金丝雀，您可以早于客户先行发现问题。

金丝雀是用 Node.js 或 Python 编写的脚本。它们以 Node.js 或 Python 为框架在您的账户中创建 Lambda 函数。金丝雀通过 HTTP 和 HTTPS 两种协议工作。

金丝雀通过 Puppeteer 或 Selenium Webdriver 提供对无头 Google Chrome 浏览器的编程访问。有关 Puppeteer 的更多信息，请参阅[Puppeteer](#)。有关 Selenium 的更多信息，请参阅www.selenium.dev/。

金丝雀检查终端节点的可用性和延迟，并可以存储加载时间数据和 UI 屏幕截图。它们可以监控您的 REST API、URL 和网站内容，并且可以检查来自网络钓鱼、代码注入和跨站脚本的未经授权更改。

您可以运行一次金丝雀，也可以定期运行。金丝雀的运行频率可达每分钟一次。您可以使用 cron 和 rate 表达式来计划金丝雀。

有关在创建和运行金丝雀之前需要考虑的安全问题相关信息，请参阅[Synthetics 金丝雀的安全注意事项 \(p. 894\)](#)。

预设情况下，金丝雀会在 CloudWatchSynthetics 命名空间中创建几个 CloudWatch 指标。这些指标使用 CanaryName 作为维度。使用函数库中 executeStep() 或 executeHttpStep() 函数的金丝雀还具有 StepName 维度。有关金丝雀函数库的更多信息，请参阅[可用于金丝雀脚本的库函数](#) (p. 244)。

CloudWatch Synthetics 与 CloudWatch ServiceLens 很好地集成，其将 CloudWatch 与 Amazon X-Ray 结合使用，以提供服务的端到端视图，帮助您更有效地查明性能瓶颈并确定受影响的用户。使用 CloudWatch Synthetics 创建的金丝雀将显示在 ServiceLens 服务映射中。有关 ServiceLens 的更多信息，请参阅[使用 ServiceLens 监控应用程序的运行状况](#) (p. 194)。

CloudWatch Synthetics 目前在所有商业 Amazon 区域和 GovCloud 区域可用。

Note

在亚太地区（大阪）不支持 Amazon PrivateLink。在亚太地区（雅加达），Amazon PrivateLink 和 X-Ray 不受支持。

主题

- [CloudWatch 金丝雀的必需角色和权限](#) (p. 209)
- [创建金丝雀](#) (p. 219)
- [组](#) (p. 274)
- [排查失败金丝雀的问题](#) (p. 275)
- [金丝雀脚本示例代码](#) (p. 278)
- [金丝雀和 X-Ray 跟踪](#) (p. 282)
- [在 VPC 上运行金丝雀](#) (p. 283)
- [加密金丝雀构件](#) (p. 283)
- [查看金丝雀统计数据和详细信息](#) (p. 285)
- [金丝雀发布的 CloudWatch 指标](#) (p. 286)
- [编辑或删除金丝雀](#) (p. 287)
- [使用 Amazon EventBridge 监控金丝雀事件](#) (p. 289)

CloudWatch 金丝雀的必需角色和权限

创建和管理金丝雀的用户以及金丝雀本身都必须拥有某些权限。

管理 CloudWatch 金丝雀的用户的必需角色和权限

若要查看金丝雀详细信息和金丝雀运行的结果，您必须以附加了 CloudWatchSyntheticsFullAccess 或 CloudWatchSyntheticsReadOnlyAccess 的 IAM 用户身份登录。要在控制台中读取所有 Synthetics 数据，您还需要 AmazonS3ReadOnlyAccess 和 CloudWatchReadOnlyAccess 策略。要查看金丝雀使用的源代码，您还需要 AWSLambda_ReadOnlyAccess 策略。

若要创建金丝雀，您必须以具有 CloudWatchSyntheticsFullAccess 策略或类似权限集的 IAM 用户身份登录。要为金丝雀创建 IAM 角色，您还需要以下内联策略语句：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:CreatePolicy",
        "iam:AttachRolePolicy"
      ],
    }
  ],
}
```

```
        "Resource": [
            "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*",
            "arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*"
        ]
    }
}
```

Important

通过向用户授予 `iam:CreateRole`、`iam:CreatePolicy` 和 `iam:AttachRolePolicy` 权限，用户将获得对 Amazon 账户的完全管理访问权限。例如，具有这些权限的用户可以创建一个对所有资源具有完全权限的策略，并将该策略附加到任何角色。请谨慎地为相关人员授予这些权限。

有关附加策略和向用户授予权限的信息，请参阅[更改 IAM 用户的权限](#)和[为用户或角色嵌入内联策略](#)。

金丝雀的必需角色和权限

每个金丝雀必须与附加某些权限的 IAM 角色相关联。当您使用 CloudWatch 控制台创建金丝雀时，您可以选择 CloudWatch Synthetics 为金丝雀创建 IAM 角色。如果您这样做，该角色将拥有所需的权限。

如果您要自己创建 IAM 角色，或者创建使用 Amazon CLI 或 API 创建金丝雀时可以使用的 IAM 角色，角色必须包含本部分中列出的权限。

金丝雀的所有 IAM 角色必须包含以下信任策略语句。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

此外，金丝雀的 IAM 角色需要以下语句之一。

不使用 Amazon KMS 或者需要 Amazon VPC 访问的基本金丝雀

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::path/to/your/s3/bucket/canary/results/folder"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],

```

```

        "Resource": [
            "arn:aws:s3:::name/of/the/s3/bucket/that/contains/canary/results"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:CreateLogGroup"
        ],
        "Resource": [
            "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/lambda/
cwsyn-canary_name-*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:ListAllMyBuckets",
            "xray:PutTraceSegments"
        ],
        "Resource": [
            "*"
        ]
    },
    {
        "Effect": "Allow",
        "Resource": "*",
        "Action": "cloudwatch:PutMetricData",
        "Condition": {
            "StringEquals": {
                "cloudwatch:namespace": "CloudWatchSynthetics"
            }
        }
    }
]
}

```

使用 Amazon KMS 加密金丝雀构件但不需要 Amazon VPC 访问的金丝雀

```

{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject"
        ],
        "Resource": [
            "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation"
        ],
        "Resource": [
            "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [

```



```

        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:CreateLogGroup"
    ],
    "Resource": [
        "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/lambda/
cwsyn-canary_name-*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "s3:ListAllMyBuckets",
        "xray:PutTraceSegments"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Allow",
    "Resource": "*",
    "Action": "cloudwatch:PutMetricData",
    "Condition": {
        "StringEquals": {
            "cloudwatch:namespace": "CloudWatchSynthetics"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource":
    "arn:aws:kms:KMS_key_region_name:KMS_key_account_id:key/KMS_key_id",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": [
                "S3.region_name_of_the_canary_results_S3_bucket.amazonaws.com"
            ]
        }
    }
}
]
}

```

不使用 Amazon KMS 但需要 Amazon VPC 访问的金丝雀

```

{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject"
        ],
        "Resource": [
            "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [

```

```

        "s3:GetBucketLocation"
    ],
    "Resource": [
        "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:CreateLogGroup"
    ],
    "Resource": [
        "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/lambda/
cwsyn-canary_name-*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "s3:ListAllMyBuckets",
        "xray:PutTraceSegments"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Allow",
    "Resource": "*",
    "Action": "cloudwatch:PutMetricData",
    "Condition": {
        "StringEquals": {
            "cloudwatch:namespace": "CloudWatchSynthetics"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2>DeleteNetworkInterface"
    ],
    "Resource": [
        "*"
    ]
}
]
}

```

使用 Amazon KMS 加密金丝雀构件并且也需要 Amazon VPC 访问的金丝雀

如果您更新非 VPC 金丝雀以开始使用 VPC，则为了包含以下策略中列出的网络接口权限，需要更新金丝雀的角色。

```

{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject"
        ]
    }]
}

```

```

    ],
    "Resource": [
      "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetBucketLocation"
    ],
    "Resource": [
      "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogStream",
      "logs:PutLogEvents",
      "logs:CreateLogGroup"
    ],
    "Resource": [
      "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/lambda/
cwsyn-canary_name-*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:ListAllMyBuckets",
      "xray:PutTraceSegments"
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Effect": "Allow",
    "Resource": "*",
    "Action": "cloudwatch:PutMetricData",
    "Condition": {
      "StringEquals": {
        "cloudwatch:namespace": "CloudWatchSynthetics"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DeleteNetworkInterface"
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey"
    ],
    "Resource":
      "arn:aws:kms:KMS_key_region_name:KMS_key_account_id:key/KMS_key_id",
    "Condition": {

```

```
        "StringEquals": {
            "kms:ViaService": [
                "S3.region_name_of_the_canary_results_S3_bucket.amazonaws.com"
            ]
        }
    }
}
]
```

用于 CloudWatch Synthetics 的 Amazon 托管式策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 Amazon 托管策略更简单。创建仅为团队提供所需权限的 IAM 客户托管式策略需要时间和专业知识。要快速入门，您可以使用我们的 Amazon 托管策略。这些策略涵盖常见使用案例，可在您的 Amazon 账户中使用。有关 Amazon 托管式策略的更多信息，请参阅 IAM 用户指南中的 [Amazon 托管式策略](#) Amazon 托管式策略。

Amazon 服务负责维护和更新 Amazon 托管策略。您无法更改 Amazon 托管策略中的权限。服务会偶尔更改 Amazon 托管式策略中的权限。此类更新会影响附加策略的所有身份（用户、组和角色）。

CloudWatch Synthetics 对 Amazon 托管式策略的更新

查看有关 CloudWatch Synthetics 的 Amazon 托管式策略更新的详细信息（从该服务开始跟踪这些更改开始）。有关此页面更改的自动提示，请订阅 CloudWatch 文档历史记录页面上的 RSS 源。

更改	说明	日期
删除了 CloudWatchSyntheticsFullAccess 中的冗余操作	CloudWatch Synthetics 删除了 CloudWatchSyntheticsFullAccess 策略中的 s3:PutBucketEncryption 和 lambda:GetLayerVersionByArn 操作，因为这些操作相较于策略中的其他权限是多余的。删除的操作不提供任何权限，策略授予的权限也未发生任何净更改。	2021 年 3 月 12 日
CloudWatch Synthetics 开始跟踪变更	CloudWatch Synthetics 开始跟踪其 Amazon 托管式策略的更改。	2021 年 3 月 10 日

CloudWatchSyntheticsFullAccess

以下是 CloudWatchSyntheticsFullAccess 策略的内容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "synthetics:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
```

```

        "s3:PutEncryptionConfiguration"
    ],
    "Resource": [
        "arn:aws:s3:::cw-syn-results-*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "iam:ListRoles",
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation",
        "xray:GetTraceSummaries",
        "xray:BatchGetTraces",
        "apigateway:GET"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "s3:GetObject",
        "s3:ListBucket"
    ],
    "Resource": "arn:aws:s3:::cw-syn-*"
},
{
    "Effect": "Allow",
    "Action": [
        "s3:GetObjectVersion"
    ],
    "Resource": "arn:aws:s3:::aws-synthetics-library-*"
},
{
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
    ],
    "Resource": [
        "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
    ],
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": [
                "lambda.amazonaws.com",
                "synthetics.amazonaws.com"
            ]
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "iam:GetRole"
    ],
    "Resource": [
        "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "cloudwatch:GetMetricData",
        "cloudwatch:GetMetricStatistics"
    ],
    "Resource": "*"
}

```

```

    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:DeleteAlarms"
      ],
      "Resource": [
        "arn:aws:cloudwatch:*:*:alarm:Synthetics-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarms"
      ],
      "Resource": [
        "arn:aws:cloudwatch:*:*:alarm:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "lambda:CreateFunction",
        "lambda:AddPermission",
        "lambda:PublishVersion",
        "lambda:UpdateFunctionConfiguration",
        "lambda:GetFunctionConfiguration"
      ],
      "Resource": [
        "arn:aws:lambda:*:*:function:cwsyn-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "lambda:GetLayerVersion",
        "lambda:PublishLayerVersion"
      ],
      "Resource": [
        "arn:aws:lambda:*:*:layer:cwsyn-*",
        "arn:aws:lambda:*:*:layer:Synthetics:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:ListTopics"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",

```

```
        "Action": [
            "sns:CreateTopic",
            "sns:Subscribe",
            "sns:ListSubscriptionsByTopic"
        ],
        "Resource": [
            "arn:*:sns:*:*:Synthetics-*"
        ]
    }
]
```

CloudWatchSyntheticsReadOnlyAccess

以下是 CloudWatchSyntheticsReadOnlyAccess 策略的内容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "synthetics:Describe*",
        "synthetics:Get*",
        "synthetics:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

限制用户查看特定的金丝雀

您可以限制用户查看金丝雀相关信息，使他们只能看到您指定金丝雀的信息。为此，请将 IAM policy 与类似以下内容的 Condition 声明配合使用，并将此策略附加到 IAM 用户或 IAM 角色。

以下示例限制了用户只能查看 name-of-allowed-canary-1 和 name-of-allowed-canary-2 相关信息。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "synthetics:DescribeCanaries",
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "synthetics:Names": [
            "name-of-allowed-canary-1",
            "name-of-allowed-canary-2"
          ]
        }
      }
    }
  ]
}
```

CloudWatch Synthetics 支持在 synthetics:Names 数组中最多列出五个项目。

您也可以创建策略，在允许的金丝雀名称中使用 * 作为通配符，如以下示例所示：


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "synthetics:DescribeCanaries",
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "synthetics:Names": [
            "my-team-canary-*"
          ]
        }
      }
    }
  ]
}
```

使用所附的这些策略之一登录的任何用户都不能使用 CloudWatch 控制台查看任何金丝雀信息。他们只能使用 [DescribeCanaries](#) API 或 [describe-canaries](#) Amazon CLI 命令，查看策略授权的金丝雀信息。

创建金丝雀

Important

确保使用 Synthetics 金丝雀来仅监控您拥有所有权或权限的终端节点和 API。根据金丝雀运行频率设置，这些端点可能会面临流量增加的情况。

当您使用 CloudWatch 控制台创建金丝雀时，您可以使用 CloudWatch 提供的蓝图创建您的金丝雀，也可以编写您自己的脚本。有关更多信息，请参阅[使用金丝雀蓝图](#) (p. 221)。

如果使用的是您自己的金丝雀脚本，您还可以使用 Amazon CloudFormation 创建金丝雀。有关更多信息，请参阅 Amazon CloudFormation 用户指南中的 [AWS::Synthetics::Canary](#)。

如果您正在编写自己的脚本，则可以使用 CloudWatch Synthetics 已内置在库中的多个函数。有关更多信息，请参阅[Synthetics 运行时版本](#) (p. 226)。

创建金丝雀

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Application monitoring (应用程序监控)、Synthetics Canaries (Synthetics 金丝雀)。
3. 选择 Create Canary (创建金丝雀)。
4. 选择以下选项之一：
 - 要将某个蓝图脚本作为金丝雀的基础，请选择 Use a blueprint (使用蓝图)，然后选择要创建的金丝雀的类型。有关每种类型的蓝图执行的操作的更多信息，请参阅[使用金丝雀蓝图](#) (p. 221)。
 - 要上传您自己的 node.js 脚本以创建自定义金丝雀，请选择 Upload a script (上传脚本)。

然后，您可以将脚本拖入 Script (脚本) 区域，或者选择 Browse files (浏览文件)，在文件系统中浏览找到该脚本。

- 要从 S3 存储桶导入脚本，请选择 Import from S3 (从 S3 导入)。然后，在 Source location (源位置) 下，输入金丝雀的完整路径或选择 Browse S3 (浏览 S3)。

您必须对您使用的 S3 存储桶具有 `s3:GetObject` 和 `s3:GetObjectVersion` 权限。存储桶必须位于您在其中创建金丝雀的同一 Amazon 区域内。

5. 在 Name (名称) 下, 输入金丝雀的名称。此名称将用于许多页面, 因此, 我们建议您为该金丝雀提供一个描述性名称, 以将它与其他金丝雀区分开来。
6. 在 Application or endpoint URL (应用程序或终端节点 URL) 下, 输入您希望金丝雀测试的 URL。此 URL 必须包含协议 (例如 `https://`)。

如果您希望金丝雀测试 VPC 上的终端节点, 还必须在此过程随后的步骤中输入有关 VPC 的信息。

7. 如果您将自己的脚本用于金丝雀, 请在 Lambda handler (Lambda 处理程序) 下, 输入您希望金丝雀开始的入口点。如果您使用 `syn-nodejs-puppeteer-3.4` 或 `syn-python-selenium-1.1` 之前版本的运行时, 则您输入的字符串必须以 `.handler` 结尾。如果您使用 `syn-nodejs-puppeteer-3.4` 或 `syn-python-selenium-1.1` 或之后版本的运行时, 则此限制不适用。
8. 如果在脚本中使用环境变量, 请选择 Environment variables (环境变量), 然后为脚本中定义的每个环境变量指定值。有关更多信息, 请参阅[环境变量 \(p. 240\)](#)。
9. 在 Schedule (计划) 下, 选择是运行此金丝雀一次、使用 rate 表达式连续运行, 还是使用 cron 表达式对其进行计划。
 - 当您使用 CloudWatch 控制台创建连续运行的金丝雀时, 您可以在每分钟一次到每小时一次之间的范围内选择一个运行速率。
 - 有关为金丝雀调度编写 cron 表达式的更多信息, 请参阅[使用 cron 安排金丝雀运行 \(p. 273\)](#)。
10. (可选) 要为金丝雀设置超时值, 请选择 Additional configuration (其它配置), 然后指定该超时值。此值不要短于 15 秒钟, 以预留 Lambda 冷启动以及启动金丝雀工具的时间。
11. 在 Data retention (数据保留) 下, 指定保留有关失败和成功的金丝雀运行的信息的时间。范围为 1 - 455 天。

此设置仅影响 CloudWatch Synthetics 存储和显示在控制台中的数据。它不会影响 Amazon S3 存储桶中存储的数据, 也不会影响 Canary 发布的日志或指标。

12. 在 Data Storage (数据存储) 下, 选择要用于存储来自金丝雀运行的数据的 S3 存储桶。存储桶名称不能包含句点 (.)。如果将此选项保留空白, 将使用或创建默认 S3 存储桶。

如果您使用的是 `syn-nodejs-puppeteer-3.0` 或更高版本的运行时, 在文本框中输入存储桶的 URL 时, 您可以指定当前区域或其他区域中的存储桶。如果您使用的是较早版本的运行时, 则存储桶必须位于当前区域中。

13. (可选) 默认情况下, 金丝雀将自己的构件存储在 Amazon S3 上, 并使用 Amazon 托管式 Amazon KMS 密钥将构件静态加密。您可以通过选择 Data Storage (数据存储) 部分的 Additional configuration (其他配置) 来使用不同的加密选项。然后, 您可以选择用于加密的密钥类型。有关更多信息, 请参阅[加密金丝雀构件 \(p. 283\)](#)。
14. 在 Access permissions (访问权限) 下, 选择是创建 IAM 角色来运行金丝雀, 还是使用现有角色。

如果您使用 CloudWatch Synthetics 创建角色, 则其会自动包含所有必要的权限。如果您要自己创建角色, 请参阅[金丝雀的必需角色和权限 \(p. 210\)](#) 获取有关必要权限的信息。

如果您在创建金丝雀时使用 CloudWatch 控制台为金丝雀创建角色, 则无法将该角色重新用于其他金丝雀, 因为这些角色仅特定于一个金丝雀。如果您已手动创建了适用于多个金丝雀的角色, 则可以使用现有角色。

若要使用现有角色, 您必须具有 `iam:PassRole` 权限, 以将角色传递到 Synthetics 和 Lambda。您还必须拥有 `iam:GetRole` 权限。

15. (可选) 在 Alarms (告警) 下, 选择是否要为此金丝雀创建默认 CloudWatch 告警。如果选择创建告警, 则会按照以下命名约定创建告警: `Synthetics-Alarm-canaryName-index`

`index` 是一个数字, 代表为此金丝雀创建的各个不同的告警。第一个告警的索引为 1, 第二个告警的索引为 2, 依此类推。

16. (可选) 要让此金丝雀测试 VPC 上的终端节点, 请选择 VPC settings (VPC 设置), 并执行以下操作:
 - a. 选择托管终端节点的 VPC。

- b. 选择 VPC 上的一个或多个子网。您必须选择私有子网，因为在执行过程中无法为 Lambda 实例分配 IP 地址时，无法将 Lambda 实例配置为在公有子网中运行。有关更多信息，请参阅[配置 Lambda 函数以访问 VPC 中的资源](#)。
- c. 在 VPC 上选择一个或多个安全组。

如果端点位于 VPC 上，则必须启用您的金丝雀才能向 CloudWatch 和 Amazon S3 发送信息。有关更多信息，请参阅[在 VPC 上运行金丝雀 \(p. 283\)](#)。

17. (可选) 在 Tags (标签) 下，添加一个或多个键/值对作为此金丝雀的标签。标签可帮助您识别和组织 Amazon 资源并跟踪 Amazon 成本。有关更多信息，请参阅[标记 Amazon CloudWatch 资源 \(p. 903\)](#)。
18. (可选) 在 Active tracing (活动跟踪) 下，选择是否启用此金丝雀的活动 X-Ray 跟踪。仅当金丝雀使用 syn-nodejs-2.0 或更高的运行时版本时，此选项才可用。有关更多信息，请参阅[金丝雀和 X-Ray 跟踪 \(p. 282\)](#)。

为金丝雀创建的资源

创建金丝雀时，将创建以下资源：

- 一个名为 CloudWatchSyntheticsRole-*canary-name-uuid* 的 IAM 角色 (如果您使用 CloudWatch 控制台创建金丝雀并指定为金丝雀创建新角色)
- 一个名为 CloudWatchSyntheticsPolicy-*canary-name-uuid* 的 IAM policy。
- 一个名为 cw-syn-results-*accountID-region* 的 S3 存储桶。
- 名为 Synthetics-Alarm-*MyCanaryName* 的警报 (如果您要为金丝雀创建警报)。
- Lambda 函数和层 (如果您使用蓝图创建金丝雀)。这些资源具有前缀 cwsyn-*MyCanaryName*。
- 名为 /aws/lambda/cwsyn-*MyCanaryName-randomId* 的 CloudWatch Logs 日志组。

使用金丝雀蓝图

本节提供有关每种金丝雀蓝图的详细信息以及每种蓝图最适合的任务。为以下金丝雀类型提供了蓝图：

- 检测信号监控器
- API 金丝雀
- 无效链接检查器
- 可视化监控
- 金丝雀记录器
- GUI 工作流程

当您使用蓝图创建金丝雀时，在填写 CloudWatch 控制台中的字段时，页面的 Script editor (脚本编辑器) 区域会将您正在创建的金丝雀显示为 Node.js 脚本。您还可以在此区域编辑金丝雀以进一步定制。

检测信号监控

检测信号脚本加载指定的 URL，并存储页面的屏幕截图和 HTTP 归档文件 (HAR 文件)。它们还存储已访问 URL 的日志。

您可以使用 HAR 文件查看有关网页性能的详细数据。您可以分析 Web 请求列表并捕获性能问题，例如某个项的加载时间。

如果您的金丝雀使用 syn-nodejs-puppeteer-3.1 或更高版本的运行时，您可以使用检测信号监控蓝图，以监控多个 URL 并查看状态、持续时间、关联的屏幕截图和金丝雀运行报告的步骤摘要中各个 URL 的故障原因。

API 金丝雀

API 金丝雀可以测试 REST API 的基本读写函数。REST 指表述性状态转移，是开发人员在创建 API 时需要遵循的一组规则。其中一条规则规定，指向特定 URL 的连接应返回一段数据。

金丝雀可与任何 API 结合使用并测试所有类型的功能。每个金丝雀都可以做出多个 API 调用。

在使用 `syn-nodejs-2.2` 或更高版本运行时的金丝雀中，API 金丝雀蓝图支持将 API 作为 HTTP 步骤进行监控的多步骤金丝雀。您可以在单个金丝雀中测试多个 API。每个步骤都是一个单独的请求，可以访问不同的 URL、使用不同的标头以及对是否捕获标题和响应正文使用不同的规则。通过不捕获标头和响应正文，可以防止敏感数据被记录下来。

API 金丝雀中的每个请求均包含以下信息：

- 终端节点，即您请求的 URL。
- 方法，它是发送到服务器的请求类型。REST API 支持 GET（读取）、POST（写入）、PUT（更新）、PATCH（更新）和 DELETE（删除）操作。
- 标头，用于向客户端和服务器提供信息。它们用于身份验证和提供有关正文内容的信息。有关有效标头的列表，请参阅 [HTTP 标头](#)。
- 数据（或正文），其中包含要发送到服务器的信息。这仅用于 POST、PUT、PATCH 或 DELETE 请求。
- 您请求的 URL。

API 金丝雀蓝图支持 GET 和 POST 方法。使用此蓝图时，您必须指定标头。例如，您可以指定 **Authorization** 作为 Key（键），并指定必要的授权数据作为该键的 Value（值）。

如果您在测试 POST 请求，还可以在 Data（数据）字段中指定要发布的内容。

与 API Gateway 的集成

API 蓝图与 Amazon API Gateway 集成。因此，您能够从与该金丝雀所在的同一个 Amazon 账户和区域选择一个 API Gateway API 和阶段，或上载 API Gateway 中的 Swagger 模板以进行跨账户和跨区域 API 监控。然后，您可以在控制台中选择其余详细信息来创建金丝雀，而不是从 Scratch 输入这些详细信息。有关 API Gateway 的更多信息，请参阅 [什么是 Amazon API Gateway？](#)

使用私有 API

您可以在 Amazon API Gateway 中创建使用私有 API 的金丝雀。有关更多信息，请参阅 [在 Amazon API Gateway 中创建私有 API？](#)

失效链接检查器

无效链接检查器通过使用 `document.getElementsByTagName('a')` 收集您正在测试的 URL 内的所有链接。它仅测试您指定数量的链接，而且 URL 本身计为第一个链接。例如，如果要检查包含五个链接的页面上的所有链接，则必须指定金丝雀跟踪六个链接。

使用 `syn-nodejs-2.0-beta` 或更高版本运行时创建的无效链接检查器金丝雀支持以下额外功能：

- 提供包含所检查链接、状态代码、故障原因（如有）以及源页面和目标页面屏幕截图的报告。
- 查看金丝雀结果时，您可以筛选以仅查看无效链接，然后根据故障原因修复链接。
- 此版本会捕获每个链接的带注释源页面屏幕截图，并突出显示链接所在的锚点。不会对隐藏组件添加注释。
- 您可以将此版本配置为捕获源页面和目标页面的屏幕截图、仅捕获源页面或仅捕获目标页面的屏幕截图。
- 此版本修复了早期版本中的一个问题，在早期版本中，即使从第一页抓取了更多链接，金丝雀脚本也会在第一个无效链接后停止。

如果您希望使用 `syn-1.0` 来更新现有金丝雀以使用新的运行时，则必须删除并重新创建金丝雀。这些功能并不会随着将现有金丝雀更新到新运行时而变得可用。

无效链接检查器金丝雀可检测以下类型的链接错误：

- 404 未找到页面
- 主机名无效
- 错误的 URL。例如，URL 缺少括号、包含多余的斜线，或者协议错误。
- 无效的 HTTP 响应代码。
- 主机服务器返回没有内容和没有响应代码的空响应。
- HTTP 请求在金丝雀运行期间持续超时。
- 主机持续丢弃连接，因为它配置错误或太忙。

可视监控蓝图

可视化监控蓝图包含用以比较在金丝雀运行期间捕获的屏幕截图与在基准金丝雀运行期间捕获的屏幕截图的代码。如果两个屏幕截图之间的差异超过阈值百分比，则金丝雀将失败。运行 `syn-puppeteer-node-3.2` 和更高版本的金丝雀支持可视化监控。运行 Python 和 Selenium 的金丝雀中目前不支持可视化监控。

可视化监控蓝图在默认蓝图金丝雀脚本中包含以下代码行，这些代码可实现可视化监控。

```
syntheticsConfiguration.withVisualCompareWithBaseRun(true);
```

将此行添加到脚本后，金丝雀首次成功运行时，它会使用在该运行期间捕获的屏幕截图作为比较基准。在金丝雀的该首次运行之后，您可以使用 CloudWatch 控制台编辑金丝雀以执行以下任一操作：

- 将金丝雀的下次运行设置为新基准。
- 在当前基准屏幕截图上绘制边界，以指定在可视化比较过程中要忽略的屏幕截图区域。
- 删除屏幕截图，使其不用于可视化监控。

有关使用 CloudWatch 控制台编辑金丝雀的更多信息，请参阅 [编辑或删除金丝雀 \(p. 287\)](#)。

您还可以通过使用 `nextrun` 或 `lastrun` 参数或在 [UpdateCanary](#) API 中指定金丝雀运行 ID 来更改用作基准的金丝雀运行。

使用可视化监控蓝图时，您可以输入要在其中捕获屏幕截图的 URL，并将差值阈值指定为百分比。在基准运行之后，只要在金丝雀的未来运行中检测到大于该阈值的可视化差异，便会触发金丝雀故障。在基准运行之后，您还可以编辑金丝雀，以在可视化监控过程中要忽略的基准屏幕截图上“绘制”边界。

可视化监控功能由 ImageMagick 开源软件工具包提供支持。有关更多信息，请参阅 [ImageMagick](#)。

金丝雀记录器

使用金丝雀记录器蓝图，您可以使用 CloudWatch Synthetics Recorder 记录您的点击并在网站上键入操作，并自动生成可用于创建按相同步骤操作的金丝雀的 Node.js 脚本。CloudWatch Synthetics Recorder 是 Amazon 推出的 Google Chrome 扩展程序。

开发团队：CloudWatch Synthetics Recorder 基于[无头记录器](#)。

有关更多信息，请参阅[使用面向 Google Chrome 的 CloudWatch Synthetics Recorder \(p. 225\)](#)。

GUI 工作流程生成器

GUI 工作流程生成器蓝图验证是否可以在您的网页上执行操作。例如，如果您有一个带有登录表单的网页，则金丝雀可以填充用户和密码字段并提交表单，以验证网页是否正常工作。

当您使用蓝图创建此类型的金丝雀时，可以指定希望金丝雀在网页上执行的操作。您可以使用的操作如下：

- 单击 – 选择您要指定的元素，然后模拟用户单击或选择该元素的行为。

若要在 Node.js 脚本中指定该元素，请使用 `[id=]` 或 `a[class=]`。

若要在 Python 脚本中指定该元素，请使用 `xpath //*[@id=]` 或 `//*[@class=]`。

- 验证选择器 – 验证网页上存在指定的元素。此测试对于验证之前的操作是否导致使用正确的元素填充页面非常有用。

若要在 Node.js 脚本中指定待验证的元素，请使用 `[id=]` 或 `a[class=]`。

若要在 Python 脚本中指定待验证的元素，请使用 `xpath //*[@id=]` 或 `//*[@class=]`。

- 验证文本 – 验证指定的字符串是否包含在目标元素中。此测试对于验证之前的操作是否导致显示正确的文本非常有用。

若要在 Node.js 脚本中指定元素，请使用诸如 `div[@id=]//h1` 之类的格式，因为此操作使用 Puppeteer 中的 `waitForXPath` 函数。

若要在 Python 脚本中指定元素，请使用 `xpath` 格式（例如 `//*[@id=]` 或 `//*[@class=]`），因为此操作使用 Selenium 中的 `implicitly_wait` 函数。

- 输入文本 – 在目标元素中写入指定的文本。

若要在 Node.js 脚本中指定待验证的元素，请使用 `[id=]` 或 `a[class=]`。

若要在 Python 脚本中指定待验证的元素，请使用 `xpath //*[@id=]` 或 `//*[@class=]`。

- 单击并导航 – 在选择指定的元素之后，等待整个页面加载。当您需要重新加载页面时，这非常有用。

若要在 Node.js 脚本中指定该元素，请使用 `[id=]` 或 `a[class=]`。

若要在 Python 脚本中指定该元素，请使用 `xpath //*[@id=]` 或 `//*[@class=]`。

例如，以下蓝图使用 Node.js。其会单击指定 URL 上的 `firstButton`，验证预期的选择器是否出现了预期文本，将名称 `Test_Customer` 输入到 Name（名称）字段中，单击 Login（登录）按钮，然后通过检查下一页上的 `Welcome（欢迎）` 文本来验证登录是否成功。

Application or endpoint URL [Info](#)

Enter the endpoint, API or url that you are testing.

Workflow builder

Select the actions you would like the canary to take.

Action	Selector	Text	
Click	[id='firstButton']		Remove action
Verify selector	div[id='screen2Text']		Remove action
Verify text	[@id='screen2Text']//h3	Type	Remove action
Input text	input[id='Name']	Test_Customer	Remove action
Click with navigation	[id='Login']		Remove action
Verify text	div[@id='welcome']//h1	Welcome	Remove action
Add action			

使用以下运行时的 GUI 工作流金丝雀还提供每次金丝雀运行所执行的步骤的摘要。您可以使用与每个步骤关联的屏幕截图和错误消息来查找故障的根本原因。

- syn-nodejs-2.0 或更高版本
- syn-python-selenium-1.0 或更高版本

使用面向 Google Chrome 的 CloudWatch Synthetics Recorder

Amazon 推出 CloudWatch Synthetics Recorder，以帮助您更轻松创建金丝雀。此记录器是一个 Google Chrome 扩展程序。

此记录器记录您在网站上的单击和键入操作，并自动生成可用于创建按相同步骤操作的金丝雀的 Node.js 脚本。

开始记录后，CloudWatch Synthetics Recorder 会在浏览器中检测您的操作并将其转换为脚本。您可以根据需要暂停和恢复记录。当您停止记录时，记录器会将您的操作生成 Node.js 脚本，您可以使用 Copy to Clipboard（复制到剪贴板）按钮轻松复制这些脚本。之后，您可以使用此脚本在 CloudWatch Synthetics 中创建金丝雀。

开发团队：CloudWatch Synthetics Recorder 基于[无头记录器](#)。

安装面向 Google Chrome 的 CloudWatch Synthetics Recorder 扩展程序

若要使用 CloudWatch Synthetics Recorder，您可以先创建一个金丝雀，然后选择 Canary Recorder（金丝雀记录器）蓝图。如果您在尚未下载记录器时执行此操作，则 CloudWatch Synthetics 控制台会提供下载该记录器的链接。

您也可以按照以下步骤直接下载并安装该记录器。

安装 CloudWatch Synthetics Recorder

1. 使用 Google Chrome，访问此网站：<https://chrome.google.com/webstore/detail/cloudwatch-synthetics-rec/bhdnlmmgipmbcdmkkdfplenecpegfno>
2. 选择 Add to Chrome（添加到 Chrome），然后选择 Add extension（添加扩展程序）。

使用面向 Google Chrome 的 CloudWatch Synthetics Recorder

若要使用 CloudWatch Synthetics Recorder 帮助您创建金丝雀，您可以在 CloudWatch 控制台中选择 Create canary（创建金丝雀），然后依次选择 Use a blueprint（使用蓝图）、Canary Recorder（金丝雀记录器）。有关更多信息，请参阅[创建金丝雀 \(p. 219\)](#)。

您也可以使用记录器记录步骤，而不必立即使用它们创建金丝雀。

使用 CloudWatch Synthetics Recorder 记录您在网站上的行为

1. 导航到要监控的页面。
2. 选择 Chrome 扩展程序图标，然后选择 CloudWatch Synthetics Recorder。
3. 选择 Start Recording（开始记录）。
4. 执行您希望记录的步骤。若要暂停记录，请选择 Pause（暂停）。
5. 完成记录工作流后，选择 Stop recording（停止记录）。
6. 选择 Copy to clipboard（复制到剪贴板）以将生成的脚本复制到剪贴板。或者，如果要重新开始，请选择 New recording（新建记录）。
7. 若要使用复制的脚本创建金丝雀，您可以将复制的脚本粘贴到记录器蓝图内联编辑器中，或者将其保存到 Amazon S3 存储桶中，然后从该存储桶中导入。
8. 如果您不立即创建金丝雀，则可以将记录的脚本保存到文件中。

CloudWatch Synthetics Recorder 的已知局限性

面向 Google Chrome 的 CloudWatch Synthetics Recorder 目前具有以下局限性。

- 不具有 ID 的 HTML 元素将使用 CSS 选择器。如果网页结构稍后发生变化，这可能会破坏金丝雀。我们计划在未来版本的记录器中针对此问题提供一些配置选项（例如使用 data-id）。
- 记录器不支持双击或复制/粘贴等操作，也不支持 CMD+0 等组合键。
- 若要验证页面上是否存在某元素或文本，用户必须在脚本后生成添加断言。记录器不支持在不对某元素执行任何操作的情况下验证该元素。这与金丝雀工作流生成器中的“验证文本”或“验证元素”选项类似。我们计划在未来版本的记录器中添加一些断言支持。
- 记录器会记录启动记录的选项卡中的所有操作。其不会记录弹出窗口（例如，允许位置跟踪的弹出窗口）或从弹出窗口到不同页面的导航。

Synthetics 运行时版本

创建或更新金丝雀时，您可以为金丝雀选择 Synthetics 运行时版本。Synthetics 运行时是调用脚本处理程序的 Synthetics 代码，以及捆绑依赖关系的 Lambda 层的组合。

CloudWatch Synthetics 目前支持将 Node.js 用于脚本和 Puppeteer 框架的运行时，以及使用 Python 编写脚本和将 Selenium Webdriver 用于框架的运行时。

我们建议您始终为金丝雀使用最新的运行时版本，以便能够使用最新的功能和对 Synthetics 库进行的更新。

主题

- [CloudWatch Synthetics 运行时支持策略 \(p. 226\)](#)
- [使用 Node.js 和 Puppeteer 的运行时版本 \(p. 230\)](#)
- [使用 Python 和 Selenium Webdriver 的运行时版本 \(p. 236\)](#)

CloudWatch Synthetics 运行时支持策略

Synthetics 运行时版本受维护和安全更新的约束。如果不再支持运行时版本的任何组件，则该 Synthetics 运行时版本将被弃用。

您无法使用已弃用的运行时版本创建金丝雀。使用已弃用运行时的金丝雀将继续运行。您可以停止、启动和删除这些金丝雀。您可以通过将金丝雀更新为使用受支持的运行时版本，来更新使用已弃用运行时版本的现有金丝雀。

如果您有使用计划将在未来 60 天内弃用的运行时的金丝雀，则 CloudWatch Synthetics 会通过电子邮件通知您。我们建议您将金丝雀迁移到受支持的运行时版本，以享受最新版本中包含的新功能、安全性和性能增强的益处。

如何将金丝雀更新到新的运行时版本？

您可以使用 CloudWatch 控制台、Amazon CloudFormation、Amazon CLI 或 Amazon SDK 更新金丝雀的运行时版本。当使用 CloudWatch 控制台更新时，您可以一次最多更新五个金丝雀，方法是在金丝雀列表页面中选中它们，然后选择 Actions（操作）、Update Runtime（更新运行时）。

要验证升级，您可以先使用 CloudWatch 控制台克隆金丝雀，然后更新其运行时版本。这会再创建一个金丝雀，其是原始金丝雀的克隆。在使用新的运行时版本验证了金丝雀后，您便可以更新原始金丝雀的运行时版本并删除克隆金丝雀。

您还可以使用升级脚本更新多个金丝雀。有关更多信息，请参阅[金丝雀运行时升级脚本 \(p. 227\)](#)。

如果升级金丝雀失败，请参阅[排查失败金丝雀的问题 \(p. 275\)](#)。

运行时弃用日期

运行时版本	弃用日期
syn-nodejs-2.2	2021 年 5 月 28 日
syn-nodejs-2.1	2021 年 5 月 28 日
syn-nodejs-2.0	2021 年 5 月 28 日
syn-nodejs-2.0-beta	2021 年 2 月 8 日
syn-1.0	2021 年 5 月 28 日

金丝雀运行时升级脚本

若要将金丝雀脚本升级到支持的运行时版本，请使用以下脚本。

```
const AWS = require('aws-sdk');

// You need to configure your Amazon credentials and Region.
// https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/setting-credentials-node.html
// https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/setting-region.html

const synthetics = new AWS.Synthetics();

const DEFAULT_OPTIONS = {
  /**
   * The number of canaries to upgrade during a single run of this script.
   */
  count: 10,
  /**
   * No canaries are upgraded unless force is specified.
   */
  force: false
};

/**
 * The number of milliseconds to sleep between GetCanary calls when
 * verifying that an update succeeded.
 */
const SLEEP_TIME = 5000;

(async () => {
  try {
    const options = getOptions();

    const versions = await getRuntimeVersions();
    const canaries = await getAllCanaries();
    const upgrades = canaries
      .filter(canary => !versions.isLatestVersion(canary.RuntimeVersion))
      .map(canary => {
        return {
          Name: canary.Name,
          FromVersion: canary.RuntimeVersion,
          ToVersion: versions.getLatestVersion(canary.RuntimeVersion)
        };
      });

    if (options.force) {
      const promises = [];
```

```

for (const upgrade of upgrades.slice(0, options.count)) {
  const promise = upgradeCanary(upgrade);
  promises.push(promise);
  // Sleep for 100 milliseconds to avoid throttling.
  await usleep(100);
}

const succeeded = [];
const failed = [];
for (let i = 0; i < upgrades.slice(0, options.count).length; i++) {
  const upgrade = upgrades[i];
  const promise = promises[i];
  try {
    await promise;
    console.log(`The update of ${upgrade.Name} succeeded.`);
    succeeded.push(upgrade.Name);
  } catch (e) {
    console.log(`The update of ${upgrade.Name} failed with error: ${e}`);
    failed.push({
      Name: upgrade.Name,
      Reason: e
    });
  }
}

if (succeeded.length) {
  console.group('The following canaries were upgraded successfully.');
```

```

  for (const name of succeeded) {
    console.log(name);
  }
  console.groupEnd()
} else {
  console.log('No canaries were upgraded successfully.');
```

```

}

if (failed.length) {
  console.group('The following canaries were not upgraded successfully.');
```

```

  for (const failure of failed) {
    console.log(`\x1b[31m`, `${failure.Name}: ${failure.Reason}`, '\x1b[0m');
```

```

  }
  console.groupEnd();
}
} else {
  console.log('Run with --force [--count <count>] to perform the first <count> upgrades
shown. The default value of <count> is 10.')
```

```

  console.table(upgrades);
}
} catch (e) {
  console.error(e);
}
})();

function getOptions() {
  const force = getFlag('--force', DEFAULT_OPTIONS.force);
  const count = getOption('--count', DEFAULT_OPTIONS.count);
  return { force, count };

  function getFlag(key, defaultValue) {
    return process.argv.includes(key) || defaultValue;
  }
  function getOption(key, defaultValue) {
    const index = process.argv.indexOf(key);
    if (index < 0) {
      return defaultValue;
    }
    const value = process.argv[index + 1];

```

```
    if (typeof value === 'undefined' || value.startsWith('-')) {
      throw `The ${key} option requires a value.`;
    }
    return value;
  }
}

function getAllCanaries() {
  return new Promise((resolve, reject) => {
    const canaries = [];

    synthetics.describeCanaries().eachPage((err, data) => {
      if (err) {
        reject(err);
      } else {
        if (data === null) {
          resolve(canaries);
        } else {
          canaries.push(...data.Canaries);
        }
      }
    });
  });
}

function getRuntimeVersions() {
  return new Promise((resolve, reject) => {
    const jsVersions = [];
    const pythonVersions = [];
    synthetics.describeRuntimeVersions().eachPage((err, data) => {
      if (err) {
        reject(err);
      } else {
        if (data === null) {
          jsVersions.sort((a, b) => a.ReleaseDate - b.ReleaseDate);
          pythonVersions.sort((a, b) => a.ReleaseDate - b.ReleaseDate);
          resolve({
            isLatestVersion(version) {
              const latest = this.getLatestVersion(version);
              return latest === version;
            },
            getLatestVersion(version) {
              if (jsVersions.some(v => v.VersionName === version)) {
                return jsVersions[jsVersions.length - 1].VersionName;
              } else if (pythonVersions.some(v => v.VersionName === version)) {
                return pythonVersions[pythonVersions.length - 1].VersionName;
              } else {
                throw Error(`Unknown version ${version}`);
              }
            }
          });
        } else {
          for (const version of data.RuntimeVersions) {
            if (version.VersionName === 'syn-1.0') {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-nodejs-2.')) {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-nodejs-puppeteer-')) {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-python-selenium-')) {
              pythonVersions.push(version);
            } else {
              throw Error(`Unknown version ${version.VersionName}`);
            }
          }
        }
      }
    });
  });
}
```

```
    }  
  });  
});  
}  
  
async function upgradeCanary(upgrade) {  
  console.log(`Upgrading canary ${upgrade.Name} from ${upgrade.FromVersion} to  
    ${upgrade.ToVersion}`);  
  await synthetics.updateCanary({ Name: upgrade.Name, RuntimeVersion:  
    upgrade.ToVersion }).promise();  
  while (true) {  
    await usleep(SLEEP_TIME);  
    console.log(`Getting the state of canary ${upgrade.Name}`);  
    const response = await synthetics.getCanary({ Name: upgrade.Name }).promise();  
    const state = response.Canary.Status.State;  
    console.log(`The state of canary ${upgrade.Name} is ${state}`);  
    if (state === 'ERROR' || response.Canary.Status.StateReason) {  
      throw response.Canary.Status.StateReason;  
    }  
    if (state !== 'UPDATING') {  
      return;  
    }  
  }  
}  
  
function usleep(ms) {  
  return new Promise(resolve => setTimeout(resolve, ms));  
}
```

使用 Node.js 和 Puppeteer 的运行时版本

使用 Node.js 和 Puppeteer 的第一个运行时版本被命名为 syn-1.0。后续运行时版本采用命名约定 syn-*language-majorversion.minorversion*。从 syn-nodejs-puppeteer-3.0 开始，命名约定为 syn-*language-framework-majorversion.minorversion*

额外添加的 -beta 后缀用以显示运行时版本当前为测试预览版。

具有相同主版本号的运行时版本将始终向后兼容。

关于所有运行时版本的说明

使用 syn-nodejs-puppeteer-3.0 运行时版本时，请确保金丝雀脚本与 Node.js 12.x 兼容。如果使用早期 syn-nodejs 运行时版本，请确保脚本与 Node.js 10.x 兼容。

金丝雀中的 Lambda 代码已配置为具有最多 1 GB 内存。在配置的超时值之后，该超时将适用于每个运行的金丝雀。如果未为金丝雀指定超时值，则 CloudWatch 会根据金丝雀的频率来选择超时值。如要配置超时值，请使此值不要短于 15 秒钟，以预留 Lambda 冷启动以及启动金丝雀工具的时间。

syn-nodejs-puppeteer-3.7

syn-nodejs-puppeteer-3.7 运行时是 Node.js 和 Puppeteer 的最新运行时版本。

主要依赖项：

- Lambda 运行时 Node.js 14.x
- Puppeteer-core 10.1.0 版
- Chromium 92.0.4512 版

syn-nodejs-puppeteer-3.7 中的新功能：

- 日志记录增强功能 — 即使 Amazon S3 超时或崩溃，金丝雀也会将日志上传到 Amazon S3。

- Lambda 层大小已减小 — 用于金丝雀的 Lambda 层的大小减小了 34%。

syn-nodejs-puppeteer-3.7 中的错误修复：

- 错误修复 — 日语、简体中文和繁体中文字体将正确呈现。

[syn-nodejs-puppeteer-3.6](#)

主要依赖项：

- Lambda 运行时 Node.js 14.x
- Puppeteer-core 10.1.0 版
- Chromium 92.0.4512 版

syn-nodejs-puppeteer-3.6 中的新功能：

- 更精确的时间戳 — 金丝雀运行的开始时间和停止时间现在精确到毫秒。

[syn-nodejs-puppeteer-3.5](#)

主要依赖项：

- Lambda 运行时 Node.js 14.x
- Puppeteer-core 10.1.0 版
- Chromium 92.0.4512 版

syn-nodejs-puppeteer-3.5 中的新功能：

- 更新的项目依赖项 — 此运行时中唯一的新功能是更新的项目依赖项。

[syn-nodejs-puppeteer-3.4](#)

主要依赖项：

- Lambda 运行时 Node.js 12.x
- Puppeteer-core 5.5.0 版
- Chromium 88.0.4298.0 版

syn-nodejs-puppeteer-3.4 中的新功能：

- 自定义处理程序函数 — 您现在可以对金丝雀脚本使用自定义处理程序函数。之前的运行时要求脚本入口点包括 `.handler`。

您还可以将金丝雀脚本放在任何文件夹中，并将文件夹名称作为处理程序的一部分进行传递。例如，`MyFolder/MyScriptFile.functionname` 可以用作入口点。

- 扩展的 HAR 文件信息 — 您现在可以在金丝雀生成的 HAR 文件中看到恶意、待处理和未完成的请求。

[syn-nodejs-puppeteer-3.3](#)

主要依赖项：

- Lambda 运行时 Node.js 12.x

- Puppeteer-core 5.5.0 版
- Chromium 88.0.4298.0 版

syn-nodejs-puppeteer-3.3 中的新功能：

- 用于构件加密的更多选项 – 对于使用此运行时或更高版本运行时（而不是使用 Amazon 托管式密钥）来加密金丝雀存储在 Amazon S3 中的构件的金丝雀，您可以选择使用 Amazon KMS 客户托管式密钥或 Amazon S3 托管式密钥。有关更多信息，请参阅[加密金丝雀构件 \(p. 283\)](#)。

[syn-nodejs-puppeteer-3.2](#)

主要依赖项：

- Lambda 运行时 Node.js 12.x
- Puppeteer-core 5.5.0 版
- Chromium 88.0.4298.0 版

syn-nodejs-puppeteer-3.2 中的新功能：

- 使用屏幕截图进行可视化监控 – 使用此运行时或更高版本运行时的金丝雀可以将运行期间捕获的屏幕截图与相同屏幕截图的基准版本进行比较。如果屏幕截图与指定百分比阈值之间的差异较大，则金丝雀将失败。有关更多信息，请参阅[可视化监控 \(p. 253\)](#) 或 [可视监控蓝图 \(p. 223\)](#)。
- 与敏感数据相关的新函数：您可以阻止敏感数据出现在金丝雀日志和报告中。有关更多信息，请参阅[SyntheticsLogHelper 类 \(p. 255\)](#)。
- 已弃用的函数：RequestResponseLogHelper 类已弃用，以支持其他新配置选项。有关更多信息，请参阅[RequestResponseLogHelper 类 \(p. 260\)](#)。

[syn-nodejs-puppeteer-3.1](#)

主要依赖项：

- Lambda 运行时 Node.js 12.x
- Puppeteer-core 5.5.0 版
- Chromium 88.0.4298.0 版

syn-nodejs-puppeteer-3.1 中的新功能：

- CloudWatch 指标配置功能 – 使用此运行时，您可以禁用不需要的指标。否则，金丝雀会发布每次金丝雀运行的各种 CloudWatch 指标。
- 屏幕截图链接 – 您可以在金丝雀步骤完成后将屏幕截图链接到该步骤。为此，您可以使用 takeScreenshot 方法捕获屏幕截图，并使用要与屏幕截图关联的步骤的名称。例如，您可能想要执行某个步骤，添加等待时间，然后捕获屏幕截图。
- 检测信号监控器蓝图可以监控多个 URL – 您可以使用 CloudWatch 控制台中的检测信号监控蓝图来监控多个 URL，并在金丝雀运行报告的步骤摘要中查看每个 URL 的状态、持续时间、关联的屏幕截图和故障原因。

[syn-nodejs-puppeteer-3.0](#)

主要依赖项：

- Lambda 运行时 Node.js 12.x
- Puppeteer-core 5.5.0 版

- Chromium 88.0.4298.0 版

syn-nodejs-puppeteer-3.0 中的新功能：

- 升级的依赖项 – 此版本使用 Puppeteer 5.5.0 版、Node.js 12.x 和 Chromium 88.0.4298.0。
- 跨区域存储桶访问 – 您现在可以指定另一个区域中的 S3 存储桶作为金丝雀存储其日志文件、屏幕截图和 HAR 文件的存储桶。
- 可用的新函数 – 此版本添加了库函数来检索金丝雀名称和 Synthetics 运行时版本。

有关更多信息，请参阅[Synthetics 类 \(p. 245\)](#)。

syn-nodejs-2.2

本节包含有关 syn-nodejs-2.2 运行时版本的信息。

Important

此运行时版本计划于 2021 年 5 月 28 日弃用。有关更多信息，请参阅[CloudWatch Synthetics 运行时支持策略 \(p. 226\)](#)。

主要依赖项：

- Lambda 运行时 Node.js 10.x
- Puppeteer-core 3.3.0 版
- Chromium 83.0.4103.0 版

syn-nodejs-2.2 中的新功能：

- 作为 HTTP 步骤监控您的金丝雀 – 您现在可以在一个金丝雀中测试多个 API。每个 API 都作为单独的 HTTP 步骤进行测试，CloudWatch Synthetics 会使用步骤指标和 CloudWatch Synthetics 步骤报告监控每个步骤的状态。CloudWatch Synthetics 会为每个 HTTP 步骤创建 SuccessPercent 和 Duration 指标。

此功能由 executeHttpStep(stepName, requestOptions, callback, stepConfig) 函数实现。有关更多信息，请参阅[executeHttpStep\(stepName, requestOptions, \[callback\], \[stepConfig\]\) \(p. 264\)](#)。

API 金丝雀蓝图已更新为使用此新功能。

- HTTP 请求报告 – 您现在可以查看详细的 HTTP 请求报告，这些报告捕获的是请求/响应标头、响应正文、状态代码、错误和性能计时、TCP 连接时间、TLS 握手时间、第一字节时间和内容传输时间等详细信息。所有使用后台 HTTP/HTTPS 模块的 HTTP 请求都在此处捕获。预设情况下不会捕获标头和响应正文，但可以通过设置配置选项来启用捕获标头和响应正文。
- 全局和步骤级配置 – 您可以在全局级别设置 CloudWatch Synthetics 配置，这些配置应用于金丝雀的所有步骤。您还可以通过传递配置键/值对来启用或禁用某些选项，在步骤级别覆盖这些配置。

有关更多信息，请参阅[SyntheticsConfiguration 类 \(p. 246\)](#)。

- 继续执行步骤故障配置 – 您可以选择在步骤故障时继续执行金丝雀。对于 executeHttpStep 函数，预设情况下，此选项处于启用状态。您可以在全局级别设置此选项一次，或者针对每个步骤做出不同的设置。

syn-nodejs-2.1

Important

此运行时版本计划于 2021 年 5 月 28 日弃用。有关更多信息，请参阅[CloudWatch Synthetics 运行时支持策略 \(p. 226\)](#)。

主要依赖项：

- Lambda 运行时 Node.js 10.x
- Puppeteer-core 3.3.0 版
- Chromium 83.0.4103.0 版

syn-nodejs-2.1 中的新功能：

- 可配置的屏幕截图行为 – 提供关闭 UI 金丝雀捕获屏幕截图的功能。在使用以前版本运行时的金丝雀中，UI 金丝雀始终在每个步骤前后捕获屏幕截图。在 syn-nodejs-2.1 中，可配置此行为。关闭屏幕截图可以降低您的 Amazon S3 存储成本，并有助于您遵守 HIPAA 法规。有关更多信息，请参阅 [SyntheticsConfiguration 类 \(p. 246\)](#)。
- 自定义 Google Chrome 启动参数：现在，您可以配置当金丝雀启动 Google Chrome 浏览器窗口时使用的参数。有关更多信息，请参阅 [launch\(options\) \(p. 259\)](#)。

与更早版本的金丝雀运行时相比，使用 syn-nodejs-2.0 或更高版本时，金丝雀持续时间可能会略有增加。

syn-nodejs-2.0

Important

此运行时版本计划于 2021 年 5 月 28 日弃用。有关更多信息，请参阅 [CloudWatch Synthetics 运行时支持策略 \(p. 226\)](#)。

主要依赖项：

- Lambda 运行时 Node.js 10.x
- Puppeteer-core 3.3.0 版
- Chromium 83.0.4103.0 版

syn-nodejs-2.0 中的新功能：

- 升级的依赖项 – 此运行时版本使用 Puppeteer-core 3.3.0 版和 Chromium 83.0.4103.0 版
- 支持 X-Ray 活动跟踪。金丝雀启用跟踪后，将针对由使用浏览器、Amazon SDK 或 HTTP 或 HTTPS 模块的金丝雀发出的所有调用发送 X-Ray 跟踪。启用了跟踪的金丝雀会显示在 CloudWatch ServiceLens 和 X-Ray 中的服务地图上，即使它们没有向启用了跟踪的其他服务或应用程序发送请求。有关更多信息，请参阅 [金丝雀和 X-Ray 跟踪 \(p. 282\)](#)。
- Synthetics 报告 – CloudWatch Synthetics 会针对每次金丝雀运行创建一个名为 SyntheticsReport-PASSED.json 或 SyntheticsReport-FAILED.json 的报告，其中记录了开始时间、结束时间、状态和故障等数据。报告中还记录金丝雀脚本的每个步骤的 PASSED/FAILED 状态，以及每个步骤捕获的失败和屏幕截图。
- 无效链接检查器报告 – 此运行时中包含的新版本无效链接检查器会创建一个报告，其中包括已检查的链接、状态代码、故障原因（如有）以及源页面和目标页面屏幕截图。
- 新 CloudWatch 指标 – Synthetics 在 CloudWatchSynthetics 命名空间中发布名为 2xx、4xx、5xx 和 RequestFailed 的指标。这些指标会显示金丝雀运行中 200 秒、400 秒、500 秒和请求故障的数量。在此运行时版本中，这些指标仅针对 UI 金丝雀报告，而不针对 API 金丝雀报告。从 syn-nodejs-puppeteer-2.2 运行时版本开始，也针对 API 金丝雀报告这些指标。
- 可排序的 HAR 文件 – 您现在可以按状态代码、请求大小和持续时间对 HAR 文件进行排序。
- 指标时间戳 – 现在基于 Lambda 调用时间而非金丝雀运行结束时间报告 CloudWatch 指标。

syn-nodejs-2.0 中的错误修复：

- 修复了不报告金丝雀构件上载错误的问题。此类错误现在显示为执行错误。

- 修复了将重定向请求 (3xx) 误记为错误的问题。
- 修复了从 0 开始编号屏幕截图的问题。现在应从 1 开始编号。
- 修复了中文和日文字体屏幕截图乱码的问题。

与更早版本的金丝雀运行时相比，使用 syn-nodejs-2.0 或更高版本时，金丝雀持续时间可能会略有增加。

syn-nodejs-2.0-beta

Important

此运行时版本已于 2021 年 2 月 8 日弃用。有关更多信息，请参阅[CloudWatch Synthetics 运行时支持策略 \(p. 226\)](#)。

主要依赖项：

- Lambda 运行时 Node.js 10.x
- Puppeteer-core 3.3.0 版
- Chromium 83.0.4103.0 版

syn-nodejs-2.0-beta 中的新功能：

- 升级的依赖项 – 此运行时版本使用 Puppeteer-core 3.3.0 版和 Chromium 83.0.4103.0 版
- Synthetics 报告 – CloudWatch Synthetics 会针对每次金丝雀运行创建一个名为 SyntheticsReport-PASSED.json 或 SyntheticsReport-FAILED.json 的报告，其中记录了开始时间、结束时间、状态和故障等数据。报告中还记录金丝雀脚本的每个步骤的 PASSED/FAILED 状态，以及每个步骤捕获的失败和屏幕截图。
- 无效链接检查器报告 – 此运行时中包含的新版本无效链接检查器会创建一个报告，其中包括已检查的链接、状态代码、故障原因（如有）以及源页面和目标页面屏幕截图。
- 新 CloudWatch 指标 – Synthetics 在 CloudWatchSynthetics 命名空间中发布名为 2xx、4xx、5xx 和 RequestFailed 的指标。这些指标会显示金丝雀运行中 200 秒、400 秒、500 秒和请求故障的数量。仅针对 UI 金丝雀报告而不针对 API 金丝雀报告这些指标。
- 可排序的 HAR 文件 – 您现在可以按状态代码、请求大小和持续时间对 HAR 文件进行排序。
- 指标时间戳 – 现在基于 Lambda 调用时间而非金丝雀运行结束时间报告 CloudWatch 指标。

syn-nodejs-2.0-beta 中的错误修复：

- 修复了不报告金丝雀构件上载错误的问题。此类错误现在显示为执行错误。
- 修复了将重定向请求 (3xx) 误记为错误的问题。
- 修复了从 0 开始编号屏幕截图的问题。现在应从 1 开始编号。
- 修复了中文和日文字体屏幕截图乱码的问题。

syn-1.0

Important

此运行时版本计划于 2021 年 5 月 28 日弃用。有关更多信息，请参阅[CloudWatch Synthetics 运行时支持策略 \(p. 226\)](#)。

第一个 Synthetics 运行时版本是 syn-1.0。

主要依赖项：

- Lambda 运行时 Node.js 10.x

- Puppeteer-core 1.14.0 版
- 与 Puppeteer-core 1.14.0 匹配的 Chromium 版本

使用 Python 和 Selenium Webdriver 的运行时版本

以下各节包含有关用于 Python 和 Selenium Webdriver 的 CloudWatch Synthetics 运行时版本的信息。Selenium 是一种开源浏览器自动化工具。有关 Selenium 的更多信息，请参阅 www.selenium.dev/

这些运行时版本的命名约定为 `syn-language-framework-majorversion.minorversion`。

syn-python-selenium-1.3

主要依赖项：

- Python 3.8
- Selenium 3.141.0
- Chromium 92.0.4512.0 版

syn-python-selenium-1.3 中的新功能：

- 更精确的时间戳 — 金丝雀运行的开始时间和停止时间现在精确到毫秒。

syn-python-selenium-1.2

主要依赖项：

- Python 3.8
- Selenium 3.141.0
- Chromium 92.0.4512.0 版

- 更新的项目依赖项 — 此运行时中唯一的新功能是更新的项目依赖项。

syn-python-selenium-1.1

主要依赖项：

- Python 3.8
- Selenium 3.141.0
- Chromium 83.0.4103.0 版

功能：

- 自定义处理程序函数 — 您现在可以对金丝雀脚本使用自定义处理程序函数。之前的运行时要求脚本入口点包括 `.handler`。

您还可以将金丝雀脚本放在任何文件夹中，并将文件夹名称作为处理程序的一部分进行传递。例如，`MyFolder/MyScriptFile.functionname` 可以用作入口点。

- 用于添加指标和步骤失败配置的配置选项 — 这些选项已在适用于 Node.js 金丝雀的运行时中提供。有关更多信息，请参阅 [SyntheticsConfiguration 类](#) (p. 266)。
- Chrome 中的自定义参数 — 您现在可以用无痕模式打开浏览器或用代理服务器配置进行传递。有关更多信息，请参阅 [Chrome\(\)](#) (p. 273)。

- 跨区域构件存储桶— 金丝雀可以将其构件存储在不同区域的 Simple Storage Service (Amazon S3) 存储桶中。
- 错误修复，包括修复 `index.py` 问题— 对于之前的运行时，名为 `index.py` 的金丝雀文件导致异常，因为其名称与库文件的名称冲突。此问题现已修复。

[syn-python-selenium-1.0](#)

主要依赖项：

- Python 3.8
- Selenium 3.141.0
- Chromium 83.0.4103.0 版

功能：

- Selenium 支持 – 您可以使用 Selenium 测试框架编写金丝雀脚本。您可以将其他地方的 Selenium 脚本添加到 CloudWatch Synthetics 中，只需做出最少的更改，它们即可用于 Amazon 服务。

编写金丝雀脚本

以下部分介绍了如何编写金丝雀脚本以及如何将金丝雀与其他 Amazon 服务集成。

主题

- [编写 Node.js 金丝雀脚本 \(p. 237\)](#)
- [编写 Python 金丝雀脚本 \(p. 243\)](#)
- [更改现有 Puppeteer 脚本以使用 Synthetics 金丝雀 \(p. 244\)](#)

编写 Node.js 金丝雀脚本

主题

- [从 Scratch 创建 CloudWatch Synthetics 金丝雀 \(p. 237\)](#)
- [将金丝雀文件打包 \(p. 238\)](#)
- [更改现有 Puppeteer 脚本以将其用作 Synthetics 金丝雀 \(p. 239\)](#)
- [环境变量 \(p. 240\)](#)
- [将您的金丝雀与其他 Amazon 服务集成 \(p. 241\)](#)
- [强制金丝雀使用静态 IP 地址 \(p. 243\)](#)

[从 Scratch 创建 CloudWatch Synthetics 金丝雀](#)

这里是一个极简 Synthetics 金丝雀脚本示例。此脚本将成功通过一次运行，并返回一个字符串。要查看失败的金丝雀示例，请将 `let fail = false;` 更改为 `let fail = true;`。

您必须为金丝雀脚本定义入口点函数。要查看如何将文件上载到指定作为金丝雀 `ArtifactS3Location` 的 Amazon S3 位置，请在 `/tmp` 文件夹下创建这些文件。脚本运行后，将“通过/失败”状态和持续时间指标发布到 CloudWatch，并将 `/tmp` 下的文件上载到 S3。

```
const basicCustomEntryPoint = async function () {  
  
    // Insert your code here
```

```
// Perform multi-step pass/fail check

// Log decisions made and results to /tmp

// Be sure to wait for all your code paths to complete
// before returning control back to Synthetics.
// In that way, your canary will not finish and report success
// before your code has finished executing

// Throw to fail, return to succeed
let fail = false;
if (fail) {
    throw "Failed basicCanary check.";
}

return "Successfully completed basicCanary checks.";
};

exports.handler = async () => {
    return await basicCustomEntryPoint();
};
```

接下来，我们将扩展脚本以使用 Synthetics 日志记录并使用 Amazon SDK 进行调用。出于演示目的，此脚本将创建 Amazon DynamoDB 客户端并调用 DynamoDB listTables API。它会记录对请求的响应，并根据请求是否成功来记录通过还是失败。

```
const log = require('SyntheticsLogger');
const AWS = require('aws-sdk');
// Require any dependencies that your script needs
// Bundle additional files and dependencies into a .zip file with folder structure
// nodejs/node_modules/additional files and folders

const basicCustomEntryPoint = async function () {

    log.info("Starting DynamoDB:listTables canary.");

    let dynamodb = new AWS.DynamoDB();
    var params = {};
    let request = await dynamodb.listTables(params);
    try {
        let response = await request.promise();
        log.info("listTables response: " + JSON.stringify(response));
    } catch (err) {
        log.error("listTables error: " + JSON.stringify(err), err.stack);
        throw err;
    }

    return "Successfully completed DynamoDB:listTables canary.";
};

exports.handler = async () => {
    return await basicCustomEntryPoint();
};
```

将金丝雀文件打包

如果您使用 Simple Storage Service (Amazon S3) 位置上传金丝雀脚本，则 zip 文件应在此文件夹结构下包含脚本: nodejs/node_modules/*myCanaryFilename.js file*。

如果您有多个 .js 文件，或者您的脚本依赖于某个依赖项，则您可以将它们打包到包含文件夹结构 nodejs/node_modules/*myCanaryFilename.js file and other folders and files* 的单个 ZIP 文件中。如果您使用 syn-nodejs-puppeteer-3.4 或者之后版本，

您可以选择将金丝雀文件放在另一个文件夹中，然后创建与此类似的文件夹结构: `nodejs/node_modules/myFolder/myCanaryFilename.js` *file and other folders and files*。

处理程序名称

请务必将您的金丝雀脚本入口点（处理程序）设置为 `myCanaryFilename.functionName`，以匹配脚本入口点的文件名。如果您使用 `syn-nodejs-puppeteer-3.4` 之前版本的运行时，则 `functionName` 必须是 `handler`。如果您使用 `syn-nodejs-puppeteer-3.4` 或之后版本，您可以选择任何函数名称作为处理程序。如果您使用 `syn-nodejs-puppeteer-3.4` 或之后版本，您也可以选择将金丝雀存储在单独的文件夹中，例如 `nodejs/node_modules/myFolder/my_canary_filename`。如果将其存储在单独的文件夹中，请在脚本入口点中指定该路径，例如 `myFolder/my_canary_filename.functionName`。

更改现有 Puppeteer 脚本以将其用作 Synthetics 金丝雀

本节介绍如何对 Puppeteer 脚本进行修改，以将其作为 Synthetics 金丝雀脚本运行。有关 Puppeteer 的更多信息，请参阅 [Puppeteer API v1.14.0](#)。

我们从这个示例 Puppeteer 开始：

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await page.screenshot({path: 'example.png'});

  await browser.close();
})();
```

转换步骤如下：

- 创建和导出 `handler` 函数。处理程序是脚本的入口点函数。如果您使用 `syn-nodejs-puppeteer-3.4` 之前版本的运行时，处理程序函数必须命名为 `handler`。如果您使用 `syn-nodejs-puppeteer-3.4` 或之后版本，函数可以有任何名称，但必须与脚本中使用的名称相同。此外，如果您使用 `syn-nodejs-puppeteer-3.4` 或之后版本，您可以将脚本存储在任何文件夹下，并将该文件夹指定为处理程序名称的一部分。

```
const basicPuppeteerExample = async function () {};

exports.handler = async () => {
  return await basicPuppeteerExample();
};
```

- 使用 Synthetics 依赖项。

```
var synthetics = require('Synthetics');
```

- 使用 `Synthetics.getPage` 函数获取 Puppeteer Page 对象。

```
const page = await synthetics.getPage();
```

`Synthetics.getPage` 函数返回的页面对象指示需要记录 `page.on request`、`response` 和 `requestfailed` 事件。Synthetics 还为页面上的请求和响应设置 HAR 文件生成，并将金丝雀 ARN 添加到页面上的传出请求的 `user-agent` 标头。

该脚本现已准备好作为 Synthetics 金丝雀运行。更新的脚本如下：


```
var synthetics = require('Synthetics'); // Synthetics dependency

const basicPuppeteerExample = async function () {
  const page = await synthetics.getPage(); // Get instrumented page from Synthetics
  await page.goto('https://example.com');
  await page.screenshot({path: '/tmp/example.png'}); // Write screenshot to /tmp folder
};

exports.handler = async () => { // Exported handler function
  return await basicPuppeteerExample();
};
```

环境变量

您可以在创建金丝雀时使用环境变量。这样，您就能够编写单个金丝雀脚本，然后将该脚本与不同的值相结合，快速创建具有类似任务的多个金丝雀。

例如，假定您的企业具有用于不同软件开发阶段的 prod、dev 和 pre-release 端点，而您需要创建金丝雀来测试其中每个端点。您可以编写一个测试软件的金丝雀脚本，然后在分别创建三个金丝雀时为端点环境变量指定不同的值。之后，在创建金丝雀时，您可以指定要用于环境变量的脚本和值。

环境变量的名称可以包含字母、数字和下划线字符，必须以字母开头，并且至少为两个字符。环境变量的总大小不能超过 4 KB。不能将任何 Lambda 预留环境变量指定为环境变量的名称。有关预留环境变量的更多信息，请参阅[运行时环境变量](#)。

Important

环境变量键和值未加密。请勿在其中存储敏感信息。

下面的示例脚本使用两个环境变量。此脚本用于检查网页是否可用的金丝雀。该金丝雀使用环境变量来参数化所检查的 URL 和所使用的 CloudWatch Synthetics 日志级别。

以下函数将 LogLevel 设置为 LOG_LEVEL 环境变量的值。

```
synthetics.setLogLevel(process.env.LOG_LEVEL);
```

此函数将 URL 设置为 URL 环境变量的值。

```
const URL = process.env.URL;
```

以下是完整的脚本。当您使用此脚本创建金丝雀时，您可以指定 LOG_LEVEL 和 URL 环境变量的值。

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const pageLoadEnvironmentVariable = async function () {

  // Setting the log level (0-3)
  synthetics.setLogLevel(process.env.LOG_LEVEL);
  // INSERT URL here
  const URL = process.env.URL;

  let page = await synthetics.getPage();
  //You can customize the wait condition here. For instance,
  //using 'networkidle2' may be less restrictive.
  const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout: 30000});
  if (!response) {
    throw "Failed to load page!";
  }
}
```

```
//Wait for page to render.
//Increase or decrease wait time based on endpoint being monitored.
await page.waitFor(15000);
await synthetics.takeScreenshot('loaded', 'loaded');
let pageTitle = await page.title();
log.info('Page title: ' + pageTitle);
log.debug('Environment variable:' + process.env.URL);

//If the response status code is not a 2xx success code
if (response.status() < 200 || response.status() > 299) {
  throw "Failed to load page!";
}
};

exports.handler = async () => {
  return await pageLoadEnvironmentVariable();
};
```

将环境变量传递到脚本

要在控制台中创建金丝雀时将环境变量传递给脚本，请在控制台上的 Environment variables (环境变量) 部分指定环境变量的密钥和值。有关更多信息，请参阅[创建金丝雀 \(p. 219\)](#)。

要通过 API 或 Amazon CLI 传递环境变量，请使用 RunConfig 部分中的 EnvironmentVariables 参数。以下为 Amazon CLI 命令示例，该命令创建一个使用两个环境变量 (具有 Environment 和 Region 密钥) 的金丝雀。

```
aws synthetics create-canary --cli-input-json '{
  "Name": "nameofCanary",
  "ExecutionRoleArn": "roleArn",
  "ArtifactS3Location": "s3://cw-syn-results-123456789012-us-west-2",
  "Schedule": {
    "Expression": "rate(0 minute)",
    "DurationInSeconds": 604800
  },
  "Code": {
    "S3Bucket": "canarycreation",
    "S3Key": "cwsyn-mycanaryheartbeat-12345678-d1bd-1234-
abcd-123456789012-12345678-6a1f-47c3-b291-123456789012.zip",
    "Handler": "pageLoadBlueprint.handler"
  },
  "RunConfig": {
    "TimeoutInSeconds": 60,
    "EnvironmentVariables": {
      "Environment": "Production",
      "Region": "us-west-1"
    }
  },
  "SuccessRetentionPeriodInDays": 13,
  "FailureRetentionPeriodInDays": 13,
  "RuntimeVersion": "syn-nodejs-2.0"
}'
```

将您的金丝雀与其他 Amazon 服务集成

所有金丝雀都可以使用 Amazon SDK 库。在编写金丝雀时，您可以使用此库将金丝雀与其他 Amazon 服务集成。

为此，您需要将以下代码添加到您的金丝雀中。对于这些例子，Amazon Secrets Manager 用作与金丝雀集成的服务。

- 导入 Amazon SDK。

```
const AWS = require('aws-sdk');
```

- 为要集成的 Amazon 服务创建客户端。

```
const secretsManager = new AWS.SecretsManager();
```

- 使用客户端对该服务进行 API 调用。

```
var params = {
  SecretId: secretName
};
return await secretsManager.getSecretValue(params).promise();
```

下面的金丝雀脚本代码段更详细地演示了与 Secrets Manager 集成的示例。

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const AWS = require('aws-sdk');
const secretsManager = new AWS.SecretsManager();

const getSecrets = async (secretName) => {
  var params = {
    SecretId: secretName
  };
  return await secretsManager.getSecretValue(params).promise();
}

const secretsExample = async function () {
  let URL = "<URL>";
  let page = await synthetics.getPage();

  log.info(`Navigating to URL: ${URL}`);
  const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout: 30000});

  // Fetch secrets
  let secrets = await getSecrets("secretname")

  /**
   * Use secrets to login.
   *
   * Assuming secrets are stored in a JSON format like:
   * {
   *   "username": "<USERNAME>",
   *   "password": "<PASSWORD>"
   * }
   */
  let secretsObj = JSON.parse(secrets.SecretString);
  await synthetics.executeStep('login', async function () {
    await page.type(">USERNAME-INPUT-SELECTOR<", secretsObj.username);
    await page.type(">PASSWORD-INPUT-SELECTOR<", secretsObj.password);

    await Promise.all([
      page.waitForNavigation({ timeout: 30000 }),
      await page.click(">SUBMIT-BUTTON-SELECTOR<")
    ]);
  });

  // Verify login was successful
  await synthetics.executeStep('verify', async function () {
    await page.waitForXPath(">SELECTOR<", { timeout: 30000 });
  });
}
```

```
});  
};  
  
exports.handler = async () => {  
    return await secretsExample();  
};
```

强制金丝雀使用静态 IP 地址

您可以设置金丝雀，使其使用静态 IP 地址。

强制金丝雀使用静态 IP 地址

1. 创建新的 VPC。有关更多信息，请参阅[在您的 VPC 中使用 DNS](#)。
2. 创建新的互联网网关。有关更多信息，请参阅[在您的 VPC 中添加互联网网关](#)。
3. 在新的 VPC 内创建公有子网。
4. 向 VPC 添加新的路由表。
5. 在新路由表中添加一条从 0.0.0.0/0 到互联网网关的路由。
6. 将新的路由表与公有子网关联。
7. 创建弹性 IP 地址。有关更多信息，请参阅[弹性 IP 地址](#)。
8. 创建一个新的 NAT 网关，并将其分配给公有子网和弹性 IP 地址。
9. 在 VPC 中创建私有子网。
10. 向 VPC 默认路由表中添加一条从 0.0.0.0/0 到 NAT 网关的路由。
11. 创建金丝雀。

编写 Python 金丝雀脚本

此脚本将成功通过一次运行，并返回一个字符串。要查看失败的金丝雀示例，请将 fail = False 更改为 fail = True

```
def basic_custom_script():  
    # Insert your code here  
    # Perform multi-step pass/fail check  
    # Log decisions made and results to /tmp  
    # Be sure to wait for all your code paths to complete  
    # before returning control back to Synthetics.  
    # In that way, your canary will not finish and report success  
    # before your code has finished executing  
    fail = False  
    if fail:  
        raise Exception("Failed basicCanary check.")  
    return "Successfully completed basicCanary checks."  
def handler(event, context):  
    return basic_custom_script()
```

将金丝雀文件打包

如果您有多个 .py 文件或脚本具有依赖项，则您可以将它们捆绑到单个 ZIP 格式文件中。如果您使用 syn-python-selenium-1.1 运行时，此 ZIP 格式文件必须将主金丝雀 .py 文件包含在 python 文件夹中，例如 python/my_canary_filename.py。如果您使用 syn-python-selenium-1.1 或之后版本，您可以选择使用其他文件夹，例如 python/myFolder/my_canary_filename.py。

此 ZIP 格式文件应包含所有必要的文件夹和文件，但其他文件不需要位于 python 文件夹中。

请务必将您的金丝雀脚本入口点设置为 my_canary_filename.functionName，以匹配脚本入口点的文件名和函数名称。如果您使用 syn-nodejs-selenium-1.0 运行时，则 functionName

必须是 handler。如果您使用 syn-nodejs-selenium-1.1 或之后版本，该处理程序名称限制不适用，您也可以选择将金丝雀存储在单独的文件夹中，例如 python/myFolder/my_canary_filename.py。如果将其存储在单独的文件夹中，请在脚本入口点中指定该路径，例如 myFolder/my_canary_filename.functionName。

更改现有 Puppeteer 脚本以使用 Synthetics 金丝雀

您可以快速修改用于 Python 和 Selenium 的现有脚本以用作金丝雀。有关 Selenium 的更多信息，请参阅 www.selenium.dev/。

在本示例中，将从以下 Selenium 脚本开始：

```
from selenium import webdriver

def basic_selenium_script():
    browser = webdriver.Chrome()
    browser.get('https://example.com')
    browser.save_screenshot('loaded.png')

basic_selenium_script()
```

转换步骤如下。

转换 Selenium 脚本以用作金丝雀

1. 更改 import 语句以使用 aws_synthetics 模块中的 Selenium：

```
from aws_synthetics.selenium import synthetics_webdriver as webdriver
```

aws_synthetics 模块中的 Selenium 确保金丝雀可以发出指标和日志、生成 HAR 文件及使用其他 CloudWatch Synthetics 功能。

2. 创建一个处理程序函数并调用您的 Selenium 方法。处理程序是脚本的入口点函数。

如果您使用 syn-python-selenium-1.0，处理程序函数必须命名为 handler。如果您使用 syn-python-selenium-1.1 或之后版本，函数可以有任何名称，但必须与脚本中使用的名称相同。此外，如果您使用 syn-python-selenium-1.1 或之后版本，您可以将脚本存储在任何文件夹下，并将该文件夹指定为处理程序名称的一部分。

```
def handler(event, context):
    basic_selenium_script()
```

脚本现已更新为 CloudWatch Synthetics 金丝雀。更新的脚本如下：

```
from aws_synthetics.selenium import synthetics_webdriver as webdriver

def basic_selenium_script():
    browser = webdriver.Chrome()
    browser.get('https://example.com')
    browser.save_screenshot('loaded.png')

def handler(event, context):
    basic_selenium_script()
```

可用于金丝雀脚本的库函数

CloudWatch Synthetics 中包含多个您在编写用作金丝雀的 Node.js 脚本时可以调用的内置类和函数。

某些类和函数适用于 UI 和 API 金丝雀。其他函数仅适用于 UI 金丝雀。UI 金丝雀是使用 `getPage()` 函数，并使用 Puppeteer 作为 Web 驱动程序进行导航和与网页交互的金丝雀。

主题

- 可用于 Node.js 金丝雀脚本的库函数 (p. 245)
- 可用于使用 Selenium 的 Python 金丝雀脚本的库函数 (p. 266)

可用于 Node.js 金丝雀脚本的库函数

本节列出了可用于 Node.js 金丝雀脚本的库函数。

主题

- 适用于所有金丝雀的 Node.js 库类和函数 (p. 245)
- 仅适用于 UI 金丝雀的 Node.js 库类和函数 (p. 257)
- 仅适用于 API 金丝雀的 Node.js 库类和函数 (p. 264)

适用于所有金丝雀的 Node.js 库类和函数

以下用于 Node.js 的 CloudWatch Synthetics 库函数适用于所有金丝雀。

主题

- Synthetics 类 (p. 245)
- SyntheticsConfiguration 类 (p. 246)
- Synthetics Logger (p. 253)
- SyntheticsLogHelper 类 (p. 255)

Synthetics 类

以下适用于所有金丝雀的函数都属于 Synthetics 类。

`addExecutionError(errorMessage, ex);`

`errorMessage` 描述错误，`ex` 指遇到的异常

您可以使用 `addExecutionError` 来设置金丝雀的执行错误。它会在不中断脚本执行的情况下导致金丝雀失败。它也不会影响 `successPercent` 指标。

只有当错误对于指示金丝雀脚本成功或故障并不重要时，才应将错误作为执行错误进行跟踪。

以下是使用 `addExecutionError` 的示例。您将会监控端点的可用性，并在页面加载完成后捕获屏幕截图。由于捕获屏幕截图故障并不会决定端点的可用性，因此您可以捕获在截图时遇到的任何错误，并将它们添加为执行错误。可用性指标仍会指示端点已启动并正在运行，但金丝雀状态会被标记为失败。以下示例代码块将会捕获此类错误并将其添加为执行错误。

```
try {
    await synthetics.takeScreenshot(stepName, "loaded");
} catch(ex) {
    synthetics.addExecutionError('Unable to take screenshot ', ex);
}
```

`getCanaryName();`

返回金丝雀的名称。

`getRuntimeVersion();`

此函数在 `syn-nodejs-puppeteer-3.0` 和更高版本的运行时中可用。其返回金丝雀的 Synthetics 运行时版本。例如，返回值可能为 `syn-nodejs-puppeteer-3.0`。

`getLogLevel();`

检索 Synthetics 库的当前日志级别。可能的值包括：

- 0 – 调试
- 1 – 信息
- 2 – 警告
- 3 – 错误

示例：

```
let logLevel = synthetics.getLogLevel();
```

`setLogLevel();`

设置 Synthetics 库的日志级别。可能的值包括：

- 0 – 调试
- 1 – 信息
- 2 – 警告
- 3 – 错误

示例：

```
synthetics.setLogLevel(0);
```

SyntheticsConfiguration 类

此类仅在 `syn-nodejs-2.1` 或更高版本的运行时中可用。

您可以使用 `SyntheticsConfiguration` 类来配置 Synthetics 库函数的行为。例如，您可以使用此类将 `executeStep()` 函数配置为不捕获屏幕截图。

您可以在全局级别设置 CloudWatch Synthetics 配置，这些配置会应用于金丝雀的所有步骤。您还可以通过传递配置键/值对，在步骤级别覆盖这些配置。

您可以在步骤级别传入选项。有关示例，请参阅 [async executeStep\(stepName, functionToExecute, \[stepConfig\]\); \(p. 257\)](#) 和 [executeHttpStep\(stepName, requestOptions, \[callback\], \[stepConfig\]\) \(p. 264\)](#)。

函数定义：

`setConfig(options)`

`options` 是一个对象，它是一组适用于您的金丝雀的可配置选项。以下各节将介绍 `options` 中的可能字段。

所有金丝雀的 `setConfig(options)`

对于使用 `syn-nodejs-puppeteer-3.2` 或更高版本的金丝雀，`setConfig` 的 (`options`) 可包含以下参数：

- `includeRequestHeaders` (布尔值) – 是否在报告中包含请求标头。默认为 `false`。
- `includeResponseHeaders` (布尔值) – 是否在报告中包含响应标头。默认为 `false`。
- `restrictedHeaders` (数组) – 要忽略的标头值列表 (如果包含标头)。这对请求标头和响应标头均适用。例如, 您可以将 `includeRequestHeaders` 传递为 `true` 并将 `restrictedHeaders` 传递为 `['Authorization']`, 来隐藏您的凭证。
- `includeRequestBody` (布尔值) – 是否在报告中包含请求体。默认为 `false`。
- `includeResponseBody` (布尔值) – 是否在报告中包含响应体。默认为 `false`。

CloudWatch 指标的相关 `setConfig(options)`

对于使用 `syn-nodejs-puppeteer-3.1` 或更高版本的金丝雀, `setConfig` 的 `(options)` 可包含以下布尔参数, 这些参数决定将由金丝雀发布的指标。其中每个选项的默认值均为 `true`。以 `aggregated` 开头的选项决定是否不带 `CanaryName` 维度发出指标。您可以使用该等指标查看所有金丝雀的聚合结果。其他选项决定是否带 `CanaryName` 维度发出指标。您可以使用该等指标查看每个金丝雀的结果。

有关金丝雀发出的 CloudWatch 指标的列表, 请参阅 [金丝雀发布的 CloudWatch 指标 \(p. 286\)](#)。

- `failedCanaryMetric` (布尔值) – 是否发出此金丝雀的 `Failed` 指标 (带 `CanaryName` 维度)。默认为 `true`。
- `failedRequestsMetric` (布尔值) – 是否发出此金丝雀的 `Failed requests` 指标 (带 `CanaryName` 维度)。默认为 `true`。
- `_2xxMetric` (布尔值) – 是否发出此金丝雀的 `2xx` 指标 (带 `CanaryName` 维度)。默认为 `true`。
- `_4xxMetric` (布尔值) – 是否发出此金丝雀的 `4xx` 指标 (带 `CanaryName` 维度)。默认为 `true`。
- `_5xxMetric` (布尔值) – 是否发出此金丝雀的 `5xx` 指标 (带 `CanaryName` 维度)。默认为 `true`。
- `stepDurationMetric` (布尔值) – 是否发出此金丝雀的 `Step duration` 指标 (带 `CanaryName` `StepName` 维度)。默认为 `true`。
- `stepSuccessMetric` (布尔值) – 是否发出此金丝雀的 `Step success` 指标 (带 `CanaryName` `StepName` 维度)。默认为 `true`。
- `aggregatedFailedCanaryMetric` (布尔值) – 是否发出此金丝雀的 `Failed` 指标 (不带 `CanaryName` 维度)。默认为 `true`。
- `aggregatedFailedRequestsMetric` (布尔值) – 是否发出此金丝雀的 `Failed Requests` 指标 (不带 `CanaryName` 维度)。默认为 `true`。
- `aggregated2xxMetric` (布尔值) – 是否发出此金丝雀的 `2xx` 指标 (不带 `CanaryName` 维度)。默认为 `true`。
- `aggregated4xxMetric` (布尔值) – 是否发出此金丝雀的 `4xx` 指标 (不带 `CanaryName` 维度)。默认为 `true`。
- `aggregated5xxMetric` (布尔值) – 是否发出此金丝雀的 `5xx` 指标 (不带 `CanaryName` 维度)。默认为 `true`。
- `visualMonitoringSuccessPercentMetric` (布尔值) – 是否发出此金丝雀的 `visualMonitoringSuccessPercent` 指标。默认为 `true`。
- `visualMonitoringTotalComparisonsMetric` (布尔值) – 是否发出此金丝雀的 `visualMonitoringTotalComparisons` 指标。默认为 `false`。
- `stepsReport` (布尔值) – 是否报告步骤执行摘要。默认为 `true`。
- `includeUrlPassword` (布尔值) – 是否包含 URL 中显示的密码。预设情况下, URL 中显示的密码会在日志和报告中编辑, 以防泄露敏感数据。默认为 `false`。
- `restrictedUrlParameters` (数组) – 要编辑的 URL 路径或查询参数的列表。这适用于出现在日志、报告和错误中的 URL。此参数区分大小写。您可以传递星号 (*) 作为值来编辑所有 URL 路径和查询参数值。默认值为空数组。
- `logRequest` (布尔值) – 是否将每个请求都记录在金丝雀日志中。对于 UI 金丝雀, 这会记录浏览器发送的每个请求。默认为 `true`。

- `logResponse` (布尔值) – 是否将每个响应都记录在金丝雀日志中。对于 UI 金丝雀，这会记录浏览器收到的每个响应。默认为 `true`。
- `logRequestBody` (布尔值) – 是否随请求一起将请求体记录在金丝雀日志中。仅当 `logRequest` 为 `true` 时，此配置才适用。默认为 `false`。
- `logResponseBody` (布尔值) – 是否随响应一起将响应体记录在金丝雀日志中。仅当 `logResponse` 为 `true` 时，此配置才适用。默认为 `false`。
- `logRequestHeaders` (布尔值) – 是否随请求一起将请求标头记录在金丝雀日志中。仅当 `logRequest` 为 `true` 时，此配置才适用。默认为 `false`。

请注意，`includeRequestHeaders` 会在构件中启用标头。

- `logResponseHeaders` (布尔值) – 是否随响应一起将响应标头记录在金丝雀日志中。仅当 `logResponse` 为 `true` 时，此配置才适用。默认为 `false`。

请注意，`includeResponseHeaders` 会在构件中启用标头。

Note

始终会发出每个金丝雀的 `Duration` 和 `SuccessPercent` 指标，这两个指标都会带有和不带 `CanaryName` 指标。

用于启用或禁用指标的方法

`disableAggregatedRequestMetrics()`

禁用金丝雀发出以不带 `CanaryName` 维度形式发出的所有请求指标。

`disableRequestMetrics()`

禁用所有请求指标，包括每个金丝雀的指标和跨所有金丝雀聚合的指标。

`disableStepMetrics()`

禁用所有步骤指标，包括步骤成功指标和步骤持续时间指标。

`enableAggregatedRequestMetrics()`

启用金丝雀发出以不带 `CanaryName` 维度形式发出的所有请求指标。

`enableRequestMetrics()`

启用所有请求指标，包括每个金丝雀的指标和跨所有金丝雀聚合的指标。

`enableStepMetrics()`

启用所有步骤指标，包括步骤成功指标和步骤持续时间指标。

`get2xxMetric()`

返回金丝雀是否发出带 `CanaryName` 维度的 `2xx` 指标。

`get4xxMetric()`

返回金丝雀是否发出带 `CanaryName` 维度的 `4xx` 指标。

`get5xxMetric()`

返回金丝雀是否发出带 `CanaryName` 维度的 `5xx` 指标。

`getAggregated2xxMetric()`

返回金丝雀是否发出不带维度的 2xx 指标。

`getAggregated4xxMetric()`

返回金丝雀是否发出不带维度的 4xx 指标。

`getAggregatedFailedCanaryMetric()`

返回金丝雀是否发出不带维度的 Failed 指标。

`getAggregatedFailedRequestsMetric()`

返回金丝雀是否发出不带维度的 Failed requests 指标。

`getAggregated5xxMetric()`

返回金丝雀是否发出不带维度的 5xx 指标。

`getFailedCanaryMetric()`

返回金丝雀是否发出带 CanaryName 维度的 Failed 指标。

`getFailedRequestsMetric()`

返回金丝雀是否发出带 CanaryName 维度的 Failed requests 指标。

`getStepDurationMetric()`

返回金丝雀是否发出带此金丝雀的 CanaryName 维度的 Duration 指标。

`getStepSuccessMetric()`

返回金丝雀是否发出带此金丝雀的 CanaryName 维度的 StepSuccess 指标。

`with2xxMetric(_2xxMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 CanaryName 维度的 2xx 指标。

`with4xxMetric(_4xxMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 CanaryName 维度的 4xx 指标。

`with5xxMetric(_5xxMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 CanaryName 维度的 5xx 指标。

`withAggregated2xxMetric(aggregated2xxMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出不带维度的 2xx 指标。

`withAggregated4xxMetric(aggregated4xxMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出不带维度的 4xx 指标。

`withAggregated5xxMetric(aggregated5xxMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出不带维度的 5xx 指标。

`withAggregatedFailedCanaryMetric(aggregatedFailedCanaryMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出不带维度的 Failed 指标。

`withAggregatedFailedRequestsMetric(aggregatedFailedRequestsMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出不带维度的 `Failed requests` 指标。

`withFailedCanaryMetric(failedCanaryMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 `CanaryName` 维度的 `Failed` 指标。

`withFailedRequestsMetric(failedRequestsMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 `CanaryName` 维度的 `Failed requests` 指标。

`withStepDurationMetric(stepDurationMetric)`

接受一个布尔参数，该参数指定是否为此 `Canary` 发出带 `CanaryName` 维度的 `Duration` 指标。

`withStepSuccessMetric(stepSuccessMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 `CanaryName` 维度的 `StepSuccess` 指标。

启用或禁用其他功能的方法

`withHarFile()`

接受一个布尔参数，该参数指定是否为此金丝雀创建 HAR 文件。

`withStepsReport()`

接受一个布尔参数，该参数指定是否报告此金丝雀的步骤执行摘要。

`withIncludeUrlPassword()`

接受一个布尔参数，该参数指定是否在日志和报告中包含 URL 中出现的密码。

`withRestrictedUrlParameters()`

接受要编辑的 URL 路径或查询参数的数组。这适用于出现在日志、报告和错误中的 URL。您可以传递星号 (*) 作为值来编辑所有 URL 路径和查询参数值

`withLogRequest()`

接受一个布尔参数，该参数指定是否将每个请求记录在金丝雀的日志中。

`withLogResponse()`

接受一个布尔参数，该参数指定是否将每个响应记录在金丝雀的日志中。

`withLogRequestBody()`

接受一个布尔参数，该参数指定是否将每个请求体记录在金丝雀的日志中。

`withLogResponseBody()`

接受一个布尔参数，该参数指定是否将每个响应体记录在金丝雀的日志中。

`withLogRequestHeaders()`

接受一个布尔参数，该参数指定是否将每个请求标头记录在金丝雀的日志中。

`withLogResponseHeaders()`

接受一个布尔参数，该参数指定是否将每个响应标头记录在金丝雀日志中。

`getHarFile()`

返回金丝雀是否创建 HAR 文件。

`getStepsReport()`

返回金丝雀是否报告步骤执行摘要。

`getIncludeUrlPassword()`

返回金丝雀是否在日志和报告中包含 URL 中出现的密码。

`getRestrictedUrlParameters()`

返回金丝雀是否编辑 URL 路径或查询参数。

`getLogRequest()`

返回金丝雀是否将每个请求记录在金丝雀的日志中。

`getLogResponse()`

返回金丝雀是否将每个响应记录在金丝雀的日志中。

`getLogRequestBody()`

返回金丝雀是否将每个请求体记录在金丝雀的日志中。

`getLogResponseBody()`

返回金丝雀是否将每个响应体记录在金丝雀的日志中。

`getLogRequestHeaders()`

返回金丝雀是否将每个请求标头记录在金丝雀的日志中。

`getLogResponseHeaders()`

返回金丝雀是否将每个响应标头记录在金丝雀的日志中。

所有金丝雀的函数

- `withIncludeRequestHeaders(includeRequestHeaders)`
- `withIncludeResponseHeaders` (包括响应标题)
- `withRestrictedHeaders(restrictedHeaders)`
- `withIncludeRequestBody(includeRequestBody)`
- `withIncludeResponseBody(includeResponseBody)`
- `enableReportingOptions()` – 启用所有报告选项--
includeRequestHeaders、includeResponseHeaders、includeRequestBody 和 includeResponseBody。
- `disableReportingOptions()` – 禁用所有报告选项--
includeRequestHeaders、includeResponseHeaders、includeRequestBody 和 includeResponseBody。

用于 UI 金丝雀的 `setConfig(options)`

对于 UI 金丝雀，`setConfig` 可包含以下布尔参数：

- `continueOnStepFailure` (布尔值) – 是否在步骤失败 (指 `executeStep` 函数) 后继续运行金丝雀脚本。如果任何步骤失败，金丝雀运行仍将被标记为失败。默认为 `false`。
- `harFile` (布尔值) – 是否创建 HAR 文件。默认为 `True`。
- `screenshotOnStepStart` (布尔值) – 是否在开始步骤之前捕获屏幕截图。

- `screenshotOnStepSuccess` (布尔值) – 是否在成功完成步骤后捕获屏幕截图。
- `screenshotOnStepFailure` (布尔值) – 是否在步骤失败后捕获屏幕截图。

启用或禁用屏幕截图的方法

`disableStepScreenshots()`

禁用所有屏幕截图选项 (`screenshotOnStepStart`、`screenshotOnStepSuccess` 和 `screenshotOnStepFailure`)。

`enableStepScreenshots()`

启用所有屏幕截图选项 (`screenshotOnStepStart`、`screenshotOnStepSuccess` 和 `screenshotOnStepFailure`)。预设情况下，这些方法均未启用。

`getScreenshotOnStepFailure()`

返回金丝雀是否在步骤失败后捕获屏幕截图。

`getScreenshotOnStepStart()`

返回金丝雀是否在开始步骤之前捕获屏幕截图。

`getScreenshotOnStepSuccess()`

返回金丝雀是否在成功完成步骤后捕获屏幕截图。

`withScreenshotOnStepStart(screenshotOnStepStart)`

接受一个布尔参数，该参数指示是否在开始步骤之前捕获屏幕截图。

`withScreenshotOnStepSuccess(screenshotOnStepSuccess)`

接受一个布尔参数，该参数指示是否在成功完成步骤后捕获屏幕截图。

`withScreenshotOnStepFailure(screenshotOnStepFailure)`

接受一个布尔参数，该参数指示是否在步骤失败后捕获屏幕截图。

在 UI 金丝雀中的使用情况

首先，导入 Synthetics 依赖关系并获取配置。

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Get Synthetics configuration
const synConfig = synthetics.getConfiguration();
```

然后，通过使用以下选项之一调用 `setConfig` 方法来设置每个选项的配置。

```
// Set configuration values
synConfig.setConfig({
  screenshotOnStepStart: true,
  screenshotOnStepSuccess: false,
  screenshotOnStepFailure: false
});
```

或者

```
synConfig.withScreenshotOnStepStart(false).withScreenshotOnStepSuccess(true).withScreenshotOnStepFailure
```

若要禁用所有屏幕截图，请使用 `disableStepScreenshots()` 函数，如本示例所示。

```
synConfig.disableStepScreenshots();
```

您可以在代码中的任何位置启用和禁用屏幕截图。例如，若要仅禁用一个步骤的屏幕截图，请在运行该步骤之前禁用屏幕截图，然后在该步骤后启用它们。

用于 API 金丝雀的 `setConfig(options)`

对于 API 金丝雀，`setConfig` 可包含以下布尔参数：

- `continueOnHttpStepFailure` (布尔值) - 是否在 HTTP 步骤 (指 `executeHttpStep` 函数) 失败后继续运行金丝雀脚本。如果任何步骤失败，金丝雀运行仍将被标记为失败。默认为 `true`。

可视化监控

可视化监控将在金丝雀运行期间捕获的屏幕截图与在基准金丝雀运行期间捕获的屏幕截图进行比较。如果两个屏幕截图之间的差异超出阈值百分比，则金丝雀失败，您可以在金丝雀运行报告中看到以高亮颜色突出显示的差异区域。运行 `syn-puppeteer-node-3.2` 和更高版本的金丝雀支持可视化监控。运行 Python 和 Selenium 的金丝雀中目前不支持可视化监控。

若要启用可视化监控，请将以下代码行添加到金丝雀脚本中。有关更多信息，请参阅 [SyntheticsConfiguration 类 \(p. 246\)](#)。

```
syntheticsConfiguration.withVisualCompareWithBaseRun(true);
```

将此行添加到脚本后，金丝雀首次成功运行时，它会使用在该运行期间捕获的屏幕截图作为比较基准。在金丝雀的该首次运行之后，您可以使用 CloudWatch 控制台编辑金丝雀以执行以下任一操作：

- 将金丝雀的下一运行设置为新基准。
- 在当前基准屏幕截图上绘制边界，以指定在可视化比较过程中要忽略的屏幕截图区域。
- 删除屏幕截图，使其不用于可视化监控。

有关使用 CloudWatch 控制台编辑金丝雀的更多信息，请参阅 [编辑或删除金丝雀 \(p. 287\)](#)。

可视化监控的其他选项

```
syntheticsConfiguration.withVisualVarianceThresholdPercentage(desiredPercentage)
```

设置可视化对比中屏幕截图差异的可接受百分比。

```
syntheticsConfiguration.withVisualVarianceHighlightHexColor("#fafa00")
```

设置在查看使用可视化监控的金丝雀运行报告时指明差异区域的突出显示颜色。

```
syntheticsConfiguration.withFailCanaryRunOnVisualVariance(failCanary)
```

设置当存在超过阈值的可视化差异时金丝雀是否失败。默认为金丝雀失败。

Synthetics Logger

`SyntheticsLogger` 将日志写入控制台和同一日志级别的本地日志文件。仅当日志级别与所调用的日志函数所需的日志记录级别相同或比其级别低时，此日志文件才会写入到这两个位置。

本地日志文件中的日志记录语句前面加上“DEBUG:”、“INFO:”等，以便与所调用的函数的日志级别相匹配。

如果您希望在与 Synthetics 金丝雀日志记录相同的日志级别运行 Synthetics 库，可以使用 SyntheticsLogger。

如果要创建上载到 S3 结果位置的日志文件，不需要使用 SyntheticsLogger。您可以在 /tmp 文件夹中创建不同的日志文件。在 /tmp 文件夹下创建的任何文件都会作为构件上传到 S3 中的结果位置。

要使用 Synthetics 库日志记录程序，请执行以下操作：

```
const log = require('SyntheticsLogger');
```

有用的函数定义：

```
log.debug(message, ex);
```

参数：**message** 是要记录的消息。**ex** 是要记录的异常（如果有）。

示例：

```
log.debug("Starting step - login.");
```

```
log.error(message, ex);
```

参数：**message** 是要记录的消息。**ex** 是要记录的异常（如果有）。

示例：

```
try {  
  await login();  
} catch (ex) {  
  log.error("Error encountered in step - login.", ex);  
}
```

```
log.info(message, ex);
```

参数：**message** 是要记录的消息。**ex** 是要记录的异常（如果有）。

示例：

```
log.info("Successfully completed step - login.");
```

```
log.log(message, ex);
```

这是 log.info 的一个别名。

参数：**message** 是要记录的消息。**ex** 是要记录的异常（如果有）。

示例：

```
log.log("Successfully completed step - login.");
```

```
log.warn(message, ex);
```

参数：**message** 是要记录的消息。**ex** 是要记录的异常（如果有）。

示例：

```
log.warn("Exception encountered trying to publish CloudWatch Metric.", ex);
```

SyntheticLogHelper 类

SyntheticLogHelper 类在 `syn-nodejs-puppeteer-3.2` 和更高版本的运行时中可用。其已在 CloudWatch Synthetic 库中初始化，并配置了 Synthetic 配置。您可以将它作为依赖项添加到脚本中。此类使您能够清理 URL、标头和错误消息，以编辑敏感信息。

Note

根据 Synthetic 配置设置 `restrictedUrlParameters`，Synthetic 会对其记录的所有 URL 和错误消息进行清理，然后再将它们纳入日志、报告、HAR 文件和金丝雀运行错误中。仅当您在脚本中记录 URL 或错误时，才必须使用 `getSanitizedUrl` 或 `getSanitizedErrorMessage`。除了脚本引发的金丝雀错误之外，Synthetic 不会存储任何金丝雀构件。金丝雀运行构件存储在您的客户账户中。有关更多信息，请参阅[Synthetic 金丝雀的安全注意事项 \(p. 894\)](#)。

`getSanitizedUrl(url, stepConfig = null)`

此函数在 `syn-nodejs-puppeteer-3.2` 和更高版本中可用。它根据配置返回经过清理的 url 字符串。您可以通过设置 `restrictedUrlParameters` 属性选择编辑敏感 URL 参数（如密码和 `access_token`）。预设情况下，会编辑 URL 中的密码。如果需要，您可以通过将 `includeUrlPassword` 设置为 `true` 来启用 URL 密码。

如果传入的 URL 不是有效 URL，此函数会引发一个错误。

参数

- `url` 是一个字符串，也是要清理的 URL。
- `StepConfig`（可选）覆盖此函数的全局 Synthetic 配置。如果 `stepConfig` 未传入，则使用全局配置清理 URL。

示例

此示例使用以下示例 URL：`https://example.com/learn/home?access_token=12345&token_type=Bearer&expires_in=1200`。在此示例中，`access_token` 包含不应记录下来的敏感信息。请注意，Synthetic 服务不会存储任何金丝雀运行构件。日志、屏幕截图和报告等构件都存储在您客户账户的 Amazon S3 存储桶中。

第一步，设置 Synthetic 配置。

```
// Import Synthetic dependency
const synthetic = require('Synthetic');

// Import Synthetic logger for logging url
const log = require('SyntheticLogger');

// Get Synthetic configuration
const synConfig = synthetic.getConfiguration();

// Set restricted parameters
synConfig.setConfig({
  restrictedUrlParameters: ['access_token'];
});
```

接下来，清理并记录 URL

```
// Import SyntheticLogHelper dependency
const syntheticLogHelper = require('SyntheticLogHelper');

const sanitizedUrl = synthetic.getSanitizedUrl('https://example.com/learn/home?access_token=12345&token_type=Bearer&expires_in=1200');
```

这一步会将以下内容记录在您的金丝雀日志中。

```
My example url is: https://example.com/learn/home?
access_token=REDACTED&token_type=Bearer&expires_in=1200
```

您可以通过传入包含 Synthetics 配置选项的可选参数来覆盖 URL 的 Synthetics 配置，如以下示例所示。

```
const urlConfig = {
  restrictedUrlParameters = ['*']
};
const sanitizedUrl = synthetics.getSanitizedUrl('https://example.com/learn/home?
access_token=12345&token_type=Bearer&expires_in=1200', urlConfig);
logger.info('My example url is: ' + sanitizedUrl);
```

上面的示例编辑了所有查询参数，并记录如下：

```
My example url is: https://example.com/learn/home?
access_token=REDACTED&token_type=REDACTED&expires_in=REDACTED
```

getSanitizedErrorMessage

此函数在 `syn-nodejs-puppeteer-3.2` 和更高版本中可用。它根据 Synthetics 配置对所存在的任何 URL 进行清理，返回已清理的错误字符串。您可以选择在调用此函数时通过传递一个可选的 `stepConfig` 参数来覆盖全局 Synthetics 配置。

参数

- **error** 是要清理的错误。它可以是 Error 对象或字符串。
- **StepConfig** (可选) 覆盖此函数的全局 Synthetics 配置。如果 `stepConfig` 未传入，则使用全局配置清理 URL。

示例

此示例使用以下内容：Failed to load url: https://example.com/learn/home?
access_token=12345&token_type=Bearer&expires_in=1200

第一步，设置 Synthetics 配置。

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Import Synthetics logger for logging url
const log = require('SyntheticsLogger');

// Get Synthetics configuration
const synConfig = synthetics.getConfiguration();

// Set restricted parameters
synConfig.setConfig({
  restrictedUrlParameters: ['access_token'];
});
```

接下来，清理并记录错误消息

```
// Import SyntheticsLogHelper dependency
const syntheticsLogHelper = require('SyntheticsLogHelper');
```

```
try {
  // Your code which can throw an error containing url which your script logs
} catch (error) {
  const sanitizedErrorMessage = synthetics.getSanitizedErrorMessage(errorMessage);
  logger.info(sanitizedErrorMessage);
}
```

这一步会将以下内容记录在您的金丝雀日志中。

```
Failed to load url: https://example.com/learn/home?
access_token=REDACTED&token_type=Bearer&expires_in=1200
```

`getSanitizedHeaders(headers, stepConfig=null)`

此函数在 `syn-nodejs-puppeteer-3.2` 和更高版本中可用。它根据 `syntheticsConfiguration` 的 `restrictedHeaders` 属性返回经清理的标头。`restrictedHeaders` 属性中指定的标头是在日志、HAR 文件和报告中编辑的。

参数

- `headers` 是一个包含要清理的标题的对象。
- `StepConfig` (可选) 覆盖此函数的全局 Synthetics 配置。如果 `stepConfig` 未传入，则使用全局配置来清理标头。

仅适用于 UI 金丝雀的 Node.js 库类和函数

以下用于 Node.js 的 CloudWatch Synthetics 库函数仅适用于 UI 金丝雀。

主题

- [Synthetics 类 \(p. 257\)](#)
- [BrokenLinkCheckerReport 类 \(p. 261\)](#)
- [SyntheticsLink 类 \(p. 263\)](#)

Synthetics 类

以下函数位于 Synthetics 类中。

`async addUserAgent(page, userAgentString);`

此函数可将 `userAgentString` 附加到指定页面的 User-Agent 标头。

示例：

```
await synthetics.addUserAgent(page, "MyApp-1.0");
```

结果是页面的 User-Agent 标头被设置为 `browsers-user-agent-header-valueMyApp-1.0`

`async executeStep(stepName, functionToExecute, [stepConfig]);`

执行提供的步骤，并使用开始/通过/失败日志记录、开始/通过/失败屏幕截图，以及通过/失败和持续时间指标对其进行包装。

Note

如果您使用的是 `syn-nodejs-2.1` 或更高版本的运行时，您可以配置是否以及何时捕获屏幕截图。有关更多信息，请参阅 [SyntheticsConfiguration 类 \(p. 246\)](#)。

`executeStep` 函数还执行以下操作：

- 记录步骤开始。
- 获取名为 `<stepName>-starting` 的屏幕截图。
- 启动计时器。
- 执行提供的函数。
- 如果函数正常返回，则计为通过。如果函数引发异常，则计为失败。
- 结束计时器。
- 记录步骤是通过还是失败
- 获取名为 `<stepName>-succeeded` 或 `<stepName>-failed` 的屏幕截图。
- 发出 `stepName SuccessPercent` 指标，100 表示通过，0 表示失败。
- 发出 `stepName Duration` 指标，指标的值基于步骤开始和结束时间。
- 最后，返回 `functionToExecute` 返回的内容或重新抛出 `functionToExecute` 抛出的内容。

如果金丝雀使用 `syn-nodejs-2.0` 或更高版本的运行时，则此函数还将步骤执行摘要添加到金丝雀的报告中。摘要包括有关每个步骤的详细信息，例如开始时间、结束时间、状态 (PASSED/FAILED)、故障原因（如失败）以及在每个步骤执行过程中捕获的屏幕截图。

示例：

```
await synthetics.executeStep('navigateToUrl', async function (timeoutInMillis = 30000) {
    await page.goto(url, {waitUntil: ['load', 'networkidle0'], timeout:
    timeoutInMillis});});
```

响应：

返回 `functionToExecute` 返回的内容。

syn-nodejs-2.2 的更新

从 `syn-nodejs-2.2` 开始，您可以选择传递步骤配置来在步骤级别覆盖 CloudWatch Synthetics 配置。有关可以传递给 `executeStep` 的选项列表，请参阅 [SyntheticsConfiguration 类 \(p. 246\)](#)。

在以下示例中，用 `true` 覆盖 `continueOnStepFailure` 的默认 `false` 配置，并指定捕获屏幕截图的时间。

```
var stepConfig = {
    'continueOnStepFailure': true,
    'screenshotOnStepStart': false,
    'screenshotOnStepSuccess': true,
    'screenshotOnStepFailure': false
}

await executeStep('Navigate to amazon', async function (timeoutInMillis = 30000) {
    await page.goto(url, {waitUntil: ['load', 'networkidle0'], timeout:
    timeoutInMillis});
}, stepConfig);
```

[getDefaultLaunchOptions\(\);](#)

`getDefaultLaunchOptions()` 函数返回 CloudWatch Synthetics 使用的浏览器启动选项。有关更多信息，请参阅 [puppeteer.launch\(\[options\]\)](#)

```
// This function returns default launch options used by Synthetics.
const defaultOptions = await synthetics.getDefaultLaunchOptions();
```

`getPage();`

返回当前打开的页面作为 Puppeteer 对象。有关更多信息，请参阅 [Puppeteer API v1.14.0](#)。

示例：

```
let page = synthetics.getPage();
```

响应：

在当前浏览器会话中打开的页面（Puppeteer 对象）。

`getRequestResponseLogHelper();`

Important

在使用 `syn-nodejs-puppeteer-3.2` 或更高版本运行时的金丝雀中，此函数将与 `RequestResponseLogHelper` 类一起弃用。对此函数的任何使用都会导致在金丝雀日志中出现警告。将在未来的运行时版本中删除此函数。如果您正在使用此函数，请改为使用 [RequestResponseLogHelper 类 \(p. 260\)](#)。

将此函数用作生成器模式，用于调整请求和响应日志记录标记。

示例：

```
synthetics.setRequestResponseLogHelper(getRequestResponseLogHelper().withLogRequestHeaders(false));;
```

响应：

```
{RequestResponseLogHelper}
```

`launch(options)`

此函数的选项仅在 `syn-nodejs-2.1` 版本或更高版本的运行时中可用。

此函数仅用于 UI 金丝雀。它的作用是关闭现有浏览器并启动一个新的浏览器。

Note

在开始运行脚本之前，CloudWatch Synthetics 始终启动浏览器。除非您想使用自定义选项启动新浏览器，否则不需要调用 `launch()`。

(options) 是在浏览器上设置的一组可配置的选项。有关更多信息，请参阅 [puppeteer.launch\(\[options\]\)](#)。

如果您在不设置选项的情况下调用此函数，Synthetics 会以默认参数、`executablePath` 和 `defaultViewport` 启动浏览器。CloudWatch Synthetics 中的默认视区是 1920 x 1080。

您可以覆盖 CloudWatch Synthetics 使用的启动参数，并在启动浏览器时传递其他参数。例如，以下代码段以默认参数和默认可执行文件路径启动浏览器，但视区为 800 x 600。

```
await synthetics.launch({
  defaultViewport: {
    "deviceScaleFactor": 1,
    "width": 800,
    "height": 600
  }});
```

以下示例代码向 CloudWatch Synthetics 启动参数中添加了一个新的 `ignoreHTTPSErrors` 参数。

```
await synthetics.launch({
    ignoreHTTPSErrors: true
});
```

您可以通过向 CloudWatch Synthetics 启动参数中的参数添加一个 `--disable-web-security` 标志来禁用 Web 安全：

```
// This function adds the --disable-web-security flag to the launch parameters
const defaultOptions = await synthetics.getDefaultLaunchOptions();
const launchArgs = [...defaultOptions.args, '--disable-web-security'];
await synthetics.launch({
    args: launchArgs
});
```

RequestResponseLogHelper 类

Important

在使用 `syn-nodejs-puppeteer-3.2` 或更高版本运行时的金丝雀中，此类将被弃用。对此类的任何使用都会导致在金丝雀日志中出现警告。将在未来的运行时版本中删除此函数。如果您正在使用此函数，请改为使用 [RequestResponseLogHelper 类](#) (p. 260)。

处理精细配置，以及创建请求和响应负载的字符串表示。

```
class RequestResponseLogHelper {

    constructor () {
        this.request = {url: true, resourceType: false, method: false, headers: false,
            postData: false};
        this.response = {status: true, statusText: true, url: true, remoteAddress: false,
            headers: false};
    }

    withLogRequestUrl(logRequestUrl);

    withLogRequestResourceType(logRequestResourceType);

    withLogRequestMethod(logRequestMethod);

    withLogRequestHeaders(logRequestHeaders);

    withLogRequestPostData(logRequestPostData);

    withLogResponseStatus(logResponseStatus);

    withLogResponseStatusText(logResponseStatusText);

    withLogResponseUrl(logResponseUrl);

    withLogResponseRemoteAddress(logResponseRemoteAddress);

    withLogResponseHeaders(logResponseHeaders);
```

示例：

```
synthetics.setRequestResponseLogHelper(getRequestResponseLogHelper()
    .withLogRequestPostData(true)
    .withLogRequestHeaders(true)
    .withLogResponseHeaders(true));
```


响应:

```
{RequestResponseLogHelper}
```

[setRequestResponseLogHelper\(\)](#);

Important

在使用 `syn-nodejs-puppeteer-3.2` 或更高版本运行时的金丝雀中，此函数将与 `RequestResponseLogHelper` 类一起弃用。对此函数的任何使用都会导致在金丝雀日志中出现警告。将在未来的运行时版本中删除此函数。如果您正在使用此函数，请改为使用 [RequestResponseLogHelper 类 \(p. 260\)](#)。

将此函数用作生成器模式，用于设置请求和响应日志记录标记。

示例：

```
synthetics.setRequestResponseLogHelper().withLogRequestHeaders(true).withLogResponseHeaders(true);
```

响应:

```
{RequestResponseLogHelper}
```

[async takeScreenshot\(name, suffix\)](#);

捕获当前页面的屏幕截图 (.PNG) 并命名和添加后缀 (可选)。

示例：

```
await synthetics.takeScreenshot("navigateToUrl", "loaded")
```

在此示例中，捕获了名为 `01-navigateToUrl-loaded.png` 的屏幕截图并将其上载到金丝雀的 S3 存储桶。

您可以传递 `stepName` 作为第一个参数，捕获特定金丝雀步骤的屏幕截图。屏幕截图会链接到报告中的金丝雀步骤，以帮助您在调试时跟踪每个步骤。

CloudWatch Synthetics 金丝雀会在开始步骤之前 (`executeStep` 函数) 和步骤完成后自动捕获屏幕截图 (除非您将金丝雀配置为禁用屏幕截图)。您可以通过将步骤名称传入 `takeScreenshot` 函数来捕获多个屏幕截图。

在以下示例中，捕获屏幕截图并将 `signupForm` 作为 `stepName` 的值。屏幕截图将被命名为 `02-signupForm-address`，并链接到金丝雀报告中名为 `signupForm` 的步骤。

```
await synthetics.takeScreenshot('signupForm', 'address')
```

[BrokenLinkCheckerReport 类](#)

此类提供了添加 Synthetics 链接的方法。仅在使用 `syn-nodejs-2.0-beta` 或更高版本运行时的金丝雀上支持此类。

若要使用 `BrokenLinkCheckerReport`，脚本中需包括以下行：

```
const BrokenLinkCheckerReport = require('BrokenLinkCheckerReport');
```

```
const brokenLinkCheckerReport = new BrokenLinkCheckerReport();
```

有用的函数定义：

`addLink(syntheticsLink, isBroken)`

syntheticsLink 是一个 `SyntheticsLink` 对象，表示链接。此函数根据状态代码添加链接。预设情况下，如果状态代码不可用或状态代码为 400 或以上，则其会将链接视为无效。您可以通过传入可选参数 `isBrokenLink`（带有值 `true` 或 `false`）来覆盖此默认行为。

此函数无返回值。

`getLinks()`

此函数返回一组 `SyntheticsLink` 对象，这些对象包含在无效链接检查器报告。

`getTotalBrokenLinks()`

此函数返回一个表示无效链接总数的数字。

`getTotalLinksChecked()`

此函数返回一个表示包含在报告中的链接总数的数字。

如何使用 `BrokenLinkCheckerReport`

以下金丝雀脚本代码段展示了导航到链接并将其添加到无效链接检查器报告的示例。

1. 导入 `SyntheticsLink`、`BrokenLinkCheckerReport` 和 `Synthetics`。

```
const BrokenLinkCheckerReport = require('BrokenLinkCheckerReport');
const SyntheticsLink = require('SyntheticsLink');

// Synthetics dependency
const synthetics = require('Synthetics');
```

2. 若要向报告添加链接，请创建 `BrokenLinkCheckerReport` 实例。

```
let brokenLinkCheckerReport = new BrokenLinkCheckerReport();
```

3. 导航到 URL 并将其添加到无效链接检查器报告中。

```
let url = "https://amazon.com";

let syntheticsLink = new SyntheticsLink(url);

// Navigate to the url.
let page = await synthetics.getPage();

// Create a new instance of Synthetics Link
let link = new SyntheticsLink(url)

try {
  const response = await page.goto(url, {waitUntil: 'domcontentloaded', timeout:
    30000});
} catch (ex) {
  // Add failure reason if navigation fails.
  link.withFailureReason(ex);
}

if (response) {
  // Capture screenshot of destination page
```

```
let screenshotResult = await synthetics.takeScreenshot('amazon-home', 'loaded');

// Add screenshot result to synthetics link
link.addScreenshotResult(screenshotResult);

// Add status code and status description to the link
link.withStatusCode(response.status()).withStatusText(response.statusText())
}

// Add link to broken link checker report.
brokenLinkCheckerReport.addLink(link);
```

4. 将报告添加到 Synthetics。这样，金丝雀每次运行都会在 S3 存储桶中创建一个名为 `BrokenLinkCheckerReport.json` 的 JSON 文件。您可以在控制台中查看金丝雀每次运行的链接报告以及屏幕截图、日志和 HAR 文件。

```
await synthetics.addReport(brokenLinkCheckerReport);
```

SyntheticsLink 类

此类提供了包装信息的方法。仅在使用 `syn-nodejs-2.0-beta` 或更高版本运行时的金丝雀上支持此类。

若要使用 `SyntheticsLink`，脚本中需包括以下行：

```
const SyntheticsLink = require('SyntheticsLink');
const syntheticsLink = new SyntheticsLink("https://www.amazon.com");
```

此函数返回 `syntheticsLinkObject`

有用的函数定义：

`withUrl(url)`

`url` 是一个 URL 字符串。此函数返回 `syntheticsLinkObject`

`withText(text)`

`text` 是一个表示锚文本的字符串。此函数返回 `syntheticsLinkObject`。它会添加与链接相对应的锚文本。

`withParentUrl(parentUrl)`

`parentUrl` 是一个表示父（源页面）URL 的字符串。此函数返回 `syntheticsLinkObject`

`withStatusCode(statusCode)`

`statusCode` 是一个表示状态代码的字符串。此函数返回 `syntheticsLinkObject`

`withFailureReason(failureReason)`

`failureReason` 是一个表示故障原因的字符串。此函数返回 `syntheticsLinkObject`

`addScreenshotResult(screenshotResult)`

`screenshotResult` 是一个对象。它是 `Synthetics` 函数 `takeScreenshot` 返回的 `ScreenshotResult` 的实例。此对象包括以下属性：

- `fileName` – 表示 `screenshotFileName` 的字符串
- `pageUrl` (可选)

- `error` (可选)

仅适用于 API 金丝雀的 Node.js 库类和函数

以下用于 Node.js 的 CloudWatch Synthetics 库函数仅适用于 API 金丝雀。

主题

- [executeHttpStep\(stepName, requestOptions, \[callback\], \[stepConfig\]\)](#) (p. 264)

executeHttpStep(stepName, requestOptions, [callback], [stepConfig])

作为步骤执行提供的 HTTP 请求，并发布 `SuccessPercent` (通过/失败) 和 `Duration` 指标。

`executeHttpStep` 在后台使用 HTTP 或 HTTPS 本机函数，具体取决于请求中指定的协议。

此函数还将步骤执行摘要添加到金丝雀的报告中。摘要中包括有关每个 HTTP 请求的详细信息，如下所示：

- 开始时间
- 结束时间
- 状态 (PASSED/FAILED)
- 故障原因 (如失败)
- HTTP 调用详细信息，如请求/响应标头、请求/响应体、状态代码、状态消息和性能计时。

参数

`stepName`(**String**)

指定步骤的名称。此名称也用于发布此步骤的 CloudWatch 指标。

`requestOptions`(**Object or String**)

此参数的值可以是 URL、URL 字符串或对象。如果为对象，则必须是用以发出 HTTP 请求的一组可配置选项。其支持 Node.js 文档中 [http.request\(options\[, callback\]\)](#) 中的所有选项。

除了这些 Node.js 选项之外，`requestOptions` 支持附加参数 `body`。您可以使用 `body` 参数将数据作为请求体传递。

`callback`(**response**)

(可选) 此函数是与 HTTP 响应一起调用的用户函数。响应的类型为 [Class: http.IncomingMessage](#)。

`stepConfig`(**object**)

(可选) 使用此参数，可以用此步骤的不同配置覆盖全局 synthetics 配置。

executeHttpStep 的使用示例

以下一系列示例相互依存，以说明此选项的各种用途。

第一个示例展示配置请求参数。您可以将 URL 作为 `requestOptions` 传递：

```
let requestOptions = 'https://www.amazon.com';
```

或者，您也可以传递一组选项：

```
let requestOptions = {
```

```
'hostname': 'myproductsEndpoint.com',  
'method': 'GET',  
'path': '/test/product/validProductName',  
'port': 443,  
'protocol': 'https:'  
};
```

在下一个示例中，创建一个接受响应的回调函数。预设情况下，如果您未指定 callback，CloudWatch Synthetics 会验证状态是否介于 200 和 299 之间（含）。

```
// Handle validation for positive scenario  
const callback = async function(res) {  
  return new Promise((resolve, reject) => {  
    if (res.statusCode < 200 || res.statusCode > 299) {  
      throw res.statusCode + ' ' + res.statusMessage;  
    }  
  
    let responseBody = '';  
    res.on('data', (d) => {  
      responseBody += d;  
    });  
  
    res.on('end', () => {  
      // Add validation on 'responseBody' here if required. For ex, your status  
      // code is 200 but data might be empty  
      resolve();  
    });  
  });  
};
```

下一个示例展示为此步骤创建覆盖全局 CloudWatch Synthetics 配置的配置。在本示例中，步骤配置允许报告中包含请求标头、响应标头、请求体（发布数据）和响应体，并限制了“X-Amz-Security-Token”和“Authorization”标头值。预设情况下，出于安全原因，报告中不包括这些值。如果您选择将它们包含在报告中，则这些数据仅存储在 S3 存储桶中。

```
// By default headers, post data, and response body are not included in the report for  
// security reasons.  
// Change the configuration at global level or add as step configuration for individual  
// steps  
let stepConfig = {  
  includeRequestHeaders: true,  
  includeResponseHeaders: true,  
  restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted header  
  // values do not appear in report generated.  
  includeRequestBody: true,  
  includeResponseBody: true  
};
```

最后一个例子展示将请求传递给 executeHttpStep 并对步骤命名。

```
await synthetics.executeHttpStep('Verify GET products API', requestOptions, callback,  
  stepConfig);
```

在这组示例中，CloudWatch Synthetics 将每个步骤的详细信息添加到报告中，并使用 stepName 生成每个步骤的指标。

您将看到 Verify GET products API 步骤的 successPercent 和 duration 指标。您可以通过监控 API 调用步骤的指标来监控 API 性能。

有关使用这些函数的完整脚本示例，请参阅 [多步骤 API 金丝雀](#) (p. 280)。

可用于使用 Selenium 的 Python 金丝雀脚本的库函数

本节列出了可用于 Python 金丝雀脚本的 Selenium 库函数。

主题

- [适用于所有金丝雀的 Python 和 Selenium 库类和函数 \(p. 266\)](#)
- [仅适用于 UI 金丝雀的 Python 和 Selenium 库类和函数 \(p. 271\)](#)

适用于所有金丝雀的 Python 和 Selenium 库类和函数

以下用于 Python 的 CloudWatch Synthetics 库函数适用于所有金丝雀。

主题

- [SyntheticsConfiguration 类 \(p. 266\)](#)
- [SyntheticsLogger 类 \(p. 269\)](#)

SyntheticsConfiguration 类

您可以使用 SyntheticsConfiguration 类来配置 Synthetics 库函数的行为。例如，您可以使用此类将 `executeStep()` 函数配置为不捕获屏幕截图。

您可以在全局级别设置 CloudWatch Synthetics 配置。

函数定义：

`set_config(options)`

```
from aws_synthetics.common import synthetics_configuration
```

`options` 是一个对象，它是一组适用于您的金丝雀的可配置选项。以下各节将介绍 `options` 中的可能字段。

- `screenshot_on_step_start` (布尔值) – 是否在开始步骤之前捕获屏幕截图。
- `screenshot_on_step_success` (布尔值) – 是否在成功完成步骤后捕获屏幕截图。
- `screenshot_on_step_failure` (布尔值) – 是否在步骤失败后捕获屏幕截图。

`with_screenshot_on_step_start(screenshot_on_step_start)`

接受一个布尔参数，该参数指示是否在开始步骤之前捕获屏幕截图。

`with_screenshot_on_step_success(screenshot_on_step_success)`

接受一个布尔参数，该参数指示是否在成功完成步骤后捕获屏幕截图。

`with_screenshot_on_step_failure(screenshot_on_step_failure)`

接受一个布尔参数，该参数指示是否在步骤失败后捕获屏幕截图。

`get_screenshot_on_step_start()`

返回是否在开始步骤之前捕获屏幕截图。

`get_screenshot_on_step_success()`

返回是否在成功完成步骤后捕获屏幕截图。

`get_screenshot_on_step_failure()`

返回是否在步骤失败后捕获屏幕截图。

`disable_step_screenshots()`

禁用所有屏幕截图选项 (`get_screenshot_on_step_start`、`get_screenshot_on_step_success` 和 `get_screenshot_on_step_failure`)。

`enable_step_screenshots()`

启用所有屏幕截图选项 (`get_screenshot_on_step_start`、`get_screenshot_on_step_success` 和 `get_screenshot_on_step_failure`)。预设情况下，这些方法均未启用。

CloudWatch 指标的相关 `setConfig(options)`

对于使用 `syn-python-selenium-1.1` 或更高版本的金丝雀，`setConfig` 的 (options) 可包含以下布尔参数，这些参数决定将由金丝雀发布的指标。其中每个选项的默认值均为 `true`。以 `aggregated` 开头的选项决定是否不带 `CanaryName` 维度发出指标。您可以使用该等指标查看所有金丝雀的聚合结果。其他选项决定是否带 `CanaryName` 维度发出指标。您可以使用该等指标查看每个金丝雀的结果。

有关金丝雀发出的 CloudWatch 指标的列表，请参阅 [金丝雀发布的 CloudWatch 指标 \(p. 286\)](#)。

- `failed_canary_metric` (布尔值) – 是否发出此金丝雀的 Failed 指标 (带 `CanaryName` 维度)。默认为 `true`。
- `failed_requests_metric` (布尔值) – 是否发出此金丝雀的 Failed requests 指标 (带 `CanaryName` 维度)。默认为 `true`。
- `2xx_metric` (布尔值) – 是否发出此金丝雀的 2xx 指标 (带 `CanaryName` 维度)。默认为 `true`。
- `4xx_metric` (布尔值) – 是否发出此金丝雀的 4xx 指标 (带 `CanaryName` 维度)。默认为 `true`。
- `5xx_metric` (布尔值) – 是否发出此金丝雀的 5xx 指标 (带 `CanaryName` 维度)。默认为 `true`。
- `step_duration_metric` (布尔值) – 是否发出此金丝雀的 Step duration 指标 (带 `CanaryName` `StepName` 维度)。默认为 `true`。
- `step_success_metric` (布尔值) – 是否发出此金丝雀的 Step success 指标 (带 `CanaryName` `StepName` 维度)。默认为 `true`。
- `aggregated_failed_canary_metric` (布尔值) – 是否发出此金丝雀的 Failed 指标 (不带 `CanaryName` 维度)。默认为 `true`。
- `aggregated_failed_requests_metric` (布尔值) – 是否发出此金丝雀的 Failed Requests 指标 (不带 `CanaryName` 维度)。默认为 `true`。
- `aggregated_2xx_metric` (布尔值) – 是否发出此金丝雀的 2xx 指标 (不带 `CanaryName` 维度)。默认为 `true`。
- `aggregated_4xx_metric` (布尔值) – 是否发出此金丝雀的 4xx 指标 (不带 `CanaryName` 维度)。默认为 `true`。
- `aggregated_5xx_metric` (布尔值) – 是否发出此金丝雀的 5xx 指标 (不带 `CanaryName` 维度)。默认为 `true`。

`with_2xx_metric(2xx_metric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 `CanaryName` 维度的 2xx 指标。

`with_4xx_metric(4xx_metric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 `CanaryName` 维度的 4xx 指标。

`with_5xx_metric(5xx_metric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 `CanaryName` 维度的 5xx 指标。

`withAggregated2xxMetric(aggregated2xxMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出不带维度的 2xx 指标。

`withAggregated4xxMetric(aggregated4xxMetric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出不带维度的 4xx 指标。

`with_aggregated_5xx_metric(aggregated_5xx_metric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出不带维度的 5xx 指标。

`with_aggregated_failed_canary_metric(aggregated_failed_canary_metric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出不带维度的 Failed 指标。

`with_aggregated_failed_requests_metric(aggregated_failed_requests_metric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出不带维度的 Failed requests 指标。

`with_failed_canary_metric(failed_canary_metric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 CanaryName 维度的 Failed 指标。

`with_failed_requests_metric(failed_requests_metric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 CanaryName 维度的 Failed requests 指标。

`with_step_duration_metric(step_duration_metric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 CanaryName 维度的 Duration 指标。

`with_step_success_metric(step_success_metric)`

接受一个布尔参数，该参数指定是否为此金丝雀发出带 CanaryName 维度的 StepSuccess 指标。

用于启用或禁用指标的方法

`disable_aggregated_request_metrics()`

禁用金丝雀发出以不带 CanaryName 维度形式发出的所有请求指标。

`disable_request_metrics()`

禁用所有请求指标，包括每个金丝雀的指标和跨所有金丝雀聚合的指标。

`disable_step_metrics()`

禁用所有步骤指标，包括步骤成功指标和步骤持续时间指标。

`enable_aggregated_request_metrics()`

启用金丝雀发出以不带 CanaryName 维度形式发出的所有请求指标。

`enable_request_metrics()`

启用所有请求指标，包括每个金丝雀的指标和跨所有金丝雀聚合的指标。

`enable_step_metrics()`

启用所有步骤指标，包括步骤成功指标和步骤持续时间指标。

在 UI 金丝雀中的使用情况

首先，导入 Synthetics 依赖关系并获取配置。然后，通过使用以下选项之一调用 `setConfig` 方法来设置每个选项的配置。

```
from aws_synthetics.common import synthetics_configuration

synthetics_configuration.set_config(
    {
        "screenshot_on_step_start": False,
        "screenshot_on_step_success": False,
        "screenshot_on_step_failure": True
    }
)

or
```

或者

```
synthetics_configuration.with_screenshot_on_step_start(False).with_screenshot_on_step_success(False).wi
```

若要禁用所有屏幕截图，请使用 `disableStepScreenshots()` 函数，如本示例所示。

```
synthetics_configuration.disable_step_screenshots()
```

您可以在代码中的任何位置启用和禁用屏幕截图。例如，若要仅禁用一个步骤的屏幕截图，请在运行该步骤之前禁用屏幕截图，然后在该步骤后启用它们。

用于 UI 金丝雀的 `set_config(options)`

从 `syn-python-selenium-1.1` 开始，对于 UI 金丝雀，`set_config` 可包含以下布尔参数：

- `continue_on_step_failure` (布尔值) – 是否在步骤失败 (指 `executeStep` 函数) 后继续运行金丝雀脚本。如果任何步骤失败，金丝雀运行仍将被标记为失败。默认为 `false`。

SyntheticsLogger 类

`synthetics_logger` 将日志写入控制台和同一日志级别的本地日志文件。仅当日志级别与所调用的日志函数所需的日志记录级别相同或比其级别低时，此日志文件才会写入到这两个位置。

本地日志文件中的日志记录语句前面加上“DEBUG:”、“INFO:”等，以便与所调用的函数的日志级别相匹配。

如果要创建上载到 Amazon S3 结果位置的日志文件，不需要使用 `synthetics_logger`。您可以在 `/tmp` 文件夹中创建不同的日志文件。在 `/tmp` 文件夹下创建的任何文件都会作为构件上载到 S3 存储桶中的结果位置。

若要使用 `synthetics_logger`：

```
from aws_synthetics.common import synthetics_logger
```

有用的函数定义：

获取日志级别：

```
log_level = synthetics_logger.get_level()
```

设置日志级别：

```
synthetics_logger.set_level()
```

使用指定级别记录消息。该级别可以是 DEBUG、INFO、WARN，或 ERROR，如以下语法示例所示：

```
synthetics_logger.debug(message, *args, **kwargs)
```

```
synthetics_logger.info(message, *args, **kwargs)
```

```
synthetics_logger.log(message, *args, **kwargs)
```

```
synthetics_logger.warn(message, *args, **kwargs)
```

```
synthetics_logger.error(message, *args, **kwargs)
```

有关调试参数的信息，请参阅标准 Python 文档中的 [logging.debug](#)

在这些日志记录函数中，message 为消息格式的字符串。args 为合并到使用字符串格式运算符的 msg 中的参数。

kwargs 中有三个关键字参数：

- exc_info – 如果未评估为 false，则会将异常信息添加到日志记录消息中。
- stack_info – 默认值为 false。如果为 true，则会将堆栈信息添加到日志记录消息中，包括实际的日志记录调用。
- extra – 第三个可选关键字参数，您可以使用该参数将用于填充 LogRecord（为具有用户定义属性的日志记录事件而创建）的 __dict__ 传入字典。

示例：

使用 DEBUG 级别记录消息：

```
synthetics_logger.debug('Starting step - login.')
```

使用 INFO 级别记录消息。logger.log 是 logger.info 的同义词：

```
synthetics_logger.info('Successfully completed step - login.')
```

或

```
synthetics_logger.log('Successfully completed step - login.')
```

使用 WARN 级别记录消息：

```
synthetics_logger.warn('Warning encountered trying to publish %s', 'CloudWatch Metric')
```

使用 ERROR 级别记录消息：

```
synthetics_logger.error('Error encountered trying to publish %s', 'CloudWatch Metric')
```

记录异常：

```
synthetics_logger.exception(message, *args, **kwargs)
```

使用 ERROR 级别记录消息。异常信息将会添加到日志记录消息中。您应该只从异常处理程序调用此函数。

有关异常参数的信息，请参阅标准 Python 文档中的 [logging.exception](#)

message 为消息格式的字符串。args 为合并到使用字符串格式运算符的 msg 中的参数。

kwargs 中有三个关键字参数：

- exc_info – 如果未评估为 false，则会将异常信息添加到日志记录消息中。
- stack_info – 默认值为 false。如果为 true，则会将堆栈信息添加到日志记录消息中，包括实际的日志记录调用。
- extra – 第三个可选关键字参数，您可以使用该参数将用于填充 LogRecord（为具有用户定义属性的日志记录事件而创建）的 __dict__ 传入字典。

示例：

```
synthetics_logger.exception('Error encountered trying to publish %s', 'CloudWatch Metric')
```

仅适用于 UI 金丝雀的 Python 和 Selenium 库类和函数

以下用于 Python 的 CloudWatch Synthetics Selenium 库函数仅适用于 UI 金丝雀。

主题

- [SyntheticsBrowser 类 \(p. 271\)](#)
- [SyntheticsWebDriver 类 \(p. 271\)](#)

SyntheticsBrowser 类

当您通过调用 synthetics_webdriver.Chrome() 创建浏览器实例时，返回的浏览器实例类型为 SyntheticsBrowser。SyntheticsBrowser 类控制 ChromeDriver，并使金丝雀脚本能够驱动浏览器，从而允许 Selenium WebDriver 使用 Synthetics。

除了标准的 Selenium 方法之外，它还提供了以下方法。

[set_viewport_size\(width, height\)](#)

设置浏览器的视区。示例：

```
browser.set_viewport_size(1920, 1080)
```

[save_screenshot\(filename, suffix\)](#)

将屏幕截图保存到 /tmp 目录。屏幕截图将上载到 S3 存储桶中的金丝雀构件文件夹。

filename 是屏幕截图的文件名，后缀是用于命名屏幕截图的可选字符串。

示例：

```
browser.save_screenshot('loaded.png', 'page1')
```

SyntheticsWebDriver 类

要使用此类，请在脚本中使用以下内容：

```
from aws_synthetics.selenium import synthetics_webdriver
```

`add_execution_error(errorMessage, ex);`

`errorMessage` 描述错误，`ex` 指遇到的异常

您可以使用 `add_execution_error` 来设置金丝雀的执行错误。它会在不中断脚本执行的情况下导致金丝雀失败。它也不会影响 `successPercent` 指标。

只有当错误对于指示金丝雀脚本成功或故障并不重要时，才应将错误作为执行错误进行跟踪。

以下是使用 `add_execution_error` 的示例。您将会监控端点的可用性，并在页面加载完成后捕获屏幕截图。由于捕获屏幕截图故障并不会决定端点的可用性，因此您可以捕获在截图时遇到的任何错误，并将它们添加为执行错误。可用性指标仍会指示端点已启动并正在运行，但金丝雀状态会被标记为失败。以下示例代码块将会捕获此类错误并将其添加为执行错误。

```
try:
    browser.save_screenshot("loaded.png")
except Exception as ex:
    self.add_execution_error("Unable to take screenshot", ex)
```

`add_user_agent(user_agent_str)`

将 `user_agent_str` 的值附加到浏览器的用户代理标头。您必须在创建浏览器实例之前分配 `user_agent_str`。

示例：

```
synthetics_webdriver.add_user_agent('MyApp-1.0')
```

`execute_step(step_name, function_to_execute)`

处理一个函数。它还执行以下操作：

- 记录步骤开始。
- 获取名为 `<stepName>-starting` 的屏幕截图。
- 启动计时器。
- 执行提供的函数。
- 如果函数正常返回，则计为通过。如果函数引发异常，则计为失败。
- 结束计时器。
- 记录步骤是通过还是失败
- 获取名为 `<stepName>-succeeded` 或 `<stepName>-failed` 的屏幕截图。
- 发出 `stepName SuccessPercent` 指标，100 表示通过，0 表示失败。
- 发出 `stepName Duration` 指标，指标的值基于步骤开始和结束时间。
- 最后，返回 `functionToExecute` 返回的内容或重新抛出 `functionToExecute` 抛出的内容。

示例：

```
def custom_actions():
    #verify contains
    browser.find_element_by_xpath("//*[@id='id_1'][contains(text(),'login')]")
    #click a button
    browser.find_element_by_xpath('//*[@id="submit"]/a').click()
```

```
await synthetics_webdriver.execute_step("verify_click", custom_actions)
```

Chrome()

启动 Chromium 浏览器的实例并返回创建的浏览器实例。

示例：

```
browser = synthetics_webdriver.Chrome()
browser.get("https://example.com/")
```

要以隐身模式启动浏览器，请遵循以下说明：

```
add_argument('--incognito')
```

要添加代理设置，请遵循以下说明：

```
add_argument('--proxy-server=%s' % PROXY)
```

示例：

```
from selenium.webdriver.chrome.options import Options
chrome_options = Options()
chrome_options.add_argument("--incognito")
browser = syn_webdriver.Chrome(chrome_options=chrome_options)
```

使用 cron 安排金丝雀运行

当您安排金丝雀时，使用 cron 表达式可以让您灵活地安排计划。Cron 表达式包含五或六个按下表所列顺序排列的字段。这些字段采用空格分隔。语法根据您创建金丝雀使用的是 CloudWatch 控制台还是 Amazon CLI 或 Amazon SDK 而有所不同。若您使用控制台，则只能指定前五个字段。若使用 Amazon CLI 或 Amazon SDK，则可以指定所有六个字段，而且必须为 Year 字段指定 *。

字段	允许的值	允许的特殊字符
分钟	0-59	, - * /
小时	0-23	, - * /
日期	1-31	, - * ? / L W
月份	1-12 或 JAN-DEC	, - * /
星期几	1-7 或 SUN-SAT	, - * ? L #
年份	*	

特殊字符

- , (逗号) 在字段的表达式中包含多个值。例如，在“Month (月份)”字段中，JAN、FEB 和 MAR 将包含 January、February 和 March。
- - (破折号) 特殊字符用于指定范围。在“日”字段中，1-15 将包含指定月份的 1 - 15 日。
- * (星号) 特殊字符包含该字段中的所有值。在“Hours (小时)”字段中，* 包括每个小时。您不能在一个表达式的“Day-of-month (日期)”和“Day-of-week (星期几)”字段中同时使用 *。如果您在一个中使用它，则必须在另一个中使用？。

- / (正斜杠) 用于指定增量。在“Minutes (分钟)”字段中，您可以输入 1/10 以指定从一个小时的第一分钟开始的每个第十分钟 (例如，第 11 分钟、第 21 分钟和第 31 分钟，依此类推)。
- ? (问号) 用于指定一个或另一个。在“日期”字段中，您可以输入 7，如果您不介意第 7 日是星期几，则可以在“星期几”字段中输入 ?。
- “日期”或“星期几”字段中的 L 通配符用于指定月或周的最后一天。
- “日期”字段中的 w 通配符用于指定工作日。在“日期”字段中，3w 用于指定最靠近当月的第三周的日。
- “星期几”字段中的 # 通配符用于指定一个月内所指定星期几的特定实例。例如，3#2 指该月的第二个星期二。3 指的是星期二，因为它是每周的第三天，2 是指该月内该类型的第二天。

限制

- 您无法在同一 Cron 表达式中为日期和星期几字段同时指定值。如果您在其中一个字段中指定值或 * (星号)，则必须在另一个字段中使用 ? (问号)。
- 不支持产生的速率快于一分钟的 Cron 表达式。
- 您不能将金丝雀设置为等待一年以上才能运行，因此 Year 字段中只能指定为 *。

示例

创建金丝雀时，您可以参考以下示例 cron 字符串。以下示例展示了使用 Amazon CLI 或 Amazon SDK 创建或更新金丝雀的正确语法。如果您使用的是 CloudWatch 控制台，请省略每个示例末尾的 *。

表达式	意义
0 10 * * ? *	每天上午的 10:00 (UTC) 运行
15 12 * * ? *	每天上午的 12:15 (UTC) 运行
0 18 ? * MON-FRI *	每星期一到星期五早上 6:00 (UTC) 运行
0 8 1 * ? *	每月第 1 天上午 8:00 (UTC) 运行
0/10 * ? * MON-SAT *	每周星期一到星期六每 10 分钟运行一次
0/5 8-17 ? * MON-FRI *	星期一到星期五的上午 8:00 和下午 5:55 (UTC) 之间，每 5 分钟运行一次

组

创建组可以将金丝雀相互关联，包括跨区域的金丝雀。使用组可以帮助您管理和自动化金丝雀，您还可以查看组中所有金丝雀的聚合运行结果和统计数据。

组是全局资源。创建组时，将在支持组的所有 Amazon 区域中复制该组，您可以将其中任何一个区域的金丝雀添加到该组，并在这些区域中查看该组。虽然组 ARN 格式反映了其创建位置的区域名称，但组不受限于任何区域。这意味着您可以将来自多个区域的金丝雀放在同一组中，然后使用该组在单个视图中查看和管理所有这些金丝雀。

除默认禁用的区域外，所有区域都支持组。有关这些区域的更多信息，请参阅[启用区域](#)。

每组最多可以容纳 10 个金丝雀。您的账户中最多可以有 20 个组。任何一个金丝雀最多可以成为 10 个组的成员。

创建组

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。

2. 在导航窗格中，选择 Application monitoring (应用程序监控)、Synthetics Canaries (Synthetics 金丝雀)。
3. 选择 Create Group (创建组)。
4. 在 Group Name (组名称) 下，输入组的名称。
5. 选择要与此组关联的金丝雀。要选择金丝雀，请在 Exact canary name (准确输入金丝雀名称) 中键入其完整名称，然后选择 Search (搜索)。然后选中金丝雀名称旁边的复选框。如果不同区域有多个同名金丝雀，请务必选择所需的金丝雀。

您可以重复此步骤，将最多 10 个金丝雀与该组关联。

6. (可选) 在 Tags (标签) 下，添加一个或多个键/值对作为此组的标签。标签可帮助您识别和组织 Amazon 资源并跟踪 Amazon 成本。有关更多信息，请参阅[标记 Amazon CloudWatch 资源 \(p. 903\)](#)。
7. 选择 Create Group (创建组)。

排查失败金丝雀的问题

如果您的金丝雀失败，请检查以下内容以排查问题。

一般故障排除

- 使用金丝雀详细信息页面查找更多信息。在 CloudWatch 控制台中，选择导航窗格中的 Canaries (金丝雀)，然后选择金丝雀的名称以打开金丝雀详细信息页。在 Availability (可用性) 选项卡上，选中 SuccessPercent 指标以查看问题是经常性还是间歇性问题。

仍在 Availability (可用性) 选项卡上时，选择失败的数据点以查看该失败运行的屏幕截图、日志和步骤报告 (如可用)。

如果步骤报告可用 (因为步骤是脚本的一部分)，请检查以查明失败的步骤，并查看相关屏幕截图以查看客户看到的问题。

您还可以检查 HAR 文件以了解是否有一个或多个请求失败。您可以使用日志对失败的请求和错误进行深入分析。最后，您可以将这些构件与成功的金丝雀运行中的构件进行比较，以确定问题。

默认情况下，CloudWatch Synthetics 会捕获 UI 金丝雀中每个步骤的屏幕截图。但您的脚本可能会配置为禁用屏幕截图。在调试期间，您可能希望再次启用屏幕截图。同样，对于 API 金丝雀，您可能希望在调试过程中查看 HTTP 请求和响应标头和响应体。有关如何将此数据包括在报告中，请参阅[executeHttpStep\(stepName, requestOptions, \[callback\], \[stepConfig\]\) \(p. 264\)](#)。

- 如果您最近对应用程序做出了部署，请执行回滚操作，然后稍后进行调试。
- 手动连接到端点，看看是否会重现相同问题。

主题

- [金丝雀运行时版本升级和降级问题 \(p. 276\)](#)
- [等待元素出现 \(p. 276\)](#)
- [节点不可见或不是用于 page.click\(\) 的 HTMLElement \(p. 276\)](#)
- [无法将构件上载到 S3，“异常：无法获取 S3 存储桶位置：访问被拒绝” \(p. 276\)](#)
- [“错误：协议错误 \(Runtime.callFunctionOn\)：目标已关闭。” \(p. 276\)](#)
- [“金丝雀失败。错误：无数据点 – 金丝雀显示超时”错误 \(p. 276\)](#)
- [尝试访问内部端点 \(p. 277\)](#)
- [跨域请求共享 \(CORS\) 问题 \(p. 277\)](#)
- [排查 VPC 上金丝雀的问题 \(p. 277\)](#)

金丝雀运行时版本升级和降级问题

如果您最近将金丝雀的运行时版本从 `syn-1.0` 升级到更高版本，则可能是跨域请求共享 (CORS) 问题。有关更多信息，请参阅[跨域请求共享 \(CORS\) 问题](#) (p. 277)。

如果您最近将金丝雀运行时版本降级到较旧版本，请检查以确保您正在使用的 CloudWatch Synthetics 函数在您降级到的旧运行时版本中可用。例如，`executeHttpStep` 函数可用于 `syn-nodejs-2.2` 或更高版本的运行时。要检查函数的可用性，请参阅[编写金丝雀脚本](#) (p. 237)。

Note

当您计划升级或降级金丝雀的运行时版本时，我们建议您首先克隆金丝雀并更新克隆金丝雀中的运行时版本。在验证了使用新运行时版本的克隆金丝雀正常运行后，您可以更新原始金丝雀的运行时版本并删除克隆金丝雀。

等待元素出现

分析日志和屏幕截图后，如果您发现脚本正在等待某个元素出现在屏幕上并超时，请检查相关的屏幕截图以查看该元素是否出现在页面上。验证 `xpath` 以确保该元素的准确性。

对于与 Puppeteer 相关的问题，请参阅 [Puppeteer 的 GitHub 页面](#) 或 Internet 论坛。

节点不可见或不是用于 `page.click()` 的 `HTMLElement`

如果节点不可见或不是用于 `page.click()` 的 `HTMLElement`，请首先验证您正在用于单击元素的 `xpath`。另外，如果您的元素位于屏幕底部，请调整视区。预设情况下，CloudWatch Synthetics 采用的视区为 1920 * 1080。您可以在启动浏览器时或使用 Puppeteer 函数 `page.setViewport` 另行设置视区。

无法将构件上传到 S3，“异常：无法获取 S3 存储桶位置：访问被拒绝”

如果您的金丝雀因 Amazon S3 错误而失败，CloudWatch Synthetics 无法上传屏幕截图、日志或因权限问题为金丝雀创建的报告。请检查以下事项：

- 检查金丝雀的 IAM 角色是否具有 `s3:ListAllMyBuckets` 权限、对正确的 Amazon S3 存储桶的 `s3:GetBucketLocation` 权限，以及对金丝雀存储其构件的存储桶的 `s3:PutObject` 权限。如果金丝雀执行可视化监控，该角色还需要存储桶的 `s3:GetObject` 权限。
- 如果金丝雀使用 Amazon KMS 客户托管式密钥而不是标准 Amazon 托管式密钥（默认）来加密，金丝雀的 IAM 角色可能没有使用该密钥加密或解密的权限。有关更多信息，请参阅[加密金丝雀构件](#) (p. 283)。
- 您的存储桶策略可能不允许金丝雀使用的加密机制。例如，如果您的存储桶策略要求使用特定的加密机制或 KMS 密钥，则必须为金丝雀选择相同的加密模式。

如果金丝雀执行可视化监控，请参阅[使用可视化监控时更新构件位置和加密](#) (p. 284) 了解更多信息。

“错误：协议错误 (Runtime.callFunctionOn)：目标已关闭。”

如果在页面或浏览器关闭后还有一些网络请求，则会出现此错误。您可能忘记了等待异步操作。执行脚本后，CloudWatch Synthetics 会关闭浏览器。关闭浏览器后执行任何异步操作可能会导致 `target closed error` 错误。

“金丝雀失败。错误：无数据点 – 金丝雀显示超时”错误

此错误指金丝雀运行超过了超时时间。金丝雀执行在 CloudWatch Synthetics 发布成功率 CloudWatch 指标或更新构件（如 HAR 文件、日志和屏幕截图）前已经停止。如果超时时间太低，您可以将其提高。

预设情况下，金丝雀的超时值等于其频率。您可以手动将超时值调整为小于或等于金丝雀频率。如果金丝雀频率较低，则必须增加频率以提高超时时间。在使用 CloudWatch Synthetics 控制台创建或更新金丝雀时，您可以在 Schedule (计划) 下调整频率和超时值。

请确保金丝雀超时值不要短于 15 秒钟，以预留 Lambda 冷启动以及启动金丝雀工具的时间。

发生此错误时，无法在 CloudWatch Synthetics 控制台中查看金丝雀构件。您可以使用 CloudWatch Logs 查看金丝雀的日志。

使用 CloudWatch Logs 查看金丝雀的日志

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在左侧导航窗格中，选择 Log groups (日志组)。
3. 在筛选器框中键入金丝雀名称来查找日志组。金丝雀的日志组名称为 `/aws/lambda/cwsyn-canaryName-randomId`。

尝试访问内部端点

如果希望您的金丝雀访问内部网络上的端点，我们建议您将 CloudWatch Synthetics 设置为使用 VPC。有关更多信息，请参阅[在 VPC 上运行金丝雀 \(p. 283\)](#)。

跨域请求共享 (CORS) 问题

在 UI 金丝雀中，如果某些网络请求失败并显示 403 或 `net::ERR_FAILED` 错误，请检查金丝雀是否启用了活动跟踪，并使用 Puppeteer 函数 `page.setExtraHTTPHeaders` 添加标头。若是如此，则网络请求失败可能是由跨域请求共享 (CORS) 限制所致。您可以通过禁用活动跟踪或删除额外的 HTTP 标头来确认是否属于这种情况。

为什么会发生这种情况？

在使用活动跟踪时，会向所有传出请求添加额外的标头以跟踪调用。通过使用 Puppeteer 的 `page.setExtraHTTPHeaders` 来添加跟踪标头或添加额外标头的方式来修改请求标头，会导致对 XMLHttpRequest (XHR) 请求进行 CORS 检查。

如果您不想禁用活动跟踪或删除额外标头，则可以更新 Web 应用程序以允许跨域访问，也可以在脚本中启动 Chrome 浏览器时使用 `disable-web-security` 标志来禁用 Web 安全。

您可以覆盖 CloudWatch Synthetics 所用的启动参数，并通过使用 CloudWatch Synthetics 启动函数传递其他 `disable-web-security` 标记参数。有关更多信息，请参阅[可用于 Node.js 金丝雀脚本的库函数 \(p. 245\)](#)。

Note

如果您使用的是 `syn-nodejs-2.1` 或更高运行时版本，则可以覆盖 CloudWatch Synthetics 所用的启动参数。

排查 VPC 上金丝雀的问题

如果您在创建或更新使用 VPC 的金丝雀后遇到问题，以下其中一部分可能会帮助您解决问题。

新的金丝雀处于错误状态或无法更新金丝雀

如果您创建要在 VPC 上运行的金丝雀，并且它立即进入错误状态，或者您无法更新金丝雀以在 VPC 上运行，则金丝雀的角色可能没有正确的权限。要在 VPC 上运行，金丝雀必须具有权限 `ec2:CreateNetworkInterface`、`ec2:DescribeNetworkInterfaces` 和

ec2:DeleteNetworkInterface。这些权限包含在 AWSLambdaVPCAccessExecutionRole 托管策略中。有关更多信息，请参阅[执行角色和用户权限](#)。

如果在创建金丝雀时发生此问题，必须删除此金丝雀，然后创建一个新的金丝雀。如果您使用 CloudWatch 控制台创建新金丝雀，请在 Access Permissions (访问权限) 下选择 Create a new role (创建新角色)。此时将创建一个新角色，该角色包含运行金丝雀所需的全部权限。

如果在更新金丝雀时发生此问题，可以再次更新金丝雀并提供具有必要权限的新角色。

“未返回测试结果”错误

如果金丝雀显示“未返回测试结果”错误，以下问题之一可能是导致这个问题的原因：

- 如果您的 VPC 不具有 Internet 访问权限，则必须使用 VPC 终端节点授予金丝雀对 CloudWatch 和 Amazon S3 的访问权限。您必须在 VPC 中启用 DNS 解析和 DNS 主机名选项，才能正确解析这些终端节点地址。有关更多信息，请参阅[在您的 VPC 中使用 DNS](#)。
- 金丝雀必须在 VPC 内的私有子网中运行。要检查是否存在此问题，请在 VPC 控制台中打开子网页面。检查您在配置金丝雀时选择的子网。如果它们具有通往互联网网关 (igw-) 的路径，则不是私有子网。

要帮助您解决这些问题，请参阅金丝雀日志。

要查看来自金丝雀的日志事件，请执行以下操作：

- 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
- 在导航窗格中，选择 Log groups (日志组)。
- 选择金丝雀日志组的名称。日志组名称以 /aws/lambda/cwsyn-*canary-name* 开头。

金丝雀脚本示例代码

本节包含对 CloudWatch Synthetics 金丝雀脚本的一些可能的函数进行说明的代码示例。

Node.js 和 Puppeteer 示例

设置 Cookie

网站依赖 Cookie 来提供自定义功能或跟踪用户。通过在 CloudWatch Synthetics 脚本中设置 Cookie，您可以模拟此自定义行为并对其进行验证。

例如，某个网站可能会对再次访问该网站的用户显示登录链接，而不是注册链接。

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const pageLoadBlueprint = async function () {

  let url = "http://smile.amazon.com/";

  let page = await synthetics.getPage();

  // Set cookies. I found that name, value, and either url or domain are required
  fields.
  const cookies = [{
    'name': 'cookie1',
    'value': 'val1',
    'url': url
  }, {
    'name': 'cookie2',
```

```
        'value': 'val2',
        'url': url
    }, {
        'name': 'cookie3',
        'value': 'val3',
        'url': url
    }
  ]];

  await page.setCookie(...cookies);

  // Navigate to the url
  await synthetics.executeStep('pageLoaded_home', async function (timeoutInMillis = 30000) {

    var response = await page.goto(url, {waitUntil: ['load', 'networkidle0'], timeout: timeoutInMillis});

    // Log cookies for this page and this url
    const cookiesSet = await page.cookies(url);
    log.info("Cookies for url: " + url + " are set to: " + JSON.stringify(cookiesSet));
  });
};

exports.handler = async () => {
  return await pageLoadBlueprint();
};
```

设备仿真

您可以编写模拟各种设备的脚本，以便粗略估计页面在这些设备上的外观和行为。

以下示例模拟的是 iPhone 6 设备。有关仿真的更多信息，请参阅 Puppeteer 文档中的 [page.emulate\(options\)](#)。

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');
const puppeteer = require('puppeteer-core');

const pageLoadBlueprint = async function () {

  const iPhone = puppeteer.devices['iPhone 6'];

  // INSERT URL here
  const URL = "https://amazon.com";

  let page = await synthetics.getPage();
  await page.emulate(iPhone);

  //You can customize the wait condition here. For instance,
  //using 'networkidle2' may be less restrictive.
  const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout: 30000});
  if (!response) {
    throw "Failed to load page!";
  }

  await page.waitFor(15000);

  await synthetics.takeScreenshot('loaded', 'loaded');

  //If the response status code is not a 2xx success code
  if (response.status() < 200 || response.status() > 299) {
    throw "Failed to load page!";
  }
}
```

```
};

exports.handler = async () => {
  return await pageLoadBlueprint();
};
```

多步骤 API 金丝雀

此示例代码演示了一个具有两个 HTTP 步骤的 API 金丝雀：对正面和负面测试用例测试同一个 API。传递步骤配置以启用请求/响应标头报告。此外，其会隐藏 Authorization（授权）标头和 X-Amz-Security-Token，因为它们二者包含用户凭证。

当此脚本用作金丝雀时，您可以查看有关每个步骤和相关 HTTP 请求的详细信息，例如步骤通过/失败、持续时间和性能指标（如 DNS 查找时间和第一字节时间）。也可以查看金丝雀运行的 2xx、4xx 和 5xx 的数量。

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const apiCanaryBlueprint = async function () {

  // Handle validation for positive scenario
  const validatePositiveCase = async function(res) {
    return new Promise((resolve, reject) => {
      if (res.statusCode < 200 || res.statusCode > 299) {
        throw res.statusCode + ' ' + res.statusMessage;
      }

      let responseBody = '';
      res.on('data', (d) => {
        responseBody += d;
      });

      res.on('end', () => {
        // Add validation on 'responseBody' here if required. For ex, your status
        code is 200 but data might be empty
        resolve();
      });
    });
  };

  // Handle validation for negative scenario
  const validateNegativeCase = async function(res) {
    return new Promise((resolve, reject) => {
      if (res.statusCode < 400) {
        throw res.statusCode + ' ' + res.statusMessage;
      }

      resolve();
    });
  };

  let requestOptionsStep1 = {
    'hostname': 'myproductsEndpoint.com',
    'method': 'GET',
    'path': '/test/product/validProductName',
    'port': 443,
    'protocol': 'https:'
  };

  let headers = {};
  headers['User-Agent'] = [synthetics.getCanaryUserAgentString(), headers['User-
Agent']].join(' ');
```

```
requestOptionsStep1['headers'] = headers;

// By default headers, post data and response body are not included in the report for
security reasons.
// Change the configuration at global level or add as step configuration for individual
steps
let stepConfig = {
  includeRequestHeaders: true,
  includeResponseHeaders: true,
  restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted header
values do not appear in report generated.
  includeRequestBody: true,
  includeResponseBody: true
};

await synthetics.executeHttpStep('Verify GET products API with valid name',
requestOptionsStep1, validatePositiveCase, stepConfig);

let requestOptionsStep2 = {
  'hostname': 'myproductsEndpoint.com',
  'method': 'GET',
  'path': '/test/canary/InvalidName(',
  'port': 443,
  'protocol': 'https:'
};

headers = {};
headers['User-Agent'] = [synthetics.getCanaryUserAgentString(), headers['User-
Agent']].join(' ');

requestOptionsStep2['headers'] = headers;

// By default headers, post data and response body are not included in the report for
security reasons.
// Change the configuration at global level or add as step configuration for individual
steps
stepConfig = {
  includeRequestHeaders: true,
  includeResponseHeaders: true,
  restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted header
values do not appear in report generated.
  includeRequestBody: true,
  includeResponseBody: true
};

await synthetics.executeHttpStep('Verify GET products API with invalid name',
requestOptionsStep2, validateNegativeCase, stepConfig);
};

exports.handler = async () => {
  return await apiCanaryBlueprint();
};
```

Python 和 Selenium 示例

以下示例 Selenium 代码是一个金丝雀，当未加载目标元素时，此金丝雀会失败并显示自定义错误消息。

```
from aws_synthetics.selenium import synthetics_webdriver as webdriver
from aws_synthetics.common import synthetics_logger as logger
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
```



```
def custom_selenium_script():
    # create a browser instance
    browser = webdriver.Chrome()
    browser.get('https://www.example.com/')
    logger.info('navigated to home page')
    # set cookie
    browser.add_cookie({'name': 'foo', 'value': 'bar'})
    browser.get('https://www.example.com/')
    # save screenshot
    browser.save_screenshot('signed.png')
    # expected status of an element
    button_condition = EC.element_to_be_clickable((By.CSS_SELECTOR, '.submit-button'))
    # add custom error message on failure
    WebDriverWait(browser, 5).until(button_condition, message='Submit button failed to
load').click()
    logger.info('Submit button loaded successfully')
    # browser will be quit automatically at the end of canary run,
    # quit action is not necessary in the canary script
    browser.quit()

# entry point for the canary
def handler(event, context):
    return custom_selenium_script()
```

金丝雀和 X-Ray 跟踪

您可以选择对使用 `syn-nodejs-2.0` 或更高版本运行时的金丝雀启用活动 Amazon X-Ray 跟踪。启用跟踪后，将针对由使用浏览器、Amazon SDK 或 HTTP 或 HTTPS 模块的金丝雀发出的所有调用发送 X-Ray 跟踪。启用了跟踪的金丝雀会显示在 CloudWatch Servicelens 和 X-Ray 中的服务地图上，即使它们没有向启用了跟踪的其他服务或应用程序发送请求。有关 X-Ray 跟踪的更多信息，请参阅[跟踪](#)。

Note

亚太地区（雅加达）尚不支持在金丝雀上激活 X-Ray 追踪。

当金丝雀出现在服务图上时，它会显示为新的客户端节点类型。您可以将鼠标悬停在某个金丝雀节点上以查看有关延迟、请求和故障的数据。您也可以选择该金丝雀节点以在页面底部查看更多数据。在页面的此区域，您可以选择 `View in Synthetics`（在 Synthetics 中查看）以跳转到 CloudWatch Synthetics 控制台，了解有关金丝雀的更多详细信息，或选择 `View Traces`（查看跟踪）来查看有关此金丝雀运行的跟踪的更多详细信息。

启用了跟踪的金丝雀的详细信息页面上还有一个 `Tracing`（跟踪）选项卡，其中包含有关金丝雀运行的跟踪和分段的详细信息。

启用跟踪会将金丝雀的运行时增加 2.5% 至 7%。

启用了跟踪的金丝雀必须使用具有以下权限的角色。如果您在创建金丝雀时使用控制台创建角色，则会为其授予这些权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Sid230934",
      "Effect": "Allow",
      "Action": [
        "xray:PutTraceSegments"
      ],
      "Resource": "*"
    }
  ]
}
```

```
}
```

金丝雀产生的跟踪会产生费用。有关 X-Ray 定价的更多信息，请参阅 [Amazon X-Ray 定价](#)。

在 VPC 上运行金丝雀

您可以在 VPC 上的终端节点上，以及在内部公有终端节点上运行金丝雀。要在 VPC 上运行金丝雀，您必须在 VPC 上同时启用 DNS 解析和 DNS 主机名选项。有关更多信息，请参阅[在您的 VPC 中使用 DNS](#)。

当您在 VPC 终端节点上运行金丝雀时，必须提供一种将其指标发送到 CloudWatch 并将其构件发送到 Amazon S3 的方法。如果 VPC 已启用 Internet 访问，则您无需再执行任何操作。金丝雀将在您的 VPC 中执行，但可以访问 Internet 以上传其指标和构件。

如果 VPC 尚未启用互联网访问功能，您有两个选项：

- 启用 VPC 的互联网访问功能。有关更多信息，请参阅以下部分 [为 VPC 上的金丝雀提供互联网访问权限](#) (p. 283)。
- 如果您希望将 VPC 保持私有状态，可以将金丝雀配置为通过私有 VPC 终端节点将其数据发送到 CloudWatch 和 Amazon S3。如果您尚未这样做，必须在 VPC 上为这些服务创建终端节点。有关更多信息，请参阅 [将 CloudWatch 和 CloudWatch Synthetics 与接口 VPC 终端节点结合使用](#) (p. 891) 和 [适用于 Amazon S3 的 Amazon VPC 终端节点](#)。

为 VPC 上的金丝雀提供互联网访问权限

请按照以下步骤为您的 VPC 金丝雀提供互联网访问权限。

为 VPC 上的金丝雀提供互联网访问

1. 在 VPC 上的公有子网中创建 NAT 网关。有关说明，请参阅[创建 NAT 网关](#)。
2. 在启动金丝雀的私有子网中的路由表中添加新路由。指定以下内容：
 - 对于 Destination (目标)，输入 **0.0.0.0/0**。
 - 对于 Target (目标)，选择 NAT Gateway (NAT 网关)，然后选择您创建的 NAT 网关的 ID。
 - 选择 Save routes (保存路由)。

有关在路由表中添加路由的更多信息，请参阅[在路由表中添加和删除路由](#)。

Note

确保通往 NAT 网关的路由处于活动状态。如果 NAT 网关被删除，但您尚未更新路由，则它们将处于黑洞状态。有关更多信息，请参阅[使用 NAT 网关](#)。

加密金丝雀构件

CloudWatch Synthetics 将屏幕截图、HAR 文件和报告等金丝雀构件存储在 Amazon S3 存储桶中。默认情况下，这些构件使用 Amazon 托管式密钥加密。有关更多信息，请参阅[Amazon 托管式密钥](#)。

您可以选择使用其他加密选项。CloudWatch Synthetics 支持以下选项：

- SSE-S3 – 使用 Amazon S3 托管式密钥进行服务器端加密 (SSE)。
- SSE-KMS – 使用 Amazon KMS 客户托管式的密钥进行服务器端加密 (SSE)。

如果您想通过 Amazon 托管式密钥使用默认加密选项，则不需要任何额外权限。

要使用 SSE-S3 加密，则需要将 SSE_S3 指定为创建或更新金丝雀时的加密模式。使用此加密模式不需要任何额外权限。有关更多信息，请参阅[使用服务器端加密与 Amazon S3 托管式加密密钥 \(SSE-S3\) 保护数据](#)。

要使用 Amazon KMS 客户托管式密钥，则需要将 SSE-KMS 指定为创建或更新金丝雀时的加密模式，并提供密钥的 Amazon Resource Name (ARN)。您还可以使用跨账户 KMS 密钥。

要使用客户托管式密钥，需要以下设置：

- 您的金丝雀的 IAM 角色必须具有使用密钥加密构件的权限。如果您使用可视化监控，还必须授予其解密构件的权限。

```
{
  "Version": "2012-10-17",
  "Statement": { "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": "Your KMS key ARN"
  }
}
```

- 您可以将 IAM 角色添加到密钥策略中，而不是向 IAM 角色添加权限。如果您对多个金丝雀使用相同的角色，您应该考虑这种方法。

```
{
  "Sid": "Enable IAM User Permissions",
  "Effect": "Allow",
  "Principal": {
    "AWS": "Your synthetics IAM role ARN"
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

- 如果您使用的是跨账户 KMS 密钥，请参阅[是否允许其他账户中的用户使用 KMS 密钥？](#)。

使用客户托管式密钥时查看加密的金丝雀构件

要查看金丝雀构件，请更新客户托管式密钥以向查看构件的用户提供 Amazon KMS 解密权限。或者，向正在查看构件的 IAM 用户或 IAM 角色添加解密权限。

默认 Amazon KMS 策略让账户中的 IAM 策略能够允许访问 KMS 密钥。如果您使用的是跨账户 KMS 密钥，请参阅[为什么跨账户用户在尝试访问由自定义 Amazon KMS 密钥加密的 Amazon S3 对象时会出现“拒绝访问”的错误？](#)

有关因 KMS 密钥而被拒绝访问的故障排除的更多信息，请参阅[密钥访问故障排除](#)。

使用可视化监控时更新构件位置和加密

为执行可视化监控，CloudWatch Synthetics 会将您的屏幕截图与在基准运行中获得的基准屏幕截图进行比较。如果您更新构件位置或加密选项，必须执行以下操作之一：

- 确保您的 IAM 角色对之前的 Amazon S3 位置和新位置的 Amazon S3 位置都具有足够的权限，以获取构件。同时，确保它对以前的和新的加密方法及 KMS 密钥都具有权限。
- 通过选择下一个金丝雀运行作为新基准来创建新基准。如果您使用此选项，则只需确保您的 IAM 角色对构件的新位置和加密选项具有足够的权限。

我们建议第二个选项，即选择下一次运行作为新基准。这避免了对之前用于金丝雀的构件位置或加密选项的依赖。

例如，假设您的金丝雀使用构件位置 A 和 KMS 密钥 K 来上传构件。如果您将金丝雀更新为构件位置 B 和 KMS 密钥 L，则可以确保您的 IAM 角色具有对两个构件位置（A 和 B）以及两个 KMS 密钥（K 和 L）的权限。或者，您可以选择下一次运行作为新基准，并确保您的金丝雀 IAM 角色具有对构件位置 B 和 KMS 密钥 L 的权限。

查看金丝雀统计数据和详细信息

您可以查看有关金丝雀的详细信息，并查看有关其运行的统计数据。

要能够查看有关金丝雀运行结果的所有详细信息，您必须登录具有足够权限的账户。有关更多信息，请参阅 [CloudWatch 金丝雀的必需角色和权限 \(p. 209\)](#)。

查看金丝雀统计数据和详细信息

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Application monitoring（应用程序监控）、Synthetics Canaries（Synthetics 金丝雀）。

在有关您已创建的金丝雀的详细信息中：

- Status (状态) 直观地显示了已通过其最新运行的金丝雀的数目。
 - Groups (组) 显示了您创建的组，并显示其中有多少组存在故障或警报的金丝雀。
 - Slowest performers (最慢执行者) 显示了金丝雀执行最慢的组和区域。这一数据是在选定的时间跨度内、将组或区域内所有金丝雀的平均持续时间相加，并除以组或区域中的金丝雀数量计算得出的。如果选择最慢组的指标，则会筛选该表，仅显示最慢组及其金丝雀。该表按平均持续时间排序。
 - 在页面底部附近有一张显示了所有金丝雀的表格。其中一列显示了为每个金丝雀创建的警报。仅显示符合金丝雀告警命名标准的告警。该标准为 `Synthetics-Alarm-canaryName-index`。您在 CloudWatch 控制台的 Synthetics 部分中创建的金丝雀告警会自动使用此命名约定。如果金丝雀告警是在 CloudWatch 控制台的 Alarms (告警) 部分或通过使用 Amazon CloudFormation 创建的，并且您未使用此命名约定，则告警会工作但不会出现在此列表中。
3. 要查看单个金丝雀的更多详细信息，请在 Canaries (金丝雀) 表中选择该金丝雀的名称。

在有关金丝雀的详细信息中：

- Availability (可用性) 选项卡显示了有关此金丝雀的最近运行的信息。

在 Canary runs (金丝雀运行) 下，您可以选择其中一条线，查看有关运行的详细信息。

在图表下，您可以选择 Steps (步骤)、Screenshot (屏幕截图)、Logs (日志) 或 HAR file (HAR 文件) 来查看这些类型的详细信息。如果金丝雀启用了活动跟踪，您还可以选择 Traces (跟踪) 查看来自金丝雀运行的跟踪信息。

金丝雀运行的日志存储在 S3 存储桶和 CloudWatch Logs 中。

屏幕截图显示客户查看您的网页的情况。您可以使用 HAR 文件 (HTTP 归档文件) 查看有关网页性能的详细数据。您可以分析 Web 请求列表并捕获性能问题，例如某个项的加载时间。日志文件显示金丝雀运行与网页之间的交互记录，可用于确定错误的详细信息。

如果金丝雀使用的是 `syn-nodejs-2.0-beta` 或更高版本的运行时，则您可以按状态代码、请求大小或持续时间对 HAR 文件进行排序。

Steps (步骤) 选项卡显示了金丝雀的步骤列表、每个步骤的状态、故障原因、步骤执行后的 URL、屏幕截图和步骤执行持续时间。对于具有 HTTP 步骤的 API 金丝雀，如果您使用的是 `syn-nodejs-2.2` 或更高版本的运行时，您可以查看步骤和相应的 HTTP 请求。

选择 HTTP Requests (HTTP 请求) 选项卡查看金丝雀发出的每个 HTTP 请求的日志。您可以查看请求/响应标头、响应体、状态代码、错误和性能计时 (总持续时间、TCP 连接时间、TLS 握手时间、第一字节时间和内容传输时间)。所有使用后台 HTTP/HTTPS 模块的 HTTP 请求都在此处捕获。

预设情况下, 在 API 金丝雀中, 出于安全原因, 请求标头、响应标头、请求体和响应体不包含在报告中。如果您选择将它们包含在报告中, 则这些数据仅存储在 S3 存储桶中。有关如何将此数据包括在报告中, 请参阅 [executeHttpStep\(stepName, requestOptions, \[callback\], \[stepConfig\]\)](#) (p. 264)。

支持文本、HTML 和 JSON 类型的响应体内容。支持文本/HTML、文本/纯文本、应用程序/JSON 和应用程序/x-amz-json-1.0 等内容类型。不支持经压缩的响应。

- Monitoring (监控) 选项卡显示了由此金丝雀发布的 CloudWatch 指标的图表。有关这些指标的更多信息, 请参阅 [金丝雀发布的 CloudWatch 指标](#) (p. 286)。

在金丝雀发布的 CloudWatch 图表下方, 是与金丝雀的 Lambda 代码相关的 Lambda 指标图表。

- Configuration (配置) 选项卡显示了有关金丝雀的配置和计划信息。
- Groups (组) 选项卡显示了与此金丝雀关联的组 (如果有)。
- Tags (标签) 选项卡显示了与金丝雀关联的标签。

金丝雀发布的 CloudWatch 指标

金丝雀会将以下指标发布到 CloudWatchSynthetics 命名空间中的 CloudWatch。有关查看 CloudWatch 指标的信息, 请参阅 [查看可用指标](#) (p. 100)。

指标	描述
SuccessPercent	此金丝雀成功运行且未发现任何故障的运行数百分比。 有效维度: CanaryName 有效统计数据: 平均值 单位: 百分比
Duration	金丝雀运行时长, 以毫秒为单位。 有效维度: CanaryName 有效统计数据: 平均值 单位: 毫秒
2xx	金丝雀执行的返回“OK”响应且响应代码介于 200 到 299 之间的网络请求数。 针对使用 syn-nodejs-2.0 或更高版本运行时的 UI 金丝雀和使用 syn-nodejs-2.2 或更高版本运行时的 API 金丝雀报告此指标。 有效维度: CanaryName 有效统计数据: Sum 单位: 计数
4xx	金丝雀执行的返回“Error”响应且响应代码介于 400 到 499 之间的网络请求数。 针对使用 syn-nodejs-2.0 或更高版本运行时的 UI 金丝雀和使用 syn-nodejs-2.2 或更高版本运行时的 API 金丝雀报告此指标。

指标	描述
	有效维度：CanaryName 有效统计数据：Sum 单位：计数
5xx	金丝雀执行的返回“Fault”响应且响应代码介于 500 到 599 之间的网络请求数。 针对使用 syn-nodejs-2.0 或更高版本运行时的 UI 金丝雀和使用 syn-nodejs-2.2 或更高版本运行时的 API 金丝雀报告此指标。 有效维度：CanaryName 有效统计数据：Sum 单位：计数
Failed	执行失败的金丝雀运行数。这些失败与金丝雀本身有关。 有效维度：CanaryName 有效统计数据：Sum 单位：计数
Failed requests	金丝雀在目标网站上执行但失败且无响应的 HTTP 请求数。 有效维度：CanaryName 有效统计数据：Sum 单位：计数
VisualMonitoringSuccessPercent	在金丝雀运行期间成功匹配基准屏幕截图的可视化比较数百分比。 有效维度：CanaryName 有效统计数据：平均值 单位：百分比
VisualMonitoringTotalComparisons	在金丝雀运行期间发生的可视化比较总数。 有效维度：CanaryName 单位：计数

Note

使用 Synthetics 库中 `executeStep()` 或 `executeHttpStep()` 方法的金丝雀还会发布 `SuccessPercent` 和 `Duration` 指标，指标包含每个步骤的 `CanaryName` 和 `StepName` 维度。

编辑或删除金丝雀

您可以编辑或删除现有金丝雀。

[编辑金丝雀](#)

在编辑金丝雀时，即使您并未更改其计划，该计划也会在您编辑金丝雀时相应地重置。例如，如果您的金丝雀每小时运行，而您编辑了该金丝雀，则该金丝雀会在编辑完成后立即运行，然后每一小时运行一次。

编辑或更新金丝雀

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Canaries（金丝雀）。
3. 选择金丝雀名称旁边的按钮，然后选择 Actions（操作）、Edit（编辑）。
4. （可选）如果此金丝雀执行屏幕截图可视化监控，并且您希望将金丝雀的下一次运行设置为基准，请选择 Set next run as new baseline（将下一次运行设置为新基准）。
5. （可选）如果此金丝雀执行屏幕截图可视化监控，并且您希望从可视化监控中删除某个屏幕截图，或者您希望在可视化比较期间指定要忽略部分屏幕截图，请在 Visual Monitoring（可视化监控）下选择 Edit Baseline（编辑基准）。

此时会显示屏幕截图，您可以执行下列操作之一：

- 若要删除用于可视化监控的屏幕截图，请选择 Remove screenshot from visual test baseline（从可视化测试基准中删除屏幕截图）。
 - 若要指定要在可视化对比过程中忽略的屏幕截图部分，请单击并拖动以绘制屏幕中要忽略的区域。针对要在对比中忽略的所有区域执行完成此操作后，选择 Save（保存）。
6. 对金丝雀做出任何其他更改，然后选择 Save（保存）。

删除金丝雀

当您删除金丝雀时，可以选择是否同时删除该金丝雀使用和其他资源。当您删除金丝雀时，还应删除以下内容：

- 此金丝雀使用的 Lambda 函数和层。其前缀为 `cwsyn-MyCanaryName`。
- 为此金丝雀创建的 CloudWatch 告警。这些告警的名称都以 `Synthetics-Alarm-MyCanaryName` 开头。有关删除警报的更多信息，请参阅[编辑或删除 CloudWatch 告警 \(p. 172\)](#)。
- Amazon S3 对象和存储桶，例如金丝雀的结果位置和构件位置。
- 为金丝雀创建的 IAM 角色。它们的名称为 `role/service-role/CloudWatchSyntheticsRole-MyCanaryName`。
- CloudWatch Logs 中为金丝雀创建的日志组。这些日志组具有以下名称：`/aws/lambda/cwsyn-MyCanaryName-randomId`。

在删除金丝雀之前，您可能需要查看金丝雀详细信息并记下此信息。这样，您就可以在删除金丝雀后删除正确的资源。

删除金丝雀

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Canaries（金丝雀）。
3. 如果金丝雀当前处于 RUNNING 状态，则您必须停止它。仅可删除处于 STOPPED、READY(NOT_STARTED) 或 ERROR 状态的金丝雀。

要停止金丝雀，请选择金丝雀名称旁边的按钮，然后选择 Actions（操作）、Stop（停止）。
4. 选择金丝雀名称旁边的按钮，然后选择 Actions（操作）、Delete（删除）。
5. 选择是否同时删除为该金丝雀创建和供其使用的其他资源。这包括 Lambda 函数和层，以及该金丝雀的 IAM 角色和 IAM policy。

要删除该金丝雀的 IAM 角色和 IAM policy，您必须拥有足够的权限。有关更多信息，请参阅[用于 CloudWatch Synthetics 的 Amazon 托管式（预定义）策略 \(p. 849\)](#)。
6. 在框中输入 **Delete**，然后选择 Delete（删除）。

7. 删除为金丝雀所用和为其创建的其他资源，如本节前面所列内容。

使用 Amazon EventBridge 监控金丝雀事件

在金丝雀更改状态或完成运行时，Amazon EventBridge 事件规则可以通知您。EventBridge 提供近乎实时的系统事件流，这些系统事件描述的是 Amazon 资源发生的变化。CloudWatch Synthetics 尽最大努力将这些事件发送到 EventBridge。尽最大努力发送事件是指 CloudWatch Synthetics 会尝试将所有事件发送到 EventBridge，但在某些极少数情况下，可能无法发送事件。EventBridge 至少会处理所接收的所有事件一次。此外，事件侦听器可能无法按事件的发生顺序接收事件。

Note

Amazon EventBridge 是一种事件总线服务，您可以用其轻松地将应用程序与来自各种来源的数据相连接。有关更多信息，请参阅 Amazon EventBridge 用户指南中的[什么是 Amazon EventBridge ?](#)。

CloudWatch Synthetics 会在金丝雀改变状态或完成运行时发出事件。您可以创建一个包含事件模式的 EventBridge 规则，以匹配从 CloudWatch Synthetics 发送的所有事件类型，或者仅匹配特定事件类型。当金丝雀触发规则时，EventBridge 会调用规则中定义的目标操作。这样，您就可以发送通知、捕获事件信息以及采取纠正操作，以响应金丝雀的状态更改或完成运行。例如，您可以为以下使用案例创建规则：

- 在金丝雀运行失败时执行调查
- 在金丝雀变为 ERROR 状态时执行调查
- 跟踪金丝雀的生命周期
- 作为工作流程的一部分监控金丝雀运行成功或故障

CloudWatch Synthetics 发送的事件示例

本节列出了 CloudWatch Synthetics 发送的事件示例。有关事件格式的更多信息，请参阅 [EventBridge 中的事件和事件模式](#)。

金丝雀状况更改

在此事件类型中，current-state 和 previous-state 的值可以是：

CREATING | READY | STARTING | RUNNING | UPDATING | STOPPING | STOPPED | ERROR

```
{
  "version": "0",
  "id": "8a99ca10-1e97-2302-2d64-316c5dedfd61",
  "detail-type": "Synthetics Canary Status Change",
  "source": "aws.synthetics",
  "account": "123456789012",
  "time": "2021-02-09T22:19:43Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "account-id": "123456789012",
    "canary-id": "EXAMPLE-dc5a-4f5f-96d1-989b75a94226",
    "canary-name": "events-bb-1",
    "current-state": "STOPPED",
    "previous-state": "UPDATING",
    "source-location": "NULL",
    "updated-on": 1612909161.767,
    "changed-config": {
      "executionArn": {
        "previous-value":
          "arn:aws:lambda:us-east-1:123456789012:function:cwsyn-events-bb-1-af3e3a05-dc5a-4f5f-96d1-989EXAMPLE:1",

```

```
      "current-value":  
        "arn:aws:lambda:us-east-1:123456789012:function:cwsyn-events-bb-1-af3e3a05-  
dc5a-4f5f-96d1-989EXAMPLE:2"  
      },  
      "vpcId": {  
        "current-value": "NULL"  
      },  
      "testCodeLayerVersionArn": {  
        "previous-  
value": "arn:aws:lambda:us-east-1:123456789012:layer:cwsyn-events-bb-1-af3e3a05-  
dc5a-4f5f-96d1-989EXAMPLE:1",  
        "current-value":  
          "arn:aws:lambda:us-east-1:123456789012:layer:cwsyn-events-bb-1-af3e3a05-  
dc5a-4f5f-96d1-989EXAMPLE:2"  
      }  
    },  
    "message": "Canary status has changed"  
  }  
}
```

已完成的成功金丝雀运行

```
{  
  "version": "0",  
  "id": "989EXAMPLE-f4a5-57a7-1a8f-d9cc768a1375",  
  "detail-type": "Synthetics Canary TestRun Successful",  
  "source": "aws.synthetics",  
  "account": "123456789012",  
  "time": "2021-02-09T22:24:01Z",  
  "region": "us-east-1",  
  "resources": [],  
  "detail": {  
    "account-id": "123456789012",  
    "canary-id": "989EXAMPLE-dc5a-4f5f-96d1-989b75a94226",  
    "canary-name": "events-bb-1",  
    "canary-run-id": "c6c39152-8f4a-471c-9810-989EXAMPLE",  
    "artifact-location": "cw-syn-results-123456789012-us-  
east-1/canary/us-east-1/events-bb-1-ec3-28ddb266797/2021/02/09/22/23-41-200",  
    "test-run-status": "PASSED",  
    "state-reason": "null",  
    "canary-run-timeline": {  
      "started": 1612909421,  
      "completed": 1612909441  
    },  
    "message": "Test run result is generated successfully"  
  }  
}
```

已完成的失败金丝雀运行

```
{  
  "version": "0",  
  "id": "2644b18f-3e67-5ebf-cdfd-bf9f91392f41",  
  "detail-type": "Synthetics Canary TestRun Failure",  
  "source": "aws.synthetics",  
  "account": "123456789012",  
  "time": "2021-02-09T22:24:27Z",  
  "region": "us-east-1",  
  "resources": [],  
  "detail": {  
    "account-id": "123456789012",  
    "canary-id": "af3e3a05-dc5a-4f5f-96d1-9989EXAMPLE",  
    "canary-name": "events-bb-1",  
    "canary-run-id": "0df3823e-7e33-4da1-8194-b04e4d4a2bf6",  
    "message": "Test run result is generated successfully"  
  }  
}
```

```
        "artifact-location": "cw-syn-results-123456789012-us-east-1/canary/us-east-1/events-bb-1-ec3-989EXAMPLE/2021/02/09/22/24-21-275",
        "test-run-status": "FAILED",
        "state-reason": "\"Error: net::ERR_NAME_NOT_RESOLVED\"",
        "canary-run-timeline": {
            "started": 1612909461,
            "completed": 1612909467
        },
        "message": "Test run result is generated successfully"
    }
}
```

可能是事件重复或者顺序颠倒。要确定事件的顺序，请使用 `time` 属性。

创建 EventBridge 规则的先决条件

在为 CloudWatch Synthetics 创建 EventBridge 规则之前，您应执行以下操作：

- 熟悉 EventBridge 中的事件、规则和目标。
- 创建和配置由 EventBridge 规则调用的目标。规则可以调用许多类型的目标，包括：
 - Amazon SNS 主题
 - Amazon Lambda 函数
 - Kinesis Streams
 - Amazon SQS 队列

有关更多信息，请参阅 Amazon EventBridge 用户指南中的 [什么是 Amazon EventBridge ?](#) 和 [Amazon EventBridge 入门](#)。

创建 EventBridge 规则 (CLI)

以下示例中的步骤将创建一条 EventBridge 规则，该规则会在 `us-east-1` 中名为 `my-canary-name` 的金丝雀完成一次运行或更改状态时发布 Amazon SNS 主题。

1. 创建规则。

```
aws events put-rule \
  --name TestRule \
  --region us-east-1 \
  --event-pattern "{\"source\": [\"aws.synthetic\"], \"detail\": {\"canary-name\": [\"my-canary-name\"]}}"
```

模式中省略的属性都将会被忽略。

2. 将主题添加为规则目标。

- 将 `topic-arn` 替换为 Amazon SNS 主题的 Amazon Resource Name (ARN)。

```
aws events put-targets \
  --rule TestRule \
  --targets "Id"="1", "Arn"="topic-arn"
```

Note

要允许 Amazon EventBridge 调用您的目标主题，您必须将基于资源的策略添加到您的主题中。有关更多信息，请参阅 Amazon EventBridge 用户指南中的 [Amazon SNS 权限](#)。

有关更多信息，请参阅 Amazon EventBridge 用户指南中的 [EventBridge 中的事件和事件模式](#)。

使用 CloudWatch Evidently 执行启动和 A/B 实验

通过在推出新功能时向指定百分比的用户提供新功能，您可以使用 Amazon CloudWatch Evidently 对功能进行安全地验证。您可以监控新功能的性能，以帮助您决定何时向用户增加流量。这有助于您在完全启动该功能之前，降低风险并识别意外后果。

您还可以进行 A/B 实验，以根据证据和数据制定功能设计决策。一个实验可以同时测试多达五个功能变体。Evidently 会收集实验数据并使用统计方法对其进行分析。它还会就哪些功能变体的性能更好提供明确建议。您可以测试面向用户的功能和后端功能。

Evidently 定价

Evidently 会根据 Evidently 事件和 Evidently 分析单位向您的账户收取费用。Evidently 事件包括单击量和页面浏览量等数据事件以及确定向用户提供的功能变体的任务事件。

Evidently 分析单元根据您在 Evidently 中创建的规则，从 Evidently 事件中生成。分析单元是事件的规则匹配数。例如，用户单击事件可能会产生单个 Evidently 分析单元，即单击计数。另一个示例是用户结算事件，其可能会产生两个 Evidently 分析单元，即结算值和购物车中的项目数量。有关定价的更多信息，请参阅 [Amazon CloudWatch 定价](#)。

CloudWatch Evidently 目前在以下区域中可用：

- 美国东部（俄亥俄）
- 美国东部（弗吉尼亚北部）
- 美国西部（俄勒冈州）
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太区域（东京）
- 欧洲（法兰克福）
- 欧洲（爱尔兰）
- 欧洲（斯德哥尔摩）

主题

- [使用 Evidently 的 IAM 策略 \(p. 293\)](#)
- [创建项目、功能、启动和实验 \(p. 294\)](#)
- [管理功能、启动和实验 \(p. 306\)](#)
- [将代码添加到应用程序 \(p. 308\)](#)
- [项目数据存储 \(p. 310\)](#)
- [Evidently 是如何计算结果的 \(p. 312\)](#)
- [在控制面板中查看启动结果 \(p. 313\)](#)
- [在控制面板中查看实验结果 \(p. 313\)](#)
- [CloudWatch Evidently 如何收集和存储数据 \(p. 314\)](#)
- [使用 Evidently 的服务相关角色 \(p. 315\)](#)
- [CloudWatch Evidently 配额 \(p. 316\)](#)
- [教程：使用 Evidently 示例应用程序进行 A/B 测试 \(p. 317\)](#)

使用 Evidently 的 IAM 策略

要完全管理 CloudWatch Evidently，您必须以具有以下权限的 IAM 用户或角色的身份登录：

- AmazonCloudWatchEvidentlyFullAccess 策略
- ResourceGroupsandTagEditorReadOnlyAccess 策略

此外，为能够创建在 Amazon S3 或 CloudWatch Logs 中存储评估事件的项目，您需要以下权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketPolicy",
        "s3:PutBucketPolicy",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs>DeleteLogDelivery",
        "logs:DescribeResourcePolicies",
        "logs:PutResourcePolicy"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

CloudWatch RUM 集成的其他权限

此外，如果您打算管理与 Amazon CloudWatch RUM 集成的 Evidently 启动或实验，并使用 CloudWatch RUM 指标进行监控，您需要 AmazonCloudWatchRUMFullAccess 策略。要创建 IAM 角色以授予 CloudWatch RUM Web 客户端向 CloudWatch RUM 发送数据的权限，您需要以下权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:CreatePolicy",
        "iam:AttachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::*:role/service-role/CloudWatchRUMEvidentlyRole-*",
        "arn:aws:iam::*:policy/service-role/CloudWatchRUMEvidentlyPolicy-*"
      ]
    }
  ]
}
```

对 Evidently 的只读访问权限

对于需要查看 Evidently 数据但不需要创建 Evidently 资源的其他用户，您可以授予 AmazonCloudWatchEvidentlyReadOnlyAccess 策略。

创建项目、功能、启动和实验

无论是出于功能启动或者 A/B 实验目的，若要开始使用 CloudWatch Evidently，您都要先创建项目。项目是资源的逻辑分组。在项目中，您可以创建具有要测试或启动的变体的功能。您可以在创建启动或实验之前创建功能，也可以同时创建。

主题

- [创建新项目 \(p. 294\)](#)
- [使用客户端评估 - 由 Amazon AppConfig 提供支持 \(p. 295\)](#)
- [向项目添加功能 \(p. 298\)](#)
- [使用细分来聚焦受众 \(p. 299\)](#)
- [创建启动 \(p. 302\)](#)
- [创建实验 \(p. 304\)](#)

创建新项目

请使用以下步骤设置新的 CloudWatch Evidently 项目。

要创建新的 CloudWatch Evidently 项目

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring（监控应用程序）、Evidently。
3. 请选择 Create project（创建项目）。
4. 对于 Project name（项目名称），输入一个名称，用于在 CloudWatch Evidently 控制台中标识此项目。

您可以选择添加项目描述。

5. 对于 Evaluation event storage（评估事件存储），请选择是否要存储使用 Evidently 收集的评估事件。即使您没有存储这些事件，Evidently 也会将其聚合以创建可在 Evidently 控制面板中查看的指标和其他实验数据。有关更多信息，请参阅[项目数据存储 \(p. 310\)](#)。
6. 对于 Use client-side evaluation（使用客户端评估），请选择是否要为此项目启用客户端评估。通过客户端评估，您的应用程序可以在本地为用户会话分配变体，而不是调用 [EvaluateFeature](#) 操作。这减少了 API 调用带来的延迟和可用性风险。有关更多信息，请参阅[使用客户端评估 - 由 Amazon AppConfig 提供支持 \(p. 295\)](#)。

要创建具有客户端评估的项目，您必须具有 `evidently:ExportProjectAsConfiguration` 权限。

如果您启用客户端评估，还要执行以下操作：

- a. 选择是使用现有 Amazon AppConfig 应用程序还是创建一个新的应用程序。
- b. 选择是使用现有 Amazon AppConfig 环境还是创建一个新的环境。

有关 Amazon AppConfig 中的应用程序和环境的更多信息，请参阅[Amazon AppConfig 的工作原理](#)。

7. （可选）要在此项目添加标签，请选择 Tags（标签）、Add new tag（添加新标签）。

然后，对于 Key（键），输入标签的名称。您可以在 Value（值）中添加可选的标签值。

要添加其他标签，请再次选择 Add new tag（添加新标签）。

有关更多信息，请参阅[标记 Amazon 资源](#)。

8. 请选择 Create project（创建项目）。

使用客户端评估 - 由 Amazon AppConfig 提供支持

您可以在项目中使用客户端评估 - 由 Amazon AppConfig 提供支持（客户端评估），它允许您的应用程序在本地为用户会话分配变体，而不是通过调用 [EvaluateFeature](#) 操作。这减少了 API 调用带来的延迟和可用性风险。

要使用客户端评估，请附上 Amazon AppConfig Lambda 扩展程序作为您的 Lambda 函数的一个层并配置环境变量。客户端评估在本地主机上作为侧进程运行。然后，您可以针对 localhost 调用 `EvaluationFeature` 和 `PutProjectEvent`。客户端评估过程负责变体分配、缓存和数据同步。有关 Amazon AppConfig 的更多信息，请参阅 [Amazon AppConfig 的工作原理](#)。

当与 Amazon AppConfig 集成时，向 Evidently 指定一个 Amazon AppConfig 应用程序 ID 和一个 Amazon AppConfig 环境 ID。您可以在 Evidently 项目中使用相同的应用程序 ID 和环境 ID。

当您创建一个启用了客户端评估的项目时，Evidently 会为该项目创建一个 Amazon AppConfig 配置文件。每个项目的配置文件将有所不同。

客户端评估访问控制

Evidently 客户端评估使用的访问控制机制与 Evidently 的其余部分不同。我们强烈建议您了解这一点，以便实施适当的安全措施。

利用 Evidently，您可以创建 IAM 策略来限制用户可对各个资源执行的操作。例如，您可以创建不允许用户拥有 `EvaluateFeature` 操作的用户角色。有关可通过 IAM 策略控制的 Evidently 操作的更多信息，请参阅 [Amazon CloudWatch Evidently 定义的操作](#)。

客户端评估模型允许对使用项目元数据的 Evidently 功能进行本地评估。启用了客户端评估的项目的用户可以针对本地主机端点调用 `EvaluateFeature` API，此 API 调用无法到达 Evidently，也未通过 Evidently 服务的 IAM 策略进行身份验证。即使用户没有 IAM 权限来使用 `EvaluateFeature` 操作，此调用也可成功。但是，用户仍然需要 `PutProjectEvents` 权限使代理缓冲评估事件或自定义事件，以及将数据异步卸载到 Evidently。

此外，用户必须具有 `evidently:ExportProjectAsConfiguration` 权限才能创建使用客户端评估的项目。这可以帮助您对客户端评估期间调用的 `EvaluateFeature` 操作进行控制访问。

如果您不小心，客户端评估安全模型可能会颠覆您对 Evidently 其余部分设置的策略。具有 `evidently:ExportProjectAsConfiguration` 权限的用户可以创建启用了客户端评估的项目，然后使用 `EvaluateFeature` 工作进行该项目的客户端评估，即使它们在 IAM policy 中被明确拒绝 `EvaluateFeature` 操作。

Lambda 入门

Evidently 当前通过使用 Amazon Lambda 环境支持客户端评估。要开始使用，请先决定要使用哪一个 Amazon AppConfig 应用程序和环境。选择现有的应用程序和环境，或创建新的应用程序和环境。

以下示例 Amazon AppConfig Amazon CLI 命令可创建应用程序和环境。

```
aws appconfig create-application --name YOUR_APP_NAME
```

```
aws appconfig create-environment --application-id YOUR_APP_ID --name YOUR_ENVIRONMENT_NAME
```

接下来，使用这些 Amazon AppConfig 资源创建一个 Evidently 项目。有关更多信息，请参阅[创建新项目](#) (p. 294)。

使用 Lambda 层在 Lambda 中支持客户端评估。这是属于 `AWS-AppConfig-Extension` 一部分的公共层，是 Amazon AppConfig 服务创建的公共 Amazon AppConfig 扩展程序。有关 Lambda 层的更多信息，请参阅[层](#)。

要使用客户端评估，您必须将此层添加到 Lambda 函数并配置权限和环境变量。

要将 Evidently 客户端评估 Lambda 层添加到您的 Lambda 函数并进行配置

1. 如果尚未创建 Lambda 函数，请创建一个。
2. 将客户端评估层添加到函数。您可以指定它的 ARN，也可以从 Amazon 层的列表中选择（如果尚未创建）。有关更多信息，请参阅[配置函数以使用层](#)和[Amazon AppConfig Lambda 扩展程序的可用版本](#)。
3. 创建名为 EvidentlyAppConfigCachingAgentPolicy 具有以下内容的 IAM policy，并将其附加到函数的执行角色。有关更多信息，请参阅[Lambda 执行角色](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "appconfig:GetLatestConfiguration",
        "appconfig:StartConfigurationSession",
        "evidently:PutProjectEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

4. 添加所需的环境变量 AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS 到您的 Lambda 函数。此环境变量指定 Evidently 项目与 Amazon AppConfig 资源之间的映射。

如果您将此函数用于一个 Evidently 项目，请将环境变量的值设置为：applications/**APP_ID**/environments/**ENVIRONMENT_ID**/configurations/**PROJECT_NAME**

如果您将此函数用于多个 Evidently 项目，请使用逗号分隔值，如下示例所示：applications/**APP_ID_1**/environments/**ENVIRONMENT_ID_1**/configurations/**PROJECT_NAME_1**, applications/**APP_ID_2**/environments/**ENVIRONMENT_ID_2**/configurations/**PROJECT_NAME_2**

5. （可选）设置其他环境变量。有关更多信息，请参阅[配置 Amazon AppConfig Lambda 扩展程序](#)。
6. 在您的应用程序中，通过将 EvaluateFeature 发送到 localhost 进行本地 Evidently 评估。

Python 示例：

```
import boto3
from botocore.config import Config

def lambda_handler(event, context):
    local_client = boto3.client(
        'evidently',
        endpoint_url="http://localhost:2772",
        config=Config(inject_host_prefix=False)
    )
    response = local_client.evaluate_feature(
        project=event['project'],
        feature=event['feature'],
        entityId=event['entityId']
    )
    print(response)
```

Node.js 示例：

```
const AWS = require('aws-sdk');
const evidently = new AWS.Evidently({
    region: "us-west-2",
```

```
    endpoint: "http://localhost:2772",
    hostPrefixEnabled: false
  });

exports.handler = async (event) => {

  const evaluation = await evidently.evaluateFeature({
    project: 'John_ETCProject_Aug2022',
    feature: 'Feature_IceCreamFlavors',
    entityId: 'John'
  }).promise()

  console.log(evaluation)
  const response = {
    statusCode: 200,
    body: evaluation,
  };
  return response;
};
```

Kotlin 示例：

```
String localhostEndpoint = "http://localhost:2772/"
public AmazonCloudWatchEvidentlyClient getEvidentlyLocalClient() {
    return AmazonCloudWatchEvidentlyClientBuilder.standard()

        .withEndpointConfiguration(AwsClientBuilder.EndpointConfiguration(localhostEndpoint,
            region))

        .withClientConfiguration(ClientConfiguration().withDisableHostPrefixInjection(true))
            .withCredentials(credentialsProvider)
            .build();
}

AmazonCloudWatchEvidentlyClient evidently = getEvidentlyLocalClient();

// EvaluateFeature via local client.
EvaluateFeatureRequest evaluateFeatureRequest = new EvaluateFeatureRequest().builder()
    .withProject(${YOUR_PROJECT}) //Required.
    .withFeature(${YOUR_FEATURE}) //Required.
    .withEntityId(${YOUR_ENTITY_ID}) //Required.
    .withEvaluationContext(${YOUR_EVAL_CONTEXT}) //Optional: a JSON object of attributes
        that you can optionally pass in as part of the evaluation event sent to Evidently.
    .build();

EvaluateFeatureResponse evaluateFeatureResponse =
    evidently.evaluateFeature(evaluateFeatureRequest);

// PutProjectEvents via local client.
PutProjectEventsRequest putProjectEventsRequest = new
    PutProjectEventsRequest().builder()
        .withData(${YOUR_DATA})
        .withTimeStamp(${YOUR_TIMESTAMP})
        .withType(${YOUR_TYPE})
        .build();

PutProjectEvents putProjectEventsResponse =
    evidently.putProjectEvents(putProjectEventsRequest);
```

配置客户端向 Evidently 发送数据的频率

要指定客户端评估向 Evidently 发送数据的频率，您可以选择配置两个环境变量。

- `AWS_APPCONFIG_EXTENSION_EVIDENTLY_EVENT_BATCH_SIZE` 指定每个项目在将事件发送到 Evidently 之前要批处理的事件数量。有效范围是介于 1 与 50 之间的整数，默认值为 40。
- `AWS_APPCONFIG_EXTENSION_EVIDENTLY_BATCH_COLLECTION_DURATION` 指定在将事件发送到 Evidently 之前等待事件的持续时间（以秒为单位）。默认值为 30。

问题排查

使用以下信息帮助对使用 CloudWatch Evidently 与客户端评估 - 由 Amazon AppConfig 提供支持产生的问题进行排查。

调用 `EvaluateFeature` 操作时出现错误 (`BadRequestException`) : 提供的路径不支持 HTTP 方法

您的环境变量可能配置不正确。例如，您可能曾将 `EVIDENTLY_CONFIGURATIONS` 用作环境变量名称，而不是 `AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS`。

`ResourceNotFoundException` : 未找到部署

您对项目元数据的更新尚未部署到 Amazon AppConfig。在您用于客户端评估的 Amazon AppConfig 环境中检查活动部署。

`ValidationException` : 没有对项目进行 Evidently 配置

您的 `AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS` 环境变量可能使用不正确的项目名称进行了配置。

向项目添加功能

CloudWatch Evidently 中的功能代表您要启动或要测试其变体的功能。

必须先创建项目，才能添加功能。有关更多信息，请参阅[创建新项目 \(p. 294\)](#)。

向项目添加功能

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring（监控应用程序）、Evidently。
3. 请选择项目名称。
4. 请选择 Add feature（添加功能）。
5. 对于 Feature name（功能名称），输入用于标识此项目中此功能的名称。

您也可以添加功能描述。

6. 对于 Feature variations（功能变体）和 Variation type（变体类型），请选择 Boolean、Long（长整数）、Double（双精度）或者 String（字符串）。有关更多信息，请参阅[变体类型 \(p. 299\)](#)。
7. 最多可向功能添加五个变体。每个变体的 Value（值）必须对您选择的 Variation type（变体类型）有效。

指定其中一个变体作为默认变体。这是将与其他变体进行比较的基准，并且应作为现在向用户提供的变体。这也是向未添加到此功能的启动或实验中的用户提供的变体。

8. 请选择 Sample code（示例代码）。此代码示例显示了为设置变体并为其分配用户会话，您需要添加到应用程序中的内容。您可以在 JavaScript、Java 和 Python 之间选择代码。

您现在无需将代码添加到应用程序中，但必须在开始启动或实验之前添加。

有关更多信息，请参阅[将代码添加到应用程序 \(p. 308\)](#)。

9. （可选）要指定某些用户可以始终看到某个变体，请选择 Overrides（覆盖）、Add override（添加覆盖）。然后，通过在 Identifier（标识符）中输入用户 ID、账户 ID 或其他一些标识符，指定用户并指定其可以看到的变体。

如果您想确保自己的测试团队成员或其他内部用户可以看到特定变体，这非常有用。已分配覆盖的用户的话无益于启动或试验指标。

您可以通过再次选择 Add override (添加覆盖)，为多达 10 个用户重复此操作。

10. (可选) 要在此功能添加标签，请选择 Tags (标签)、Add new tag (添加新标签)。

然后，对于 Key (键)，输入标签的名称。您可以在 Value (值) 中添加可选的标签值。

要添加其他标签，请再次选择 Add new tag (添加新标签)。

有关更多信息，请参阅[标记 Amazon 资源](#)。

11. 请选择 Add feature (添加功能)。

变体类型

创建功能并定义变体时，必须选择 variation type (变体类型)。可能的类型如下：

- 布尔值
- 长整数
- 双精度浮点数
- 字符串

变体类型设置了代码中不同变体的区分方式。您可以使用变体类型来简化 CloudWatch Evidently 的实施，也可以简化启动和实验中的功能修改过程。

例如，如果您使用长整数变体类型定义功能，则指定用于区分变体的整数可以是直接传递到代码中的数字。示例可能为按钮的像素大小测试。变体类型的值可以是每个变体中所使用的像素数。每个变体的代码可以读取变体类型值并将其用作按钮大小。要测试新的按钮大小，您可以更改用于变体值的数字，而无需更改任何其他代码。

除想要进行 A/A 测试以初步试用 CloudWatch Evidently 或者出于其他目的这么做之外，当您在功能中为变体类型设置值时，应避免向多个变体分配相同的值。

当将 JSON 作为类型时，Evidently 不向其提供本机支持，但您可以在 String (字符串) 变体类型中传入 JSON，并在代码中解析该 JSON。

使用细分来聚焦受众

您可以定义受众细分并将它们用于发布和实验中。细分是您的受众中具有一个或多个相同特征的部分。例如 Chrome 浏览器用户、欧洲用户或欧洲的 Firefox 浏览器用户，他们也符合您的应用程序收集的其他标准，例如年龄。

在实验中使用细分将该实验限制为仅评估符合细分标准的用户。当您在发布中使用一个或多个细分时，您可以为不同的受众细分定义不同的流量拆分。

细分规则模式语法

要创建细分，请定义细分规则模式。指定要用于评估用户会话是否在细分中的属性。将您创建的模式与 Evidently 在用户会话中找到的 evaluationContext 的值进行比较。有关更多信息，请参阅[使用 EvaluateFeature \(p. 309\)](#)。

要创建细分规则模式，请指定希望模式匹配的字段。您也可以在模式中使用逻辑，例如 And、Or、Not 和 Exists。

为了使 evaluationContext 匹配模式，evaluationContext 必须匹配规则模式的所有部分。Evidently 忽略了 evaluationContext 中并不包含在规则模式中的字段。

规则模式匹配的值遵循 JSON 规则。可以包括用引号 (") 括起来的字符串、数字和关键字 true、false 和 null。

对于字符串，Evidently 使用精确的逐个字符匹配，而不进行小写化或任何其他字符串标准化。因此，规则匹配区分大小写。例如，如果您的 evaluationContext 包含一个 browser 属性，但您的规则模式检查 Browser，则它将不匹配。

此外，Evidently 使用字符串表示。例如，300、300.0 和 3.0e2 不相等。

在编写规则模式来匹配 evaluationContext 时，您可以使用 TestSegmentPattern API 或 test-segment-pattern CLI 命令以测试模式是否匹配正确的 JSON。有关更多信息，请参阅 [TestSegmentPattern](#)。

以下汇总显示了 Evidently 细分模式中提供的所有比较运算符。

Comparison (比较)	示例	Rule syntax (规则语法)
Null	用户 ID 为空	<pre>{ "UserID": [null] }</pre>
Empty	姓氏为空	<pre>{ "LastName": [""] }</pre>
等于	浏览器是“Chrome”	<pre>{ "Browser": ["Chrome"] }</pre>
And	国家/地区为“法国”，设备为“移动设备”	<pre>{ "Country": ["France"], "Device": ["Mobile"] }</pre>
或 (单个属性的多个值)	浏览器是“Chrome”或“Firefox”	<pre>{ "Browser": ["Chrome", "Firefox"] }</pre>
或 (不同的属性)	浏览器为“Safari”或设备为“平板电脑”	<pre>{ "\$or": [{ "Browser": ["Safari"] }, { "Device": ["Tablet"] }] }</pre>
非	浏览器是除“Safari”以外的任何浏览器	<pre>{ "Browser": [{ "anything-but": ["Safari"] }] }</pre>
数值 (等于)	价格为 100	<pre>{ "Price": [{ "numeric": ["=", 100] }] }</pre>

Comparison (比较)	示例	Rule syntax (规则语法)
		<pre>}</pre>
数值 (范围)	价格大于 10，且小于等于 20	<pre>{ "Price": [{ "numeric": [">", 10, "<=", 20] }] }</pre>
Exists	年龄字段存在	<pre>{ "Age": [{ "exists": true }] }</pre>
不存在	年龄字段不存在	<pre>{ "Age": [{ "exists": false }] }</pre>
开头为前缀	地区位于美国	<pre>{ "Region": [{ "prefix": "us-" }] }</pre>
结尾为后缀	位置有后缀“West”	<pre>{ "Region": [{ "suffix": "West" }] }</pre>

细分规则示例

以下所有示例都假定您正在使用您在规则模式中使用的相同字段标签和值传递 `evaluationContext` 的值。

如果 `Browser` 为 `Chrome` 或 `Firefox` 且 `Location` 为美国西部，则以下示例匹配。

```
{
  "Browser": ["Chrome", "Firefox"],
  "Location": ["US-West"]
}
```

如果 `Browser` 是除 `Chrome` 以外的浏览器，`Location` 以 `US` 开头且 `Age` 字段存在，则以下示例匹配。

```
{
  "Browser": [ { "anything-but": ["Chrome"] } ],
  "Location": [ { "prefix": "US" } ],
  "Age": [ { "exists": true } ]
}
```

如果 `Location` 为日本且 `Browser` 为 `Safari` 或 `Device` 为平板设备，则以下示例匹配。

```
{
  "Location": ["Japan"],
  "$or": [
```

```
{ "Browser": [ "Safari" ] },  
  { "Device": [ "Tablet" ] }  
}
```

创建分段

创建细分后，您即可在任何项目的任何发布或实验中使用它。

创建分段

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 选择 Segments (细分) 选项卡。
4. 选择 Create segment (创建分段)。
5. 对于 Segment name (细分名称)，请输入用于标识此细分的名称。

或者，添加描述。

6. 对于 Segment pattern (细分模式)，请输入定义规则模式的 JSON 块。有关规则模式语法的更多信息，请参阅 [细分规则模式语法 \(p. 299\)](#)。

创建启动

要向指定百分比的用户公开新功能或更改，请创建启动。然后，在向所有用户推出该功能之前，您可以监控关键指标，例如页面加载时间和转化率。

必须先创建项目，然后才能添加启动。有关更多信息，请参阅 [创建新项目 \(p. 294\)](#)。

添加启动时，您可以使用已创建的功能，也可以在创建启动时创建新功能。

向项目添加启动

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 选择项目名称旁边的按钮，然后选择 Project actions (项目操作)、Create launch (创建启动)。
4. 对于 Launch name (启动名称)，输入用于标识此项目中此功能的名称。

您也可以添加描述。

5. 选择 Select from existing features (从现有功能中选择) 或者 Add new feature (添加新功能)。

如果您使用的是现有功能，请在 Feature name (功能名称) 下进行选择。

如果您选择 Add new feature (添加新功能)，请执行以下操作：

- a. 对于 Feature name (功能名称)，输入用于标识此项目中此功能的名称。您也可以添加描述。
- b. 对于 Feature variations (功能变体) 和 Variation type (变体类型)，请选择 Boolean、Long (长整数)、Double (双精度) 或者 String (字符串)。有关更多信息，请参阅 [变体类型 \(p. 299\)](#)。
- c. 最多可向功能添加五个变体。每个变体的 Value (值) 必须对您选择的 Variation type (变体类型) 有效。

指定其中一个变体作为默认变体。这是将与其他变体进行比较的基准，并且应作为现在向用户提供的变体。如果您停止实验，将向所有用户提供此默认变体。

- d. 请选择 Sample code (示例代码)。此代码示例显示了为设置变体并为其分配用户会话，您需要添加到应用程序中的内容。您可以在 JavaScript、Java 和 Python 之间选择代码。

您现在无需将代码添加到应用程序中，但必须在开始启动之前添加。

有关更多信息，请参阅[将代码添加到应用程序 \(p. 308\)](#)。

6. 对于 Launch configuration (启动配置)，请选择立即开始启动或者计划稍后开始启动。
7. (可选) 要为已定义的受众细分指定不同的流量拆分，而不是将用于一般受众的流量拆分，请选择 Add Segment Overrides (添加细分覆盖)。

在 Segment Overrides (细分覆盖) 中，选择一个细分并定义要用于该细分的流量拆分。

您可以选择定义更多细分来定义流量拆分，方法是选择 Add Segment Override (添加细分覆盖)。一次发布最多可以有六个细分覆盖。

有关更多信息，请参阅[使用细分来聚焦受众 \(p. 299\)](#)。

8. 对于 Traffic configuration (流量配置)，选择要分配给不匹配细分覆盖的普通受众的每个变化的流量百分比。您也可以选择不向用户提供变体。

Traffic summary (流量摘要) 显示了可用于本次启动的总流量。

9. 如果您选择计划稍后开始启动，则可以为启动添加多个步骤。每个步骤都可以使用不同的百分比来提供变体。为此，请选择 Add another step (添加其他步骤)，然后指定下一步的时间表和流量百分比。您可在启动中包含最多五个步骤。
10. 如果您想在启动期间使用指标跟踪功能性能，请依次选择 Metrics (指标)、Add metric (添加指标)。您可以使用 CloudWatch RUM 指标或自定义指标。

要使用自定义指标，您可以使用 Amazon EventBridge 规则在此处创建指标。要创建自定义指标，请执行以下操作：

- 选择 Custom metrics (自定义指标) 并输入指标名称。
- 对于 Entity ID (实体 ID)，请在 Metric rule (指标规则) 下，输入标识实体的方法。这可以是执行会导致记录指标值的操作的用户或会话。示例是 `userDetails.userID`。
- 对于 Value key (值键)，请输入要跟踪以生成指标的值。
- 您也可以选择输入指标的单位名称。这个用于 Evidently 控制台中图表的单位名称仅供显示之用。

输入这些字段时，此框会显示如何对 EventBridge 规则进行编码以创建指标的示例。有关 EventBridge 的更多信息，请参阅[什么是 Amazon EventBridge ?](#)

要使用 RUM 指标，您必须为应用程序设置 RUM 应用程序监控。有关更多信息，请参阅[设置应用程序以使用 CloudWatch RUM \(p. 325\)](#)。

Note

如果您使用 RUM 指标，并且应用程序监控未配置为对 100% 的用户会话进行采样，则并非所有参与启动的用户会话都会将指标发送给 Evidently。为确保启动指标准确无误，我们建议应用程序监控使用 100% 的用户会话进行采样。

11. (可选) 如果您为启动创建了至少一个指标，便可将现有 CloudWatch 告警与此启动相关联。为此，请选择 Associate CloudWatch alarms (关联 CloudWatch 告警)。

当您将告警与启动相关联时，CloudWatch Evidently 必须向告警添加带有项目名称和启动名称的标签。这样一来，CloudWatch Evidently 便可以在控制台的启动信息中显示正确的告警。

要确认 CloudWatch Evidently 会添加这些标签，请选择 Allow Evidently to tag the alarm resource identified below with this launch resource. (允许 Evidently 用此启动资源标记以下标识的告警资源。) 然后，选择 Associate alarm (关联告警)，输入告警名称。

有关创建 CloudWatch 告警的信息，请参阅 [使用 Amazon CloudWatch 告警 \(p. 151\)](#)。

12. (可选) 要向此启动添加标签，请选择 Tags (标签)、Add new tag (添加新标签)。

然后，对于 Key (键)，输入标签的名称。您可以在 Value (值) 中添加可选的标签值。

要添加其他标签，请再次选择 Add new tag (添加新标签)。

有关更多信息，请参阅[标记 Amazon 资源](#)。

13. 选择 Create launch (创建启动)。

创建实验

使用实验来测试功能或网站的不同版本，并从真实用户会话中收集数据。这样，您便可以根据证据和数据为应用程序做出选择。

在添加实验之前，您必须创建项目。有关更多信息，请参阅[创建新项目 \(p. 294\)](#)。

添加实验时，您可以使用已创建的功能，或在创建实验时创建新功能。

向项目添加实验

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 选择项目名称旁边的按钮，然后选择 Project actions (项目操作)、Create experiment (创建实验)。
4. 对于 Experiment name (实验名称)，输入用于标识此项目中此功能的名称。

您也可以添加描述。

5. 选择 Select from existing features (从现有功能中选择) 或者 Add new feature (添加新功能)。

如果您使用的是现有功能，请在 Feature name (功能名称) 下进行选择。

如果您选择 Add new feature (添加新功能)，请执行以下操作：

- a. 对于 Feature name (功能名称)，输入用于标识此项目中此功能的名称。您也可以选择输入描述。
- b. 对于 Feature variations (功能变体) 和 Variation type (变体类型)，请选择 Boolean、Long (长整数)、Double (双精度) 或者 String (字符串)。该类型定义了每个变体使用哪种类型的值。有关更多信息，请参阅[变体类型 \(p. 299\)](#)。
- c. 最多可向功能添加五个变体。每个变体的 Value (值) 必须对您选择的 Variation type (变体类型) 有效。

指定其中一个变体作为默认变体。这是将与其他变体进行比较的基准，并且应作为现在向用户提供的变体。如果您停止使用此功能的实验，则将向之前参加实验的用户百分比提供该默认变体。

- d. 请选择 Sample code (示例代码)。此代码示例显示了为设置变体并为其分配用户会话，您需要添加到应用程序中的内容。您可以在 JavaScript、Java 和 Python 之间选择代码。

您现在无需将代码添加到应用程序中，但必须在开始实验之前添加。有关更多信息，请参阅[将代码添加到应用程序 \(p. 308\)](#)。

6. 对于 Audience (受众)，如果您希望此实验仅应用于与该细分匹配的用户，则可以选择您已创建的细分。有关细分的更多信息，请参阅[使用细分来聚焦受众 \(p. 299\)](#)。
7. 对于 Traffic split for the experiment (实验的流量拆分)，指定将在实验中使用会话的选定受众的百分比。然后为实验使用的不同变体分配流量。

如果同一功能同时运行启动和实验，会将受众首先引至启动。然后，总体受众会产生为启动指定的流量百分比。您在此处指定的百分比是用于实验的剩余受众的百分比。将向之后的任何剩余流量提供默认变体。

8. 对于 Metrics (指标)，请选择用于评估实验期间变体的指标。必须使用至少一个指标进行评估。

- a. 对于 Metric source (指标来源) , 请选择使用 CloudWatch RUM 指标或自定义指标。
- b. 输入指标的名称。对于 Goal (目标) , 如果您希望表示变体更好的指标值较高, 请选择 Increase (增加) 。如果您希望表示变体更好的指标值较低, 请选择 Decrease (减少) 。
- c. 如果您正在使用自定义指标, 可以使用 Amazon EventBridge 规则在此创建指标。要创建自定义指标, 请执行以下操作:
 - 对于 Entity ID (实体 ID) , 请在 Metric rule (指标规则) 下输入标识实体的方法, 这可以是执行导致记录指标值的操作的用户或会话。示例是 `userDetails.userID` 。
 - 对于 Value key (值键) , 请输入要跟踪以生成指标的值。
 - 您也可以选择输入指标的单位名称。这个用于 Evidently 控制台中图表的单位名称仅供显示之用。

仅当已设置 RUM 以监控此应用程序时, 才能使用 RUM 指标。有关更多信息, 请参阅[使用 CloudWatch Rum \(p. 323\)](#)。

Note

如果您使用 RUM 指标, 并且应用程序监控未配置为对 100% 的用户会话进行采样, 则并非实验中的所有用户会话都会将指标发送给 Evidently。为确保实验指标准确无误, 我们建议应用程序监控使用 100% 的用户会话进行采样。

- d. (可选) 要添加更多要评估的指标, 请选择 Add metric (添加指标) 。在实验期间, 您可以评估多达三个指标。
9. (可选) 要创建用于此实验的 CloudWatch 告警, 请选择 CloudWatch alarms (CloudWatch 告警) 。告警可以监控每个变体与默认变体之间的结果差异是否大于您指定的阈值。如果变体的性能比默认变体差, 且差异大于阈值, 便会进入告警状态并通知您。

在此处创建告警意味着为每个非默认变体的变体创建告警。

如果您创建告警, 请指定以下内容:

- 对于 Metric name (指标名称) , 请选择要用于告警的实验指标。
- 对于 Alarm condition (告警状态) , 当变体指标值与默认变体指标值进行比较时, 请选择引发告警进入告警状态的条件。例如, 如果数字较大表示变体的性能不佳, 请选择 Greater (更大) 或 Greater/Equal (更大/等于) 。例如, 如果指标用于测量页面加载时间, 此选项会非常合适。
- 输入阈值数字, 这是会引发告警进入 ALARM 状态的性能差异百分比。
- 对于 Average over period (期间的平均值) , 选择比较之前每个变体的指标数据聚合数量。

您可以再次选择 Add new alarm (添加新告警) , 以向实验添加更多告警。

下一步, 选择 Set notifications for the alarm (设置告警通知) , 然后选择或创建用于向其发送告警通知的 Amazon Simple Notification Service 主题。有关更多信息, 请参阅[设置 Amazon SNS 通知 \(p. 157\)](#)。

10. (可选) 要向此实验添加标签, 请选择 Tags (标签) 、Add new tag (添加新标签) 。

然后, 对于 Key (键) , 输入标签的名称。您可以在 Value (值) 中添加可选的标签值。

要添加其他标签, 请再次选择 Add new tag (添加新标签) 。

有关更多信息, 请参阅[标记 Amazon 资源](#)。

11. 选择 Create experiment (创建实验) 。
12. 如果您尚未执行, 请将功能变体构建到应用程序中。
13. 选择完成。只有您将其启动, 实验才会开始。

完成以下过程中的步骤后, 实验会立即开始。

开始进行已创建的实验

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 请选择项目名称。
4. 请选择 Experiments (实验) 选项卡。
5. 选择实验名称旁边的按钮，然后依次选择 Actions (操作)、Start experiment (开始实验)。
6. (可选) 要查看或修改创建实验设置时指定的实验设置，请选择 Experiment setup (实验设置)。
7. 请选择实验结束的时间。
8. 请选择 Start experiment (开始实验)。

实验会立即开始。

管理功能、启动和实验

使用这些部分中的步骤来管理您已创建的功能、启动和实验。

主题

- [查看功能的当前评估规则和受众流量 \(p. 306\)](#)
- [修改启动流量 \(p. 307\)](#)
- [修改启动的未来步骤 \(p. 307\)](#)
- [修改实验流量 \(p. 307\)](#)
- [停止启动 \(p. 308\)](#)
- [停止实验 \(p. 308\)](#)

查看功能的当前评估规则和受众流量

您可以使用 CloudWatch Evidently 控制台，了解该功能的评估规则如何在功能的当前启动、实验和变体之间分配受众流量。

查看功能的受众流量

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 选择包含该功能的项目名称。
4. 选择 Features (功能) 选项卡。
5. 选择功能的名称。

在 Evaluation rules (评估规则) 选项卡中，您可以查看功能的受众流量流动，如下所示：

- 首先，对覆盖进行评估。这意味着会始终为某些用户提供特定变体。已分配覆盖的用户的会话无益于启动或试验指标。
- 然后，剩余流量可用于持续启动 (如有)。如果启动正在进行，Launches (启动) 部分中的表格会显示功能变体中的启动名称和启动流量拆分。在 Launches (启动) 部分的右侧，Traffic (流量) 指示器会显示分配给此启动的可用受众 (覆盖后) 的数量。未分配给此启动的其余流量会流向实验 (如有)，然后流向默认变体。
- 然后，剩余流量可用于正在进行的实验 (如有)。如果有实验正在进行，Experiments (实验) 部分中的表格会显示实验名称和进度。在 Experiments (实验) 部分的右侧，Traffic (流量) 指示器会显示分配给此实验的可用受众 (覆盖和启动后) 的数量。将向未分配给启动或实验的其余流量提供功能的默认变体。

修改启动流量

您可以随时（包括启动期间）修改启动的流量分配。

如果您对同一功能同时进行启动和实验，则对功能流量的任何更改都会导致实验流量发生变化。这是因为可用于实验的受众是尚未分配给启动的总受众的一部分。增加启动流量会导致可用于实验的受众减少，减少启动流量或结束启动则会导致可用于实验的受众增加。

修改启动的流量分配

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring（监控应用程序）、Evidently。
3. 请选择包含启动的项目名称。
4. 请选择 Launches（启动）选项卡。
5. 请选择启动的名称。

请选择 Modify launch traffic（修改启动流量）。

6. 对于 Serve（服务），选择要分配给每个变体的新流量百分比。您也可以选择不向用户提供变体。当您更改这些值时，您可以在 Traffic summary（流量摘要）下查看对整体功能流量的更新影响。

Traffic summary（流量摘要）显示了可用于此次启动的总流量数量，以及其中可分配给此次启动的数量。

7. 选择 Modify（修改）。

修改启动的未来步骤

您可以修改尚未进行的启动步骤配置，还可以为启动添加更多步骤。

修改启动步骤

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring（监控应用程序）、Evidently。
3. 请选择包含启动的项目名称。
4. 请选择 Launches（启动）选项卡。
5. 请选择启动的名称。

请选择 Modify launch traffic（修改启动流量）。

6. 选择 Schedule launch（计划启动）。
7. 对于尚未开始的任何步骤，您可以修改在实验中使用的可用受众百分比。您还可以修改其流量在变体中的分配方式。

您可以通过选择 Add another step（添加其他步骤）为启动添加更多步骤。启动可具有最多五个步骤。

8. 选择 Modify（修改）。

修改实验流量

您可以随时（包括实验进行期间）修改实验的流量分配。如果您修改正在进行的实验的流量，我们建议您只增加流量分配，以免产生偏差。

修改实验的流量分配

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。

2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 请选择包含启动的项目名称。
4. 请选择 Experiments (实验) 选项卡。
5. 请选择启动的名称。
6. 选择 Modify experiment traffic (修改实验流量)。
7. 输入百分比或使用滑块，指定向此实验分配的可用流量。可用流量为总受众减去分配给当前启动的流量 (如有)。将向未分配给启动或实验的流量提供默认变体。
8. 选择 Modify (修改)。

停止启动

如果停止正在进行的启动，您将无法恢复或重启。此外，不会将其作为流量分配的规则进行评估，并且分配给此次启动的流量可用于功能的实验 (如有)。否则，在启动停止后，将向所有流量提供默认变体。

永久停止启动

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 请选择包含启动的项目名称。
4. 请选择 Launch (启动) 选项卡。
5. 选择启动名称左侧的按钮。
6. 选择 Actions (操作)、Cancel launch (取消启动) 或 Actions (操作)、Mark as complete (标记为完成)。

停止实验

如果您停止正在进行的实验，将无法恢复或重启。将向之前在实验中使用的流量部分提供默认变体。

如果试验未被手动停止并且超过结束日期，流量不会改变。分配给实验的流量部分仍然用于实验。要停止这种情况，并向实验流量提供默认变体，请将实验标记为完成。

如果您想停止实验，可以选择将其取消或将其标记为完成。如果将其取消，将在实验列表中显示 Cancelled (已取消)。如果选择将其标记为完成，将显示 Completed (已完成)。

永久停止实验

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 请选择包含此实验的项目名称。
4. 请选择 Experiments (实验) 选项卡。
5. 请选择实验名称左侧的按钮。
6. 请选择 Actions (操作)、Cancel experiment (取消实验) 或 Actions (操作)、Mark as complete (标记为完成)。

将代码添加到应用程序

要使用 CloudWatch Evidently，您可以将代码添加到应用程序，以向每个用户会话分配变体，并将指标发送到 Evidently。使用 CloudWatch Evidently `EvaluateFeature` 操作将变体分配给用户会话，然后使用 `PutProjectEvents` 操作将事件发送到 Evidently，用于计算启动或实验的指标。

当您创建变体或自定义指标时，CloudWatch Evidently 控制台会提供需要添加的代码示例。

有关端到端示例的信息，请参阅 [教程：使用 Evidently 示例应用程序进行 A/B 测试 \(p. 317\)](#)。

使用 EvaluateFeature

在启动或实验中使用功能变体时，应用程序会使用 [EvaluateFeature](#) 操作为每个用户会话分配一个变体。将变体分配给用户即为评估事件。如果您调用此操作，会传递以下内容：

- Feature name (功能名称) – 必填。Evidently 会根据启动或实验的功能评估规则进行评估，并为实体选择变体。
- entityId – 必填。表示唯一的用户。
- evaluationContext – 可选。一个 JSON 对象，表示有关用户的其他信息。如果您已经创建了细分，Evidently 会使用此值在功能评估期间将用户与受众细分匹配。有关更多信息，请参阅[使用细分来聚焦受众 \(p. 299\)](#)。

下面是您可以发送至 Evidently 的 evaluationContext 值的示例。

```
{
  "Browser": "Chrome",
  "Location": {
    "Country": "United States",
    "Zipcode": 98007
  }
}
```

粘性评估

CloudWatch Evidently 使用“粘性”评估。entityId 的单个配置、功能、功能配置 和 evaluationContext 始终会接收相同的变体分配。此变体分配仅在实体添加到覆盖或实验流量增加时更改。

功能配置包括以下内容：

- 功能变体
- 此功能当前运行的实验的变体配置 (分配给每个变体的百分比) (如果有)。
- 此功能当前运行的启动的变体配置 (如果有)。变体配置包括定义的区段覆盖 (如果有)。

使用 PutProjectEvents

要为 Evidently 编写自定义指标，请使用 [PutProjectEvents](#) 操作。以下是一个简单的有效负载示例。

```
{
  "events": [
    {
      "timestamp": {{$timestamp}},
      "type": "aws.evidently.custom",
      "data": "{\"details\": {\"pageLoadTime\": 800.0}, \"userDetails\": {\"userId\": \"test-user\"}}\"
    }
  ]
}
```

entityIdKey 可以只是 entityId，您也可以将其重命名为其他任何名称，例如 userId。在实际事件中，entityId 可以是用户名、会话 ID 等等。


```
"metricDefinition":{
  "name": "noFilter",
  "entityIdKey": "userDetails.userId", //should be consistent with jsonValue in events
  "data" fields
  "valueKey": "details.pageLoadTime"
},
```

为了确保事件与正确的启动或实验相关联，您必须在调用 `EvaluateFeature` 和 `PutProjectEvents` 时传递相同的 `entityId`。请务必在调用 `EvaluateFeature` 之后调用 `PutProjectEvents`，否则数据会丢失，并且 CloudWatch Evidently 不会使用。

`PutProjectEvents` 操作不要求将功能名称作为输入参数。这样一来，您可在多个实验中使用一个事件。例如，假设您调用 `EvaluateFeature` 并将 `entityId` 设置为 `userDetails.userId`。如果您有两个或更多运行中的实验，则可以从该用户的会话中获得单个事件，并为每个这些实验发出指标。为此，使用相同的 `entityId` 为每个实验调用一次 `PutProjectEvents`。

Timing

在应用程序调用 `EvaluateFeature` 之后，有一个小时的时间段，其中根据该评估限制来自 `PutProjectEvents` 的指标事件。如果在一小时之后发生了更多的事件，则不会限制这些事件。

但是，如果在初始调用的一小时时段内将相同的 `entityId` 用于新 `EvaluateFeature` 调用，则现在改为使用后一个 `EvaluateFeature` 的结果，并重新启动一个小时的计时器。仅在某些情况下才会发生此操作，例如在两此分配之间增加实验流量时，如前面的粘性评估部分所解释。

有关端到端示例的信息，请参阅 [教程：使用 Evidently 示例应用程序进行 A/B 测试 \(p. 317\)](#)。

项目数据存储

Evidently 收集两种类型的事件：

- 评估事件与分配给用户会话的功能变体有关。Evidently 使用这些事件，生成可在 Evidently 控制台中查看的指标与其他实验和启动数据。

您也可以选择将这些评估事件存储在 Amazon CloudWatch Logs 或 Amazon S3 中。

- 自定义事件用于通过单击和结算等用户操作生成指标。Evidently 不会提供存储自定义事件的方法。如果要将其保存，您必须修改应用程序代码以将其发送到 Evidently 之外的存储选项。

评估事件日志的格式

如果您选择将评估事件存储在 CloudWatch Logs 或 Amazon S3 中，则每个评估事件都将以下列格式存储为日志事件：

```
{
  "event_timestamp": 1642624900215,
  "event_type": "evaluation",
  "version": "1.0.0",
  "project_arn": "arn:aws:evidently:us-east-1:123456789012:project/petfood",
  "feature": "petfood-upsell-text",
  "variation": "Variation1",
  "entity_id": "7",
  "entity_attributes": {},
  "evaluation_type": "EXPERIMENT_RULE_MATCH",
  "treatment": "Variation1",
  "experiment": "petfood-experiment-2"
}
```

以下是关于上述评估事件格式的更多信息：

- 时间戳为以毫秒为单位的 UNIX 时间
- 此变体是分配给此用户会话的功能变体的名称。
- 实体 ID 是字符串。
- 实体属性是客户端发送的任意值的哈希值。例如，如果 `entityId` 映射为蓝色或绿色，然后您可以选择从关联和数据仓库的角度选择性地发送用户 ID、会话数据或任何您想要的其它内容。

用于在 Amazon S3 中存储评估事件的 IAM policy 和加密

如果您选择使用 Amazon S3 存储评估事件，则必须添加类似以下内容的 IAM policy，以允许 Evidently 将日志发布到 Amazon S3 存储桶。这是因为 Amazon S3 存储桶及其包含的对象是私有的，默认情况下它们不允许访问其他服务。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {"Service": "delivery.logs.amazonaws.com"},
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::bucket_name/optional_folder/AWSLogs/account_id/*",
      "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-control"}}
    },
    {
      "Sid": "AWSLogDeliveryCheck",
      "Effect": "Allow",
      "Principal": {"Service": "delivery.logs.amazonaws.com"},
      "Action": ["s3:GetBucketAcl", "s3:ListBucket"],
      "Resource": "arn:aws:s3:::bucket_name"
    }
  ]
}
```

如果您将 Evidently 数据存储到 Amazon S3 中，您还可以选择使用带 Amazon Key Management Service 密钥的服务器端加密 (SSE-KMS)，对其进行加密。有关更多信息，请参阅[使用服务器端加密保护数据](#)。

如果您使用 Amazon KMS 的客户托管式密钥，您必须将以下内容添加到密钥的 IAM policy 中。这样一来，Evidently 便可以写入存储桶。

```
{
  "Sid": "AllowEvidentlyToUseCustomerManagedKey",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "delivery.logs.amazonaws.com"
    ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

Evidently 是如何计算结果的

您可以使用 Amazon CloudWatch Evidently A/B 测试作为数据驱动型决策的工具。在 A/B 测试中，用户被随机分配到对照组（也称为默认变体）或其中一个治疗组（也称为测试变体）。例如，对照组中的用户体验网站、服务或应用程序的方式可能与实验开始之前相同。同时，治疗组中的用户可能会经历这种变化。

CloudWatch Evidently 在一项实验中最多支持五种不同的变体。Evidently 将流量随机分配给这些变体。这样，您可以跟踪每个组的业务指标（例如收入）和绩效指标（例如延迟）。Evidently 可执行以下操作：

- 将治疗与对照进行比较。（例如，通过新的结账流程比较收入是增加还是减少。）
- 表明观察到的治疗与对照之间的差异是否为显著。为此，Evidently 提供了两种方法：频率论显著性水平和贝叶斯概率。

为什么要使用频率论方法和贝叶斯方法？

考虑治疗与对照组相比没有效果的病例，或者治疗与对照组相同的病例（A/A 测试）。您仍然可以在数据中观察到治疗与对照之间存在细微差异。这是因为测试参加者由有限的用户样本组成，占网站、服务或应用程序所有用户的一小部分。通过频率论显著性水平和贝叶斯概率可以洞察观察到的差异是显著的还是偶然的。

Evidently 考虑以下因素来确定观察到的差异是否显著：

- 差异有多大
- 测试中有多少样本
- 数据是如何分布的

Evidently 中的频率论分析

Evidently 使用顺序测试，这样可以避免常见的窥视问题，这是频率论统计方法的常见陷阱。窥视是检查正在进行的 A/B 测试结果的一种做法，以便停止测试并根据观察到的结果做出决定。有关顺序测试的更多信息，请参阅欧几里德项目中的[时间均匀、非参数、非渐近置信序列](#)。

因为 Evidently 的结果在任何时候都是有效的（随时有效结果），您可以在实验过程中窥视结果，且仍然可以得出合理的结论。这可以降低一些实验成本，因为如果实验结果已经具有显著性，则可以在预定时间之前停止实验。

Evidently 可以生成随时有效的显著性水平，以及目标指标中测试变体与默认变体之间差异的 95% 置信区间。实验结果中的 Result（结果）列表示测试的变体性能，包括以下各项之一：

- Inconclusive（不确定）– 显著性水平低于 95%
- Better（更好）– 显著性水平为 95% 或以上，出现以下情况之一：
 - 95% 置信区间的下限高于零，指标应增加
 - 95% 置信区间的上限低于零，指标应减小
- Worse（更差）– 显著性水平为 95% 或以上，出现以下情况之一：
 - 95% 置信区间的上限高于零，指标应增加
 - 95% 置信区间的下限低于零，指标应减小
- Best（最好）– 除了默认变体外，实验还有两个或多个经过测试的变体，并且满足以下条件：
 - 该变体符合 Better（更好）名称条件
 - 满足以下条件：
 - 95% 置信区间的下限高于所有其他变体的 95% 置信区间的上限，指标应增加
 - 95% 置信区间的上限低于所有其他变体的 95% 置信区间的下限，指标应减小

Evidently 中的贝叶斯分析

使用贝叶斯分析，您可以计算出测试变体中的均值大于或小于默认变体中的均值的概率。Evidently 使用共轭先验对目标指标的平均值进行贝叶斯推断。使用共轭先验，Evidently 可以更有效地推断出贝叶斯分析所需的后验分布。

Evidently 要等到实验结束日期才计算贝叶斯分析的结果。结果页显示以下内容：

- 增加的概率 – 测试变体中指标的均值比默认变体中的均值至少大 3% 的概率
- 减小的概率 – 测试变体中指标的均值比默认变体中的均值至少小 3% 的概率
- 不变概率 – 测试变体中指标的均值在默认变体中的均值的 $\pm 3\%$ 内的概率

Result (结果) 列表示变体性能，可以是以下各项之一：

- Better (更好) – 增加概率至少为 90%，指标应增加，或者减小概率至少为 90%，指标应降低
- Worse (更差) – 减小概率至少为 90%，指标应增加，或者增加概率至少为 90%，指标应减小

在控制面板中查看启动结果

在实验进行期间和实验完成后，您可以查看实验的进度和指标结果。

查看启动的进度和结果

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 请选择包含启动的项目名称。
4. 请选择 Launch (启动) 选项卡。
5. 请选择启动的名称。
6. 要查看启动步骤和每个步骤的流量分配，请选择 Launch (启动) 选项卡。
7. 要查看一段时间内分配给每个变体的用户会话数，并查看启动中每个变体的性能指标，请选择 Monitoring (监控) 选项卡。

此视图还会显示在启动期间是否有任何启动告警进入 ALARM 状态。

8. 要查看此次启动的变体、指标、告警和标签，请选择 Configuration (配置) 选项卡。

在控制面板中查看实验结果

您可以在实验进行期间和实验完成后，查看实验的统计结果。实验结果在实验开始后 63 天内可用。由于 CloudWatch 数据保留策略，在此之后实验结果将不可用。

每个变体至少有 100 个事件，才会显示统计结果。

Evidently 在实验结束时执行额外的离线 p 值分析。在实验期间使用的任何 p 值找不到统计显著性的某些情况下，离线 p 值分析可以检测统计显著性。

有关 CloudWatch Evidently 如何计算实验结果的更多信息，请参阅 [Evidently 是如何计算结果的 \(p. 312\)](#)。

查看实验结果

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。

3. 请选择包含此实验的项目名称。
4. 请选择 Experiments (实验) 选项卡。
5. 请选择实验名称，然后选择 Results (结果) 选项卡。
6. 通过 Variation performance (变体性能)，您可以使用一个控件来选择要显示的实验统计数据。如果您选择多个统计数据，Evidently 会显示每个统计数据的图表和表格。

每个图表和表格都会显示迄今为止的实验结果。

每个图表都会显示以下结果。您可以使用图表右侧的控件来确定显示以下哪些项目：

- 为每个变体记录的用户会话事件数。
- 在图表顶部选择的每个变体的指标平均值。
- 实验的统计意义。这会将图表顶部所选指标的差异与默认变体和其他每个变体进行比较。
- 每个变体与默认变体之间所选指标的差异，存在 95% 的置信上限和下限。

此表会显示每个变体的一行。对于每个非默认变体，Evidently 会显示其是否收到足够的数据以表明结果具有统计意义。它还会显示变体的统计值改进是否已达到 95% 的置信水平。

最后，在 Result (结果) 列中，Evidently 会根据此统计数据就哪个变体性能最佳或者结果是否不确定提出建议。

CloudWatch Evidently 如何收集和存储数据

Amazon CloudWatch Evidently 会收集和存储与项目配置相关的数据，以便客户运行实验和启动。该数据包括以下内容：

- 有关项目、功能、启动和实验的元数据
- 指标事件
- 评估数据

资源元数据存储在 Amazon DynamoDB 中。默认情况下，数据将使用 Amazon 拥有的密钥进行静态加密。这些密钥是 Amazon Web Service 拥有并管理以用于多个 Amazon Web Services 账户的 Amazon KMS 密钥的集合。客户无法查看、管理或审计这些密钥的使用情况。客户也无需采取行动或更改计划以保护用于加密其数据的密钥。

有关更多信息，请参阅 Amazon Key Management Service 开发人员指南中的 [Amazon 拥有的密钥](#)。

Evidently 指标事件和评估事件会直接发送到客户的地址。

在轮换到摄取中心之前，Evidently 还会在 Amazon Elastic Container Service 主机中暂时创建并存储评估日志。这些日志目前可保留最多三秒钟，并会在处理后删除。传输中的数据将使用 HTTPS 自动加密。这些数据将传送到客户的地址。

您也可以选择将评估事件存储在 Amazon Simple Storage Service 或 Amazon CloudWatch Logs 中。有关如何保护这些服务中的数据的更多信息，请参阅[启用 Amazon S3 默认存储桶加密](#)和[使用 Amazon KMS 对 CloudWatch Logs 中的日志数据进行加密](#)。

检索数据

您可以使用 CloudWatch Evidently API 检索数据。要检索项目数据，请使用 [GetProject](#) 或 [ListProjects](#)。

要检索功能数据，请使用 [GetFeature](#) 或 [ListFeatures](#)。

要检索启动数据，请使用 [GetLaunch](#) 或 [ListLaunches](#)。

要检索实验数据，请使用 [GetExperiment](#)、[ListExperiments](#) 或者 [GetExperimentResults](#)。

修改和删除数据

您可以使用 CloudWatch Evidently API 修改和删除数据。对于项目数据，请使用 [UpdateProject](#) 或 [DeleteProject](#)。

对于功能数据，请使用 [UpdateFeature](#) 或 [DeleteFeature](#)。

对于启动数据，请使用 [UpdateLaunch](#) 或 [DeleteLaunch](#)。

对于实验数据，请使用 [UpdateExperiment](#) 或 [DeleteExperiment](#)。

使用 Evidently 的服务相关角色

CloudWatch Evidently 使用 Amazon Identity and Access Management (IAM) [服务相关角色](#)。服务相关角色是一种独特类型的 IAM 角色，它与 Evidently 直接相关。服务相关角色由 Evidently 预定义，并包含服务代表您调用其他 Amazon 服务所需的所有权限。

服务相关角色可让您更轻松设置 Evidently，因为您不必手动添加必要的权限。Evidently 定义其服务相关角色的权限，除非另外定义，否则只有 Evidently 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

只有在首先删除服务相关角色的相关资源后，才能删除该角色。这将保护您的 Evidently 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅[与 IAM 配合使用的 Amazon 服务](#)，并查找 Service-linked roles (服务相关角色) 列中显示为 Yes (是) 的服务。请选择 Yes 与查看该服务的[服务相关角色文档](#)的链接。

Evidently 的服务相关角色权限

Evidently 使用名为 `AWSServiceRoleForCloudWatchEvidently` 的服务相关角色 — 允许 CloudWatch Evidently 代表您的客户管理关联的 Amazon 资源。

`AWSServiceRoleForCloudWatchEvidently` 服务相关角色信任以下服务代入角色：

- CloudWatch Evidently

名为 `AmazonCloudWatchEvidentlyServiceRolePolicy` 的角色权限策略允许 Evidently 对指定资源完成以下操作：

- 操作：对 Evidently 胖客户端进行 `appconfig:StartDeployment`、`appconfig:StopDeployment`、`appconfig:ListDeployments` 和 `appconfig:TagResource`。

必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅 IAM 用户指南中的[服务相关角色权限](#)。

创建 Evidently 的服务相关角色

无需手动创建服务相关角色。当您在 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 中开始使用 Evidently 胖客户端时，Evidently 会为您创建服务相关角色。

如果删除此服务相关角色，然后需要再次创建，可以使用相同流程在账户中重新创建此角色。当您开始使用 Evidently 胖客户端时，Evidently 会再次为您创建服务相关角色。

编辑 Evidently 的服务相关角色

Evidently 不允许您编辑 `AWSServiceRoleForCloudWatchEvidently` 服务相关角色。创建服务相关角色后，将无法更改角色名称，因为可能有多个实体引用该角色。但是可以使用 IAM 编辑角色说明。有关更多信息，请参阅 IAM 用户指南中的[编辑服务相关角色](#)。

删除 Evidently 的服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除服务相关角色的资源，然后才能手动删除它。您必须删除所有使用胖客户端的 Evidently 项目。

Note

如果在您试图删除资源时 Evidently 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

要删除 `AWSServiceRoleForCloudWatchEvidently` 使用的 Evidently 资源

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 在项目列表中，选中使用胖客户端的项目旁边的复选框。
4. 选择 Project actions (项目操作)、Delete project (删除项目)。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台、Amazon CLI 或 Amazon API 删除 `AWSServiceRoleForCloudWatchEvidently` 服务相关角色。有关更多信息，请参阅 IAM 用户指南中的[删除服务相关角色](#)。

Evidently 服务相关角色的受支持区域

Evidently 支持在服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅 [Amazon 区域和终端节点](#)。

CloudWatch Evidently 配额

CloudWatch Evidently 具有以下配额。

资源	默认配额
项目	每账户每区域 50 个 您可以请求提高配额。
分段	每账户每区域 500 个 您可以请求提高配额。
每个项目的配额	• 共 100 个功能 • 共 500 个启动 • 50 个正在运行的启动 • 共 500 个实验 • 50 个正在运行的实验 对于所有这些配额，您可以请求增加配额。

资源	默认配额
API 配额 (所有配额均针对每个区域)	- PutProjectEvents：在美国东部 (弗吉尼亚州北部)、美国西部 (俄勒冈州) 和欧洲 (爱尔兰)，每秒 1000 个事务 (TPS)。所有其他区域为 200 TPS。 - EvaluateFeature：在美国东部 (弗吉尼亚州北部)、美国西部 (俄勒冈州) 和欧洲 (爱尔兰)，1000 TPS。所有其他区域为 200 TPS。 - BatchEvaluateFeature：50 TPS - 创建、读取、更新、删除 (CRUD) API：在所有 CRUD API 中组合 10 TPS 对于所有这些配额，您可以请求增加配额。

教程：使用 Evidently 示例应用程序进行 A/B 测试

本节介绍如何使用 Amazon CloudWatch Evidently 进行 A/B 测试的教程。本教程涉及 Evidently 示例应用程序，其是一个简单的响应应用程序。示例应用程序将被配置为显示/不显示 showDiscount 功能。当向用户显示该功能时，购物网站上以 20% 的折扣显示价格。

除了向一些用户而非向其他用户显示折扣之外，在本教程中，您还设置 Evidently 以从两个变体收集页面加载时间指标。

步骤 1：下载示例应用程序

首先下载 Evidently 示例应用程序。

下载示例应用程序

1. 从以下 Simple Storage Service (Amazon S3) 存储桶下载示例应用程序：

```
https://evidently-sample-application.s3.us-west-2.amazonaws.com/evidently-sample-shopping-app.zip
```

2. 解压缩程序包。

步骤 2：添加 Evidently 端点并设置凭证

接下来，将 Evidently 的区域和端点添加到示例应用程序包中 src 目录下的 config.js 文件，如以下示例所示：

```
evidently: {  
  REGION: "us-west-2",  
  ENDPOINT: "https://evidently.us-west-2.amazonaws.com (https://evidently.us-west-2.amazonaws.com/)",  
},
```

您还必须确保应用程序有调用 CloudWatch Evidently 的权限。

授予示例应用程序调用 Evidently 的权限

1. 联合到您的 Amazon 账户。
2. 创建 IAM 用户并将 AmazonCloudWatchEvidentlyFullAccess 策略附加到该用户。

- 记录 IAM 用户的访问密钥 ID 和秘密访问密钥，因为您在下一步中需要使用它们。
- 在您本节早些时候修改的同一 `config.js` 文件中，输入访问密钥 ID 和秘密访问密钥的值，如以下示例所示：

```
credential: {
  accessKeyId: "Access key ID",
  secretAccessKey: "Secret key"
}
```

Important

我们使用此步骤使示例应用程序尽可能简单，以便您尝试。我们不建议您将 IAM 用户凭证放到实际的生产应用程序中。我们建议您使用 Amazon Cognito 进行身份验证。有关更多信息，请参阅[将 Amazon Cognito 与 Web 和移动应用程序集成](#)。

步骤 3：为功能评估设置代码

当您使用 CloudWatch Evidently 评估功能时，必须使用 `EvaluateFeature` 操作为每个用户会话随机选择功能变体。此操作根据您在实验中指定的百分比，将用户会话分配给功能的每个变体。

为书店演示应用程序设置功能评估代码

- 将客户端生成器添加到 `src/App.jsx` 文件，以便示例应用程序可以调用 Evidently。

```
import Evidently from 'aws-sdk/clients/evidently';
import config from './config';

const defaultClientBuilder = (
  endpoint,
  region,
) => {
  const credentials = {
    accessKeyId: config.credential.accessKeyId,
    secretAccessKey: config.credential.secretAccessKey
  }
  return new Evidently({
    endpoint,
    region,
    credentials,
  });
};
```

- 将以下内容添加到 `const App` 代码部分以启动客户端。

```
if (client == null) {
  client = defaultClientBuilder(
    config.evidently.ENDPOINT,
    config.evidently.REGION,
  );
}
```

- 通过添加以下代码构建 `evaluateFeatureRequest`。此代码会预填充我们在本教程后面推荐的项目名称和功能名称。您可以替换自己的项目名称和功能名称，只要您也可以在 Evidently 控制台中指定这些项目名称和功能名称。

```
const evaluateFeatureRequest = {
  entityId: id,
  // Input Your feature name
  feature: 'showDiscount',
  // Input Your project name'
```

```
    project: 'EvidentlySampleApp',  
  };
```

4. 添加代码以调用 Evidently 进行功能评估。发送请求时，Evidently 会随机分配查看或不查看 showDiscount 功能的用户会话。

```
client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {  
  if(res.value?.boolValue !== undefined) {  
    setShowDiscount(res.value.boolValue);  
  }  
  getPageLoadTime()  
})
```

步骤 4：为实验指标设置代码

对于自定义指标，请使用 Evidently 的 PutProjectEvents API 向 Evidently 发送指标结果。以下示例说明了如何设置自定义指标并向 Evidently 发送实验数据。

添加以下函数来计算页面加载时间，并使用 PutProjectEvents 向 Evidently 发送指标值。将以下函数添加到 Home.tsx 中并在 EvaluateFeature API 中调用此函数：

```
const getPageLoadTime = () => {  
  const timeSpent = (new Date().getTime() - startTime.getTime()) * 1.000001;  
  const pageLoadTimeData = `{  
    "details": {  
      "pageLoadTime": ${timeSpent}  
    },  
    "UserDetails": { "userId": "${id}", "sessionId": "${id}" }  
  }`;   
  const putProjectEventsRequest = {  
    project: 'EvidentlySampleApp',  
    events: [  
      {  
        timestamp: new Date(),  
        type: 'aws.evidently.custom',  
        data: JSON.parse(pageLoadTimeData)  
      },  
    ],  
  };  
  client.putProjectEvents(putProjectEventsRequest).promise();  
}
```

自您下载 App.js 文件并对其进行所有编辑之后，该文件应该如此处所示。

```
import React, { useEffect, useState } from "react";  
import { BrowserRouter as Router, Switch } from "react-router-dom";  
import AuthProvider from "contexts/auth";  
import CommonProvider from "contexts/common";  
import ProductsProvider from "contexts/products";  
import CartProvider from "contexts/cart";  
import CheckoutProvider from "contexts/checkout";  
import RouteWrapper from "layouts/RouteWrapper";  
import AuthLayout from "layouts/AuthLayout";  
import CommonLayout from "layouts/CommonLayout";  
import AuthPage from "pages/auth";  
import HomePage from "pages/home";  
import CheckoutPage from "pages/checkout";  
import "assets/scss/style.scss";  
import { Spinner } from 'react-bootstrap';
```

```
import Evidently from 'aws-sdk/clients/evidently';
import config from './config';

const defaultClientBuilder = (
  endpoint,
  region,
) => {
  const credentials = {
    accessKeyId: config.credential.accessKeyId,
    secretAccessKey: config.credential.secretAccessKey
  }
  return new Evidently({
    endpoint,
    region,
    credentials,
  });
};

const App = () => {
  const [isLoading, setIsLoading] = useState(true);
  const [startTime, setStartTime] = useState(new Date());
  const [showDiscount, setShowDiscount] = useState(false);
  let client = null;
  let id = null;

  useEffect(() => {
    id = new Date().getTime().toString();
    setStartTime(new Date());
    if (client == null) {
      client = defaultClientBuilder(
        config.evidently.ENDPOINT,
        config.evidently.REGION,
      );
    }
    const evaluateFeatureRequest = {
      entityId: id,
      // Input Your feature name
      feature: 'showDiscount',
      // Input Your project name
      project: 'EvidentlySampleApp',
    };

    // Launch
    client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
      if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
      }
    });

    // Experiment
    client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
      if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
      }
    })
    getPageLoadTime()
  })

  setIsLoading(false);
}, []);

const getPageLoadTime = () => {
  const timeSpent = (new Date().getTime() - startTime.getTime()) * 1.000001;
  const pageLoadTimeData = `{
    "details": {
      "pageLoadTime": ${timeSpent}
    }
  },`
```

```
      "UserDetails": { "userId": "${id}", "sessionId": "${id}" }
    };
    const putProjectEventsRequest = {
      project: 'EvidentlySampleApp',
      events: [
        {
          timestamp: new Date(),
          type: 'aws.evidently.custom',
          data: JSON.parse(pageLoadTimeData)
        },
      ],
    };
    client.putProjectEvents(putProjectEventsRequest).promise();
  }
  return (
    !isLoading? (
      <AuthProvider>
        <CommonProvider>
          <ProductsProvider>
            <CartProvider>
              <CheckoutProvider>
                <Router>
                  <Switch>
                    <RouteWrapper
                      path="/"
                      exact
                      component={() => <HomePage showDiscount={showDiscount}/>}
                      layout={CommonLayout}
                    />
                    <RouteWrapper
                      path="/checkout"
                      component={CheckoutPage}
                      layout={CommonLayout}
                    />
                    <RouteWrapper
                      path="/auth"
                      component={AuthPage}
                      layout={AuthLayout}
                    />
                  </Switch>
                </Router>
              </CheckoutProvider>
            </CartProvider>
          </ProductsProvider>
        </CommonProvider>
      </AuthProvider> ) : (
        <Spinner animation="border" />
      )
    );
  };
};

export default App;
```

每次用户访问示例应用程序时，都会向 Evidently 发送自定义指标进行分析。Evidently 会分析每个指标并在 Evidently 控制板上实时显示结果。以下示例显示了指标有效负载：

```
[ { "timestamp": 1637368646.468, "type": "aws.evidently.custom", "data": "{ \"details\": { \"pageLoadTime\": 2058.002058 }, \"userDetails\": { \"userId\": \"1637368644430\", \"sessionId\": \"1637368644430\" } }" } ]
```

步骤 5：创建项目、功能和实验

然后，您可以在 CloudWatch Evidently 控制台中创建项目、功能和实验。

为本教程创建项目、功能和实验

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 请选择 Create project (创建项目) 并填写字段。您必须使用 **EvidentlySampleApp** 作为项目名，以便示例正常工作。对于 Evaluation event storage (评估事件存储)，选择 Don't store Evaluation events (不要存储评估事件)。

填写字段后，选择 Create Project (创建项目)。

有关更多信息，请参阅 [创建新项目 \(p. 294\)](#)。

4. 创建项目后，在该项目中创建功能。将功能命名为 **showDiscount**。在此功能中，创建 **Boolean** 类型的两个变体。使用值 **disable** 将第一个变体命名为 **False**，然后使用值 **enable** 将第二个变体命名为 **True**。

有关创建功能的更多信息，请参阅 [向项目添加功能 \(p. 298\)](#)。

5. 完成功能创建后，请在项目中创建实验。将实验命名为 **pageLoadTime**。

本实验将使用名为 **pageLoadTime** 的自定义指标，用以衡量被测试页面的页面加载时间。用于实验的自定义指标使用 Amazon EventBridge 创建而成。有关 EventBridge 的更多信息，请参阅 [什么是 Amazon EventBridge ?](#)。

要创建该自定义指标，请在创建实验时执行以下操作：

- 对于 Metric source (指标来源)，请在 Metrics (指标) 下选择 Custom metrics (自定义指标)。
- 对于 Metric name (指标名称)，请输入 **pageLoadTime**。
- 对于 Goal (目标)，请选择 Decrease (减少)。这表示我们希望该指标的值较低，以指明该功能变体最佳。
- 对于 Metric rule (指标规则)，请输入以下内容：
 - 对于 Entity ID (实体 ID)，请输入 **UserDetails.userId**。
 - 对于 Value key (值键)，请输入 **details.pageLoadTime**。
 - 对于 Units (单位) 中，请输入 **ms**。
- 请选择 Add metric (添加指标)。

对于 Audiences (受众)，请选择 100% 以便所有用户都能进入实验。将变体之间的流量拆分设置为每个变体 50%。

然后，请选择 Create experiment (创建实验) 来创建实验。创建它后，只有您告知 Evidently 将其启动，它才会启动。

步骤 6：开始实验并测试 CloudWatch Evidently

最后的步骤是开始实验并开启示例应用程序。

开始教程实验

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，依次选择 Application monitoring (监控应用程序)、Evidently。
3. 选择 EvidentlySampleApp 项目。
4. 请选择 Experiments (实验) 选项卡。
5. 请选择 **pageLoadTime** 旁边的按钮，然后依次选择 Actions (操作)、Start experiment (开始实验)。
6. 请选择实验结束的时间。
7. 请选择 Start experiment (开始实验)。

实验会立即开始。

然后，使用以下命令开启 Evidently 示例应用程序：

```
npm install -f && npm start
```

开启应用程序后，您将被分配到正在测试的两个功能变体之一。一个变体会显示“20% 折扣”，另一个变体不会显示。继续刷新页面以查看不同的变体。

Note

Evidently 包含粘性评估。功能评估是确定性的，这意味着对于相同的 `entityId` 和功能，用户将始终收到相同的变体分配。变体分配仅在实体被添加到覆盖或实验流量增加时更改。

但是，为了让您轻松使用示例应用程序教程，Evidently 会在您每次刷新页面时重新分配示例应用程序功能评估，以便您在无需添加覆盖的情况下即可体验两种变体。

故障排除

我们建议您使用 npm 版本 6.14.14。如果你看到有关构建或开启示例应用程序的任何错误，并且您正在使用不同版本的 npm，请执行以下操作。

安装 npm 版本 6.14.14

1. 使用浏览器连接到 <https://nodejs.org/download/release/v14.17.5/>。
2. 下载 [node-v14.17.5.pkg](#) 并运行此 pkg 安装 npm。

如果您看到 `webpack not found` 错误，导航至 `evidently-sample-shopping-app` 文件夹并尝试以下操作：

- a. 删除 `package-lock.json`
- b. 删除 `yarn-lock.json`
- c. 删除 `node_modules`
- d. 从 `package.json` 中删除 webpack 依赖项
- e. 运行以下命令：

```
npm install -f && npm
```

使用 CloudWatch RUM

借助 CloudWatch RUM，可执行真实用户监控，从实际用户会话中近乎实时地收集和查看有关 Web 应用程序性能的客户端数据。可视化和可分析的数据包括页面加载时间、客户端错误和用户行为。当查看这些数据时，可以看到所有数据汇总在一起，也可以查看按客户使用的浏览器和设备划分的细分数据。

您可以使用收集的数据快速识别和调试客户端性能问题。CloudWatch RUM 可帮助您可视化应用程序性能中的异常，并查找相关的调试数据，例如错误消息、堆栈跟踪和用户会话。还可以使用 RUM 了解终端用户的影响范围，包括用户数量、地理位置和使用的浏览器。

为 CloudWatch RUM 收集的终端用户数据将保留 30 天，然后自动删除。如果要保留 RUM 事件更长时间，可以选择让应用程序监控将事件的副本发送到账户中的 CloudWatch Logs。然后，便可以调整该日志组的保留期。

要使用 RUM，需要创建应用程序监控并提供一些信息。RUM 会生成一个可粘贴到应用程序中的 JavaScript 代码段。该代码段拉入了 RUM Web 客户端代码。RUM Web 客户端从应用程序中一定百分比的用户会话中捕获数据，这些数据显示在预构建的控制面板中。您可以指定要从中收集数据的用户会话百分比。

RUM Web 客户端是开源的。有关更多信息，请参阅 [CloudWatch RUM Web 客户端](#)。

性能注意事项

本节讨论使用 CloudWatch RUM 时的性能注意事项。

- 负载性能影响 – CloudWatch RUM Web 客户端可作为 JavaScript 模块被安装到您的 Web 应用程序当中，或从内容分发网络（CDN）异步加载到 Web 应用程序。这不会阻止应用程序的加载过程。CloudWatch RUM 旨在不对应用程序的加载时间产生可感知的影响。
- 运行时影响 – RUM Web 客户端执行处理以记录 RUM 数据并将其分派到 CloudWatch RUM 服务。由于事件不频繁且处理量很小，因此 CloudWatch RUM 旨在不对应用程序的性能产生可检测的影响。
- 网络影响 – RUM Web 客户端定期向 CloudWatch RUM 服务发送数据。应用程序运行期间会定期分派数据，也会在浏览器卸载应用程序之前立即分派数据。浏览器卸载应用程序之前立即发送的数据将作为信标发送，该信标旨在不对应用程序的卸载时间产生可检测的影响。

RUM 定价

使用 CloudWatch RUM 时，CloudWatch RUM 收到的每个 RUM 事件可能都会收费。使用 RUM Web 客户端收集的每个数据项目都视为 RUM 事件。RUM 事件示例包括页面视图、JavaScript 错误和 HTTP 错误。您可以选择各应用程序监控收集的事件类型。您可以激活或停用选项来收集性能遥测事件、JavaScript 错误、HTTP 错误和 X-Ray 跟踪。有关选择这些选项的更多信息，请参阅 [步骤 2：创建应用程序监控 \(p. 327\)](#) 和 [CloudWatch RUM Web 客户端收集的信息 \(p. 335\)](#)。有关定价的更多信息，请参阅 [Amazon CloudWatch 定价](#)。

区域可用性

CloudWatch RUM 目前在以下区域中可用：

- 美国东部（弗吉尼亚州北部）
- 美国东部（俄亥俄州）
- 美国西部（俄勒冈州）
- 欧洲（法兰克福）
- 欧洲（斯德哥尔摩）
- 欧洲（爱尔兰）
- 欧洲（伦敦）
- 亚太地区（东京）
- 亚太地区（新加坡）
- 亚太地区（悉尼）

主题

- [使用 CloudWatch RUM 的 IAM 策略 \(p. 325\)](#)
- [设置应用程序以使用 CloudWatch RUM \(p. 325\)](#)
- [配置 CloudWatch RUM Web 客户端 \(p. 331\)](#)
- [使用页面组 \(p. 331\)](#)
- [查看 CloudWatch RUM 控制面板 \(p. 332\)](#)
- [您可以使用 CloudWatch RUM 收集的 CloudWatch 指标 \(p. 333\)](#)
- [CloudWatch RUM 的数据保护和数据隐私 \(p. 335\)](#)
- [CloudWatch RUM Web 客户端收集的信息 \(p. 335\)](#)
- [管理使用 CloudWatch RUM 的应用程序 \(p. 350\)](#)
- [CloudWatch RUM 配额 \(p. 351\)](#)
- [CloudWatch RUM 故障排除 \(p. 351\)](#)

使用 CloudWatch RUM 的 IAM 策略

为了完全管理 CloudWatch RUM，必须以具有 AmazonCloudWatchRUMFullAccess IAM policy 的 IAM 用户或角色身份登录。此外，可能需要其他策略或权限：

- 要创建应用程序监控来创建新的 Amazon Cognito 身份池进行授权，需要具备 Admin IAM 角色或 AdministratorAccess IAM policy。
- 要创建将数据发送到 CloudWatch Logs 的应用程序监控，必须登录具有以下权限的 IAM 角色或策略：

```
{
  "Effect": "Allow",
  "Action": [
    "logs:PutResourcePolicy"
  ],
  "Resource": [
    "*"
  ]
}
```

需要查看 CloudWatch RUM 数据但不需要创建 CloudWatch RUM 资源的其他用户，可以授予 AmazonCloudWatchRUMReadOnlyAccess 策略。

设置应用程序以使用 CloudWatch RUM

使用本章节中的步骤设置应用程序，以开始使用 CloudWatch RUM 从真实用户会话中收集性能数据。

主题

- [步骤 1：授权应用程序将数据发送到 Amazon \(p. 325\)](#)
- [步骤 2：创建应用程序监控 \(p. 327\)](#)
- (可选) [步骤 3：手动修改代码段以配置 CloudWatch RUM Web 客户端 \(p. 328\)](#)
- [步骤 4：将代码段插入应用程序 \(p. 329\)](#)
- [步骤 5：通过生成用户事件测试应用程序监控设置 \(p. 330\)](#)

步骤 1：授权应用程序将数据发送到 Amazon

要使用 CloudWatch RUM，应用程序必须获得授权。

您有三种设置授权的选项：

- 用 CloudWatch RUM 为应用程序创建新的 Amazon Cognito 身份池。这是最简单的设置方法。这是默认选项。

身份池将包含未经验证的身份。这允许 CloudWatch RUM Web 客户端将数据发送到 CloudWatch RUM，而无需对应用程序的用户进行身份验证。

Amazon Cognito 身份池具有附加的 IAM 角色。Amazon Cognito 未经验证的身份允许 Web 客户端担任 IAM 角色，此角色已获得授权，可向 CloudWatch RUM 发送数据。

- 使用现有的 Amazon Cognito 身份池。在这种情况下，还必须修改附加到身份池的 IAM 角色。
- 使用已设置的现有身份提供商的身份验证。在这种情况下，您必须从身份提供商处获取凭证，并且应用程序必须将这些凭证转发到 RUM Web 客户端。

以下各节包含了这些选项的更多详细信息。

CloudWatch RUM 创建新的 Amazon Cognito 身份池

这是最简单的设置选项，如果选择此选项，则无需其他设置步骤。您必须拥有管理权限才能使用此选项。有关更多信息，请参阅[使用 CloudWatch RUM 的 IAM 策略 \(p. 325\)](#)。

通过此选项，CloudWatch RUM 将创建以下资源：

- 新的 Amazon Cognito 身份池
- 未经身份验证的 Amazon Cognito 身份。这允许 RUM Web 客户端在不验证应用程序用户的情况下担任 IAM 角色。
- RUM Web 客户端将担任的 IAM 角色。附加到此角色的 IAM policy 允许其使用带有应用程序监控资源的 PutRumEvents API。换句话说，其允许 RUM Web 客户端向 RUM 发送数据。

RUM Web 客户端使用 Amazon Cognito 身份获取 Amazon 凭证。Amazon 凭证与 IAM 角色关联。IAM 角色有权通过 AppMonitor 资源使用 PutRumEvents。

Amazon Cognito 会发送必要的安全令牌，以确保应用程序能够向 CloudWatch RUM 发送数据。CloudWatch RUM 生成的 JavaScript 代码段包括以下代码行以启用身份验证。

```
{
  identityPoolId: [identity pool id], // e.g., 'us-west-2:EXAMPLE4a-66f6-4114-902a-EXAMPLEbad7'
  guestRoleArn: [iam role arn]        // e.g., 'arn:aws:iam::123456789012:role/Nexus-Monitor-us-east-1-123456789012_Unauth_5889316876161'
}
);
```

使用现有的 Amazon Cognito 身份池

如果您选择使用现有的 Amazon Cognito 身份池，则需要在将应用程序添加到 CloudWatch RUM 时指定身份池。身份池必须支持访问未经身份验证的身份。您还必须将以下权限添加到已附加到此身份池关联的 IAM 角色的 IAM policy。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "rum:PutRumEvents",
      "Resource": "arn:aws:rum:[region]:[accountid]:appmonitor/[app monitor name]"
    }
  ]
}
```

Amazon Cognito 会发送必要的安全令牌，以确保应用程序能够访问 CloudWatch RUM。

第三方提供商

如果您选择使用第三方提供商的私有身份验证，则必须从身份提供商处获取凭证并将其转发到 Amazon。执行此操作的最佳方法是使用安全令牌供应商。您可以使用任意安全令牌供应商，包括使用 Amazon Security Token Service 的 Amazon Cognito。有关 Amazon STS 的更多信息，请参阅[欢迎使用 Amazon Security Token Service API 参考](#)。

如果您想在这种情况下使用 Amazon Cognito 作为令牌供应商，则可以配置 Amazon Cognito，以与身份验证提供商结合使用。有关更多信息，请参阅[Amazon Cognito 身份池入门（联合身份）](#)。

配置 Amazon Cognito 以与身份提供商结合使用后，还需要执行以下操作：

- 创建具有以下权限的 IAM 角色。应用程序将使用该角色访问 Amazon。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "rum:PutRumEvents",
      "Resource": "arn:aws:rum:[region]:[accountID]:appmonitor/[app monitor name]"
    }
  ]
}
```

- 将以下内容添加到应用程序中，让应用程序将提供商的凭证传递给 CloudWatch RUM。插入该行，以便在用户登录到应用程序并且应用程序收到用于访问 Amazon 的凭证后运行。

```
cwr('setAwsCredentials', { /* Credentials or CredentialProvider */ });
```

有关 Amazon JavaScript SDK 凭证提供商的更多信息，请参阅 SDK for JavaScript v3 开发者指南中的[在 Web 浏览器中设置凭证](#)、SDK for JavaScript v2 开发人员指南中的[在 Web 浏览器中设置凭证](#)以及 [@aws-sdk/凭证提供商](#)。

您还可以使用适用于 CloudWatch RUM Web 客户端的软件开发工具包来配置 Web 客户端身份验证方法。有关 Web 客户端软件开发工具包的更多信息，请参阅 [CloudWatch RUM Web 客户端软件开发工具包](#)。

步骤 2：创建应用程序监控

要在应用程序中开始使用 CloudWatch RUM，您可以创建应用程序监控。创建应用程序监控后，RUM 会生成一个可粘贴到应用程序中的 JavaScript 代码段。该代码段拉入了 RUM Web 客户端代码。RUM Web 客户端从应用程序一定百分比的用户会话中捕获数据并将其发送到 RUM。

创建应用程序监控

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Application monitoring (应用程序监控)、RUM。
3. 选择 Add app monitor (添加应用程序监控)。
4. 输入应用程序的信息和设置：
 - 在 App monitor name (应用程序监控名称) 中，输入在 CloudWatch RUM 控制台中用于识别此应用程序监控的名称。
 - 在 Application domain (应用程序域) 中，输入应用程序具有管理权限的顶级域名。必须采用 URL 域格式。
选择 Include sub domain (包括子域)，让应用程序监控也能从顶级域名下的所有子域中收集数据。
5. 在 Configure RUM data collection (配置 RUM 数据收集) 中，指定是否希望应用程序监控收集以下内容：
 - Performance telemetry (性能遥测) – 收集有关页面加载和资源加载时间的信息
 - JavaScript errors (JavaScript 错误) – 收集应用程序引发的未处理 JavaScript 错误的信息
 - HTTP errors (HTTP 错误) – 收集应用程序引发的 HTTP 错误的信息

选择这些选项可以提供有关应用程序的更多信息，但也会生成更多 CloudWatch RUM 事件，从而产生更多的费用。

如果未选择以上任何选项，应用程序监控仍会收集会话开启事件和页面 ID，以便您可以查看有多少用户正在使用应用程序，包括按操作系统类型和版本、浏览器类型和版本、设备类型和位置的细分。

6. 如果您希望能够从采样的用户会话中收集用户 ID 和会话 ID，选择 Check this option to allow the CloudWatch RUM Web Client to set cookies (选中此选项以允许 CloudWatch RUM Web 客户端设置 Cookie)。用户 ID 由 RUM 随机生成。有关更多信息，请参阅[CloudWatch RUM Web 客户端 Cookie \(p. 335\)](#)。
7. 在 Session samples (会话样本) 中，输入将用于收集 RUM 数据的用户会话百分比。默认值为 100%。减少此数字将提供更少数据，但可以降低费用。有关 RUM 定价的更多信息，请参阅[RUM 定价 \(p. 324\)](#)。
8. 为 CloudWatch RUM 收集的终端用户数据将保留 30 天，然后删除。如果您想在 CloudWatch Logs 中保留 RUM 事件的副本并配置这些副本的保留时间，请选择 Data storage (数据存储) 下的 Check this option to store your application telemetry data in your CloudWatch Logs account (选中此选项以将应用程序遥测数据存储到 CloudWatch Logs 账户中)。预设情况下，CloudWatch Logs 日志组会将数据保留 30 天。您可以在 CloudWatch Logs 控制台中调整保留期。
9. 在 Authorization (授权) 中，指定是使用新的或现有的 Amazon Cognito 身份池，还是使用其他身份提供商。创建新的身份池是最简单的选择，无需其他设置步骤。有关更多信息，请参阅[步骤 1：授权应用程序将数据发送到 Amazon \(p. 325\)](#)。

创建新的 Amazon Cognito 身份池需要管理权限。有关更多信息，请参阅[使用 CloudWatch RUM 的 IAM 策略 \(p. 325\)](#)。

10. (可选) 预设情况下，您将 RUM 代码段添加到应用程序时，Web 客户端会将 JavaScript 标签注入到应用程序所有页面的 HTML 代码中以监控使用情况。要更改此选项，请选择 Configure pages (配置页面)，然后选择 Include only these pages (仅包含这些页面) 或 Exclude these pages (排除这些页面)。然后指定要包含或排除的页面。要指定包含或排除的页面，请输入其完整 URL。要指定其他页面，请选择 Add URL (添加 URL)。
11. 要启用 Amazon X-Ray 跟踪应用程序监控采样的用户会话，选择 Active tracing (活动跟踪)，然后选择 Trace my service with Amazon X-Ray (使用跟踪我的服务)。

如果选择此项，将会跟踪在应用程序监控采样用户会话期间发出的 XMLHttpRequest 和 fetch 请求。然后，您可在 RUM 控制面板中查看来自这些用户会话的跟踪和分段、CloudWatch ServiceLens 控制台和 X-Ray 控制台。

通过对 CloudWatch RUM Web 客户端进行其他配置更改，您可以向 HTTP 请求添加 X-Ray 跟踪标头，以启用下游 Amazon 托管式服务的用户会话端到端跟踪。有关更多信息，请参阅[启用 X-Ray 端到端跟踪 \(p. 329\)](#)。

12. (可选) 要向应用程序监控添加标签，请选择 Tags (标签)、Add new tag (添加新标签)。

然后，对于 Key (键)，输入标签的名称。您可以在 Value (值) 中添加可选的标签值。

要添加其他标签，请再次选择 Add new tag (添加新标签)。

有关更多信息，请参阅[标记 Amazon 资源](#)。

13. 选择 Add app monitor (添加应用程序监控)。
14. 在 Sample code (示例代码) 部分，您可以复制要使用的代码段以添加到您的应用程序当中。我们建议您选择 JavaScript 或 TypeScript，使用 NPM 安装 CloudWatch RUM Web 客户端，并将其作为 JavaScript 模块。

或者，您可以选择 HTML 来使用内容分发网络 (CDN)，以便安装 CloudWatch RUM Web 客户端。使用 CDN 的缺点在于，Web 客户端常被广告拦截器拦截。

15. 选择 Copy (复制) 或 Download (下载)，然后选择 Done (完成)。

(可选) 步骤 3：手动修改代码段以配置 CloudWatch RUM Web 客户端

您可以在代码段插入应用程序之前对其进行修改，以激活或停用多个选项。有关更多信息，请参阅[CloudWatch RUM Web 客户端文档](#)。

正如本章节中所述，您必须注意三个配置选项。

防止收集可能包含个人信息的资源 URL

预设情况下，CloudWatch RUM Web 客户端配置为记录应用程序下载的资源 URL。这些资源包括 HTML 文件、图像、CSS 文件、JavaScript 文件等。对于某些应用程序，URL 可能包含个人身份信息 (PII)。

如果应用程序属于这种情况，则强烈建议您通过在代码段配置中设置 `recordResourceUrl: false` 以禁用收集资源 URL，然后再将其插入到应用程序中。

手动记录页面浏览次数

默认情况下，Web 客户端会在页面首次加载以及调用浏览器的历史 API 时记录页面浏览次数。页面 ID 是 `window.location.pathname`。在某些情况下，Web 客户端工具不会记录所需的页面 ID。在这种情况下，必须使用 `disableAutoPageView` 配置禁用 Web 客户端的页面浏览自动化，并且必须对应用程序进行分析，以采用 `recordPageView` command 记录页面浏览次数。

启用 X-Ray 端到端跟踪

在创建应用程序监控时，选择 `Trace my service with Amazon X-Ray` (使用跟踪我的服务) 将启用跟踪在应用程序监控采样的用户会话期间发出的 `XMLHttpRequest` 和 `fetch` 请求。然后，您可在 CloudWatch RUM 控制面板、CloudWatch ServiceLens 控制台和 X-Ray 控制台中查看来自这些 HTTP 请求的跟踪。

默认情况下，这些客户端跟踪未连接到下游服务器端跟踪。要将客户端跟踪连接到服务器端跟踪并启用端到端跟踪，请在 Web 客户端中将 `addXRayTraceIdHeader` 选项设置为 `true`。这将导致 CloudWatch RUM Web 客户端向 HTTP 请求添加 X-Ray 跟踪标头。

下面的代码块展示了添加客户端跟踪的示例。为提高可读性，此示例中省略了一些配置选项。

```
<script>
  (function(n,i,v,r,s,c,u,x,z){...})(
    'cwr',
    '00000000-0000-0000-0000-000000000000',
    '1.0.0',
    'us-west-2',
    'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
    {
      enableXRay: true,
      telemetries: [
        'errors',
        'performance',
        [ 'http', { addXRayTraceIdHeader: true } ]
      ]
    }
  );
</script>
```

Warning

如果使用 SigV4 签署请求，则将 CloudWatch RUM Web 客户端配置为向 HTTP 请求添加 X-Ray 跟踪标头可能会导致跨源资源共享 (CORS) 失败或该请求的签名无效。有关更多信息，请参阅 [CloudWatch RUM Web 客户端文档](#)。我们强烈建议您在生产环境中添加客户端 X-Ray 跟踪标头之前测试应用程序。

有关更多信息，请参阅 [CloudWatch RUM Web 客户端文档](#)

步骤 4：将代码段插入应用程序

接下来，将上一部分中创建的代码段插入到应用程序中。

Warning

由代码段下载和配置的 Web 客户端使用 Cookie (或类似技术) 来帮助收集终端用户数据。在插入代码段之前, 请参阅 [CloudWatch RUM 的数据保护和数据隐私 \(p. 335\)](#)。

如果您没有以前生成的代码段, 则可以按照 [如何查找已生成的代码段? \(p. 350\)](#) 中的说明进行查找。

将 CloudWatch RUM 代码段插入应用程序

1. 将您在上一部分中复制或下载的代码段插入应用程序的 <head> 元素。在 <body> 元素或任何其他 <script> 标签之前插入。

以下是一个已生成的代码段示例：

```
<script>
(function (n, i, v, r, s, c, x, z) {
  x = window.AwsRumClient = {q: [], n: n, i: i, v: v, r: r, c: c};
  window[n] = function (c, p) {
    x.q.push({c: c, p: p});
  };
  z = document.createElement('script');
  z.async = true;
  z.src = s;
  document.head.insertBefore(z, document.getElementsByTagName('script')[0]);
})('cwr',
  '194a1c89-87d8-41a3-9d1b-5c5cd3dafbd0',
  '1.0.0',
  'us-east-2',
  'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
  {
    sessionSampleRate: 1,
    guestRoleArn: "arn:aws:iam::123456789012:role/RUM-Monitor-us-east-2-123456789012-5934510917361-Unauth",
    identityPoolId: "us-east-2:c90ef0ac-e3b8-4d1a-b313-7e73cfd21443",
    endpoint: "https://dataplane.rum.us-east-2.amazonaws.com",
    telemetries: ["performance", "errors", "http"],
    allowCookies: true,
    enableXRay: false
  });
</script>
```

2. 如果应用程序是多页 Web 应用程序, 则必须对要包含在数据集中的每个 HTML 页面重复步骤 1。

步骤 5：通过生成用户事件测试应用程序监控设置

插入代码段并且已更新的应用程序正在运行之后, 您可以通过手动生成用户事件对其进行测试。要进行测试, 建议您执行以下操作。此测试将产生标准 CloudWatch RUM 费用。

- 在 Web 应用程序中页面之间导航。
- 使用不同的浏览器和设备创建多个用户会话。
- 提出请求。
- 导致 JavaScript 错误。

生成一些事件后, 请在 CloudWatch RUM 控制面板中查看。有关更多信息, 请参阅[查看 CloudWatch RUM 控制面板 \(p. 332\)](#)。

来自用户会话的数据可能需要 15 分钟才能在控制面板中显示。

如果在应用程序中生成事件 15 分钟后没有看到数据, 请参阅 [CloudWatch RUM 故障排除 \(p. 351\)](#)。

配置 CloudWatch RUM Web 客户端

应用程序可以使用 CloudWatch RUM 生成的其中一个代码段来安装 CloudWatch RUM Web 客户端。生成的代码段支持两种安装方法：作为 JavaScript 模块并通过 NPM，或通过内容分发网络 (CDN)。为了获得最佳性能，我们建议使用 NPM 安装方法。有关使用此方法的更多信息，请参阅[作为 JavaScript 模块安装](#)。

如果您使用 CDN 安装选项，广告拦截器可能拦截由 CloudWatch RUM 提供的默认 CDN。若用户安装了广告拦截器，它将禁用应用程序监控。因此，我们建议您只在第一次启动 CloudWatch RUM 时使用默认的 CDN。有关此问题缓解方法的更多信息，请参阅[检测应用程序](#)。

代码段位于 HTML 文件的 <head> 标签，并且下载 Web 客户端来完成其安装，然后为其监控的应用程序配置 Web 客户端。该代码段是一种自动执行的函数，类似于以下内容。在此示例中，为提高可读性，已省略代码段函数的正文。

```
<script>
(function(n,i,v,r,s,c,u,x,z){...})({
  'cwr',
  '000000000-0000-0000-0000-000000000000',
  '1.0.0',
  'us-west-2',
  'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
  { /* Configuration Options Here */ }
});
</script>
```

Arguments

代码段接受六个参数：

- 用于在 Web 客户端上运行命令的命名空间，例如 'cwr'
- 应用程序监控的 ID，例如 '000000000-0000-0000-0000-000000000000'
- 应用程序版本，例如 '1.0.0'
- 应用程序监控的 Amazon 区域，例如 'us-west-2'
- Web 客户端的 URL，例如 'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js'
- 特定于应用程序的配置选项。有关更多信息，请参阅以下部分。

忽略错误

CloudWatch RUM Web 客户端会侦听应用程序中发生的所有类型的错误。如果您的应用程序发出了 JavaScript 错误，而您不想在 CloudWatch RUM 控制面板中查看这些错误，则可以配置 CloudWatch RUM Web 客户端以筛选出这些错误，以便在 CloudWatch RUM 控制面板上仅看到相关的错误事件。例如，您可能选择不在控制面板中查看某些 JavaScript 错误，因为您已经确定了这些错误的修复方法，而且这些错误的数量掩盖了其他错误。您可能还想忽略那些您无法修复的错误，因为它们归第三方所有的库所有。

有关如何分析 Web 客户端以筛选出特定 JavaScript 错误的更多信息，请参阅 Web 客户端 Github 文档中的[错误](#)中的示例。

配置选项

有关 CloudWatch RUM Web 客户端可用的配置选项信息，请参阅 [CloudWatch RUM Web 客户端文档](#)

使用页面组

使用页面组将应用程序中的不同页面相互关联，以便您可以查看页面组的汇总分析。例如，您可能希望查看所有登录页面的汇总页面加载时间。

您可以在 CloudWatch RUM Web 客户端中向页面查看事件添加一个或多个标签，从而将页面放入页面组中。以下示例将 /home 页面放入名为 en 的页面组和名为 landing 的页面组。

嵌入式脚本示例

```
cwr('recordPageView', { pageId: '/home', pageTags: ['en', 'landing']});
```

JavaScript 模块示例

```
awsRum.recordPageView({ pageId: '/home', pageTags: ['en', 'landing']})
```

查看 CloudWatch RUM 控制面板

CloudWatch RUM 可帮助您从用户会话中收集有关应用程序性能的数据，包括页面加载时间、Apdex 分数、使用的浏览器和设备、用户会话的地理位置以及存在错误的会话。所有这些信息都显示在控制面板中。

查看 RUM 控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Application monitoring (应用程序监控)、RUM。

Overview (概览) 选项卡显示您创建的其中一个应用程序监控收集的信息。

窗格的顶行显示此应用程序监控的以下信息：

- 页面加载数
- 页面平均加载速度
- Apdex 分数
- 与应用程序监控相关的任何告警的状态

应用程序性能索引 (Apdex) 分数表示终端用户的满意度。分数范围为 0 (最不满意) 到 1 (最满意)。分数仅基于应用程序性能。不要求用户对应用程序进行评级。有关 Apdex 分数的更多信息，请参阅 [CloudWatch RUM 如何设置 Apdex 分数 \(p. 333\)](#)。

其中一些窗格包含可用于进一步检查数据的链接。选择这些链接中的任何一个都将显示详细视图，视图顶部为 Performance (性能)、Errors & Sessions (错误和会话)、Browsers & Devices (浏览器和设备) 和 User Journey (用户历程) 选项卡。这些选项卡中的每个视图都包含筛选条件，可用于将显示结果缩小到特定浏览器、设备和国家/地区。

3. 要进一步聚焦，请选择 List view (列表视图) 选项卡，然后选择要聚焦的应用程序监控名称。这将显示所选应用程序监控的以下选项卡。

- Performance (性能) 选项卡显示页面性能信息，包括加载时间、会话信息、请求信息、Web 重要信息和一段时间内的页面加载次数。此视图包含在 Page loads (页面加载)、Requests (请求) 和 Location (位置) 焦点间切换视图的控制件。
- Errors & Sessions (错误和会话) 选项卡显示页面错误信息，包括用户最常看到的错误信息、错误最多的设备和浏览器，以及会话错误信息。此视图包括在 Errors (错误) 和 Sessions (会话) 焦点间切换视图的控制件。

错误列表视图位于 Errors (错误) 视图底部旁。要查看有关某个错误的更多详细信息，选择列表中该行左侧的按钮，然后选择 View details (查看详细信息)。

会话列表视图位于 Sessions (会话) 视图底部旁。要查看有关某个会话的更多详细信息，选择列表中该行左侧的按钮，然后选择 View details (查看详细信息)。

- Browsers & Device (浏览器和设备) 选项卡显示访问应用程序的不同浏览器和设备的性能和使用情况等信息。此视图包括在 Browsers (浏览器) 和 Devices (设备) 焦点间切换视图的控制件。

如果将范围缩小到单个浏览器，则会看到按浏览器版本细分的数据。

- User Journey (用户历程) 选项卡显示客户用于导航应用程序的路径。您可以看到客户进入应用程序的位置以及从应用程序退出的页面。您还可以看到其所用的路径以及遵循这些路径的客户百分比。您可以在节点上暂停以获取有关该页面的更多详细信息。您可以选择单个路径来突出显示连接以便更轻松查看。
4. (可选) 在前三个选项卡中的任何一个选项卡上，可以选择 Pages (页面) 按钮，并从列表选择一个页面或页面组。这会将显示的数据缩小到应用程序的单个页面或页面组。您还可以将列表中的页面和页面组标记为收藏。

CloudWatch RUM 如何设置 Apdex 分数

应用程序性能索引 (Apdex) 是一种开放标准，其定义了报告、基准和评估应用程序响应时间的方法。Apdex 分数可帮助您了解和识别随时间推移对应用程序性能的影响。

Apdex 分数表示终端用户的满意度分数，分数范围为 0 (最不满意) 到 1 (最满意度)。分数仅基于应用程序性能。不要求用户对应用程序进行评级。

每个 Apdex 分数均属于三个阈值之一。根据 Apdex 阈值和实际应用程序响应时间，有三种性能，如下所示：

- Satisfied (满意) – 实际应用程序响应时间小于等于 Apdex 阈值。对于 CloudWatch RUM，此阈值为 2000 毫秒或更低。
- Tolerable (尚可) – 实际应用程序响应时间大于 Apdex 阈值，但小于或等于 Apdex 阈值的四倍。对于 CloudWatch RUM，此阈值为 2000–8000 毫秒。
- Frustrating (不满) – 实际应用程序响应时间大于 Apdex 阈值的四倍。对于 CloudWatch RUM，此阈值为大于 8000 毫秒。

使用以下公式计算 0-1 Apdex 总分数：

$$(\text{positive scores} + \text{tolerable scores}/2)/\text{total scores} * 100$$

您可以使用 CloudWatch RUM 收集的 CloudWatch 指标

下表列出了可以使用 CloudWatch RUM 收集的指标。您可以在 CloudWatch 控制台中查看这些指标。有关更多信息，请参阅[查看可用指标 \(p. 100\)](#)。

这些指标在名为 AWS/RUM 的指标命名空间中发布。所有以下指标通过 application_name 维度发布。此维度的值为应用程序监控的名称。一些指标还会通过其他维度发布，如表中所列。

指标	Unit	描述
HttpStatusCodeCount	计数	应用程序中 HTTP 响应的计数，按响应状态代码显示。 其他维度： • event_details.response.status_code 是响应状态代码，例如 200、400、404 等。

指标	Unit	描述
		event_type 事件类型。目前，该维度的唯一可能值是 http。
JsErrorCount	计数	摄入的 JavaScript 错误事件的计数。
NavigationFrustratedCount	计数	比不满阈值 (8000ms) 高出 duration 的导航事件的计数。导航事件的持续时间在 PerformanceNavigationDuration 指标中跟踪。
NavigationSatisfiedCount	计数	duration 低于 Apdex 目标 (2000ms) 的导航事件的计数。导航事件的持续时间在 PerformanceNavigationDuration 指标中跟踪。
NavigationToleratedCount	计数	duration 在 2000ms 和 8000ms 之间的导航事件的计数。导航事件的持续时间在 PerformanceNavigationDuration 指标中跟踪。
PerformanceResourceDuration	毫秒	资源事件的 duration。 其他维度： - event_details.file.type 是资源事件的文件类型，例如样式表、文档、图像、脚本或字体。 - event_type 事件类型。目前，该维度的唯一可能值是 resource。
PerformanceNavigationDuration	毫秒	导航事件的 duration。
RumEventPayloadSize	字节	CloudWatch RUM 摄入的每个事件的规模。您还可以使用此指标的 SampleCount 统计数据来监控应用程序监控摄入的事件数量。
SessionCount	计数	应用程序监控摄入的会话启动事件的计数。即，已启动的新会话的数量。

指标	Unit	描述
WebVitalsCumulativeLayoutShift	None (无)	跟踪累计布局转移事件的值。
WebVitalsFirstInputDelay	毫秒	跟踪第一个输入延迟事件的值。
WebVitalsLargestContentfulPaint	毫秒	跟踪最大内容绘制事件的值。

CloudWatch RUM 的数据保护和数据隐私

Amazon [责任共担模式](#)适用于 Amazon CloudWatch RUM 中的数据保护和数据隐私。如该模式所述，Amazon 负责保护运行所有 Amazon 云的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。有关数据隐私的更多信息，请参阅[数据隐私常见问题](#)。有关欧洲数据保护的信息，请参阅 Amazon 安全博客上的 [Amazon 责任共担模式和 GDPR](#) 博客文章。有关遵守 GDPR 要求的更多资源，请参阅[一般数据保护条例 \(GDPR\)](#) 中心。

Amazon CloudWatch RUM 会根据对要收集的终端用户数据的输入生成一个代码段，可将其嵌入到网站或 Web 应用程序代码中。由代码段下载和配置的 Web 客户端使用 Cookie (或类似技术) 来帮助收集终端用户数据。Cookie (或类似技术) 的使用受某些司法管辖区的数据隐私法规约束。在使用 Amazon CloudWatch RUM 之前，强烈建议您评估适用法律规定的合规性义务，包括要求您针对 Cookie 的使用和终端用户数据的处理 (包括收集) 提供合法的隐私声明，并获得任何必要同意的任何适用法律要求。有关 Web 客户端如何使用 Cookie 以及 Web 客户端所收集的终端用户数据的更多信息，请参阅 [CloudWatch RUM Web 客户端收集的信息 \(p. 335\)](#) 和 [CloudWatch RUM Web 客户端 Cookie \(p. 335\)](#)。

强烈建议您不要在自由格式字段中输入敏感标识信息 (例如终端用户客户账户号码、邮箱地址或其他个人信息)。您输入到 Amazon CloudWatch RUM 或其他服务中的任何数据都可能包含在诊断日志中。

CloudWatch RUM Web 客户端 Cookie

默认情况下，CloudWatch RUM Web 客户端会收集有关用户会话的某些数据。您可以选择启用 Cookie，让 Web 客户端也收集每个会话的用户 ID 和会话 ID。用户 ID 由 RUM 随机生成。

如果启用了这些 Cookie，当您查看此应用程序监控的 RUM 控制面板时，RUM 可以显示以下类型的数据。

- 基于用户 ID 的汇总数据，例如唯一用户数量和出错的不同用户数量。
- 基于会话 ID 的汇总数据，例如会话数量和出错的会话数量。
- 用户历程，每个采样用户会话所包含的页面序列。

Important

如果不启用这些 Cookie，Web 客户端仍会记录有关终端用户会话的某些信息，例如浏览器类型/版本、操作系统类型/版本、设备类型等。收集这些信息是为了提供特定于页面的洞察，例如 Web 重要信息、页面浏览次数和出错页面。有关所记录数据的更多信息，请参阅 [CloudWatch RUM Web 客户端收集的信息 \(p. 335\)](#)。

CloudWatch RUM Web 客户端收集的信息

本节将记录 PuTrumEvent 架构，其定义了可以使用 CloudWatch RUM 从用户会话中收集的数据结构。

PuTrumEvent 请求将包含以下字段的数据结构发送到 CloudWatch RUM。

- 此批次 RUM 事件的 ID

- 应用程序监控详细信息，包括以下内容：
 - 应用程序监控 ID
 - 受监控的应用程序版本
- 用户详细信息，包括以下内容。只有应用程序监控启用 Cookie 时才会收集此信息。
 - Web 客户端生成的用户 ID
 - 会话 ID
- 此批次中的 [RUM 事件 \(p. 336\)](#) 数组。

RUM 活动架构

每个 RUM 事件的结构均包含以下字段。

- 事件 ID
- 时间戳
- 事件类型
- 用户代理
- [Metadata \(p. 336\)](#)
- [RUM 事件详细信息 \(p. 337\)](#)

RUM 事件元数据

元数据包括页面元数据、用户代理元数据、地理位置元数据和域元数据。

页面元数据

页面元数据包括以下内容：

- 页面 ID
- 页面标题
- 父页面 ID。只有应用程序监控启用 Cookie 时才会收集此信息。
- 交互深度只有在应用程序监控启用 Cookie 时才会收集此信息。

用户代理元数据

用户代理元数据包括以下内容：

- 浏览器语言
- 浏览器名称
- 浏览器版本
- 操作系统名称
- 操作系统版本
- 设备类型
- 平台类型

地理位置元数据

地理位置元数据包括以下内容：

- 国家/地区代码

- 细分代码

域元数据

域元数据包括 URL 域。

RUM 事件详细信息

根据事件类型，事件的详细信息遵循以下类型的架构之一。

会话开启事件

此事件不包含字段。只有应用程序监控启用 Cookie 时才会收集此信息。

页面视图架构

Page view (页面视图) 事件包含以下属性。您可以通过配置 Web 客户端来停用页面视图收集。有关更多信息，请参阅 [CloudWatch RUM Web 客户端文档](#)。

名称	类型	描述
页面 ID	字符串	在应用程序中唯一代表此页面的 ID。默认情况下，这是 URL 路径。
父页面 ID	字符串	用户导航到当前页面时所在页面的 ID。只有应用程序监控启用 Cookie 时才会收集此信息。
交互深度	字符串	只有应用程序监控启用 Cookie 时才会收集此信息。

JavaScript 错误架构

代理生成的 JavaScript 错误事件包含以下属性。仅当您选择收集错误遥测时，Web 客户端才会收集这些事件。

名称	类型	描述
错误类型	字符串	错误名称 (如果存在)。有关更多信息，请参阅 Error.prototype.name 。 某些浏览器可能不支持错误类型。
错误消息	字符串	错误的消息。有关更多信息，请参阅 Error.prototype.message 。如果错误字段不存在，则这是错误事件消息。有关更多信息，请参阅 ErrorEvent 。 不同浏览器之间的错误消息可能不一致。
堆栈跟踪	字符串	错误的堆栈跟踪 (如果存在) 将被截断为 150 个字符。有关更多信息，请参阅 Error.prototype.stack 。 某些浏览器可能不支持堆栈跟踪。

DOM 事件架构

代理生成的文档对象模型 (DOM) 事件包含以下属性。默认情况下，不会收集这些事件。只有在激活交互遥测时才会收集。有关更多信息，请参阅 [CloudWatch RUM Web 客户端文档](#)。

名称	类型	描述
Event (事件)	字符串	DOM 事件的类型，例如单击、滚动或悬停。有关更多信息，请参阅 事件参考 。
元素 ID	字符串	如果生成事件的元素有 ID，则此属性存储该 ID。有关更多信息，请参阅 Element.id 。 否则，此属性存储标签名称。有关更多信息，请参阅 Element.tagName 。

导航事件架构

只有在应用程序监控激活性能遥测时，才会收集导航事件。

导航事件使用 [Navigation timing Level 1](#) 和 [Navigation timing Level 2](#) API。并非所有浏览器都支持 Level 2 API，因此这些较新的字段是可选的。

Note

时间戳指标基于 [DOMHighResTimestamp](#)。使用 Level 2 API，默认情况下，所有时间都相对于 `startTime`。但是对于 Level 1，需从时间戳指标中减去 `navigationStart` 指标以获取相对值。所有时间戳值均以毫秒为单位。

导航事件包含以下属性。

名称	类型	描述	注意
initiatorType	字符串	表示启动性能事件的资源类型。	值：“navigation” Level 1：“navigation” Level 2：entryData.initiatorType
navigationType	字符串	表示导航类型。 此属性非必需。	值：该值必须是以下之一： • <code>navigate</code> 是通过选择链接、在浏览器的地址栏中输入 URL、表单提交或通过脚本操作的启动而开始的导航，而不是通过 <code>reload</code> 或 <code>back_forward</code> 开始的导航。 • <code>reload</code> 是通过浏览器重新加载操作或 <code>location.reload()</code> 的导航。

名称	类型	描述	注意
			• <code>back_forward</code> 是通过浏览器历史遍历操作的导航。 • <code>prerender</code> 是由预渲染提示启动的导航。有关更多信息，请参阅预渲染。
<code>startTime</code>	数字	指示触发事件的时间。	值：0 Level 1： <code>entryData.navigationStart - entryData.navigationStart</code> Level 2： <code>entryData.startTime</code>
<code>unloadEventStart</code>	数字	指示引发 <code>unload</code> 事件后，窗口中上一个文档开始卸载的时间。	值：如果不存在上一个文档，或者上一个文档或者其中一个所需的重新导向不是同一来源，则返回值为 0。 Level 1： <code>entryData.unloadEventStart > 0 ? entryData.unloadEventStart - entryData.navigationStart : 0</code> Level 2： <code>entryData.unloadEventStart</code>

名称	类型	描述	注意
promptForUnload	数字	卸载文档所用的时间。换言之，unloadEventStart 和 unloadEventEnd 之间的时间。UnloadEventEnd 表示卸载事件处理程序完成的时刻（以毫秒为单位）。	值：如果不存在上一个文档，或者上一个文档或者其中一个所需的重新导向不是同一来源，则返回值为 0。 Level 1：entryData.unloadEventEnd - entryData.unloadEventStart Level 2：entryData.unloadEventEnd - entryData.unloadEventStart
redirectCount	数字	表示在当前浏览的上下文中自上次非重新导向导航以来的重新导向次数。 此属性非必需。	值：如果没有重新导向，或者如果有与目标文档不同源的任何重新导向，则返回值为 0。 Level 1：不可用 Level 2：entryData.redirectCount
redirectStart	数字	第一次 HTTP 重新导向开始的时间。	值：如果没有重新导向，或者如果有与目标文档不同源的任何重新导向，则返回值为 0。 Level 1： entryData.redirectStart > 0 ? entryData.redirectStart - entryData.navigationStart : 0 Level 2：entryData.redirectStart

名称	类型	描述	注意
redirectTime	数字	HTTP 重新导向所用的时间。这是 redirectStart 与 redirectEnd 之间的差异。	Level 1 : : entryData.redirectEnd - entryData.redirectStart Level 2 : : entryData.redirectEnd - entryData.redirectStart
WorkerStart	数字	这是 PerformanceResourceTiming 接口的属性。这标志着工件线程操作开始。 此属性非必需。	值：如果 Service Worker 线程已运行，或紧靠启动 Service Worker 线程之前，则此属性将返回紧靠分派 FetchEvent 前的时间。如果 Service Worker 未拦截资源，则返回 0。 Level 1 : 不可用 Level 2 : entryData.workerStart
workerTime	数字	如果 Service Worker 未拦截资源，则返回工件线程操作所需的时间。 此属性非必需。	Level 1 : 不可用 Level 2 : $\text{entryData.workerStart} > 0 ? \text{entryData.fetchStart} - \text{entryData.workerStart} : 0$
fetchStart	数字	浏览器准备好使用 HTTP 请求获取文档的时间。这在检查任何应用程序缓存之前完成。	Level 1 : $: \text{entryData.fetchStart} > 0 ? \text{entryData.fetchStart} - \text{entryData.navigationStart} : 0$ Level 2 : entryData.fetchStart

名称	类型	描述	注意
domainLookupStart	数字	域查找开始的时间。	值：如果使用持久连接或信息存储在缓存或本地资源中，则该值将与 <code>fetchStart</code> 相同。 Level 1 : <pre>entryData.domainLookupStart > 0 ? entryData.domainLookupStart - entryData.navigationStart : 0</pre> Level 2 : <code>entryData.domainLookupStart</code>
dns	数字	域名查找所需的时间。	值：如果缓存了资源和 DNS 记录，则预期值为 0。 Level 1 : <code>entryData.domainLookupEnd</code> - <code>entryData.domainLookupStart</code> Level 2 : <code>entryData.domainLookupEnd</code> - <code>entryData.domainLookupStart</code>
nextHopProtocol	字符串	表示用于获取资源的网络协议的字符串。 此属性非必需。	Level 1 : 不可用 Level 2 : <code>entryData.nextHopProtocol</code>

名称	类型	描述	注意
connectStart	数字	紧靠用户代理开始建立与服务器的连接以检索文档之前的时间。	值：如果使用了 RFC2616 持久连接，或者如果从相关应用程序缓存或本地资源中检索当前文档，则此属性将返回 domainLookupEnd 值。 Level 1 : $\text{entryData.connectStart} > 0$ $?$ $\text{entryData.connectStart} - \text{entryData.navigationStart} : 0$ Level 2 : entryData.connectStart
connect	数字	评估建立传输连接或执行 SSL 身份验证所需的时间。其中还包括浏览器发出过多并发请求时所用的受阻时间。	Level 1 : entryData.connectEnd - entryData.connectStart Level 2 : entryData.connectEnd - entryData.connectStart
secureConnectionStart	数字	如果当前页面的 URL 模式为“https”，则此属性将返回紧靠用户代理启动握手过程以保护当前连接之前的时间。如果未使用 HTTPS，则返回 0。有关 URL 模式的更多信息，请参阅 URL 表示形式 。	公式 : entryData.secureConnect

名称	类型	描述	注意
tlsTime	数字	完成 SSL 握手所需的时间。	Level 1 : <pre>entryData.secureConnection > 0 ? entryData.connectEnd - entryData.secureConnection : 0</pre> Level 2 : <pre>entryData.secureConnection > 0 ? entryData.connectEnd - entryData.secureConnection : 0</pre>
requestStart	数字	紧靠用户代理开始从服务器、相关应用程序缓存或本地资源请求资源之前的时间。	Level 1 : <pre>: entryData.requestStart > 0 ? entryData.requestStart - entryData.navigationStart : 0</pre> Level 2 : entryData.requestStart
timeToFirstByte	数字	发出请求后接收信息第一个字节所需的时间。此时间相对于 startTime。	Level 1 : entryData.responseStart - entryData.requestStart Level 2 : entryData.responseStart - entryData.requestStart
responseStart	数字	紧靠用户代理的 HTTP 解析器从相关应用程序缓存、本地资源或服务器接收响应的第一个字节之后的时间。	Level 1 : <pre>entryData.responseStart > 0 ? entryData.responseStart - entryData.navigationStart : 0</pre> Level 2 : entryData.responseStart

名称	类型	描述	注意
responseTime	字符串	以字节形式从相关应用程序缓存、本地资源或服务器接收完整响应所用的时间。	Level 1 : $\text{entryData.responseStart} > 0$ $?$ $\text{entryData.responseEnd} - \text{entryData.responseStart} : 0$ Level 2 : $\text{entryData.responseStart} > 0$ $?$ $\text{entryData.responseEnd} - \text{entryData.responseStart} : 0$
domInteractive	数字	解析器在主文档上完成工作以及构建 HTML DOM 所用的时间。此时，其 Document.readyState 变为“交互式”，并引发相应的 readystatechange 事件。	Level 1 : $\text{entryData.domInteractive} > 0$ $?$ $\text{entryData.domInteractive} - \text{entryData.navigationStart} : 0$ Level 2 : entryData.domInteractive
domContentLoadedEventStart	数字	表示紧靠用户代理在当前文档中触发 DOMContentLoaded 事件之前的时间值。初始 HTML 文档完全加载和解析后，将触发 DOMContentLoaded 事件。此时，主 HTML 文档已完成解析，浏览器开始构建渲染树，并且还需要加载子资源。此步骤不会等待样式表、图像和子框架完成加载。	Level 1 : $\text{entryData.domContentLoaded} > 0$ $?$ $\text{entryData.domContentLoaded} - \text{entryData.navigationStart} : 0$ Level 2 : entryData.domContentLoaded

名称	类型	描述	注意
domContentLoaded	数字	渲染树构建的此开启时间和结束时间使用 domContentLoadedEventStart 和 domContentLoadedEventEnd 标记。其允许 CloudWatch RUM 跟踪执行情况。此属性是 domContentLoadedStart 与 domContentLoadedEnd 之间的差异。 在此期间，DOM 和 CSSOM 已准备就绪。此属性将等待脚本执行，但异步和动态创建的脚本除外。如果脚本依赖样式表，domContentLoaded 也将等待样式表。其不会等待图像。 Note 在 Google Chrome 的网络面板中，domContentLoadedStart 和 domContentLoadedEnd 的实际值与 domContentLoaded 近似。这表示从页面加载过程开始的 HTML DOM + CSSOM 渲染树构建时间。如果是导航指标，domContentLoaded 值表示开始和结束值之间的差异，这是仅下载子资源和渲染树构建所需的时间。	Level 2 : entryData.domContentLoaded - entryData.domContentLoaded Level 2 : entryData.domContentLoaded - entryData.domContentLoaded
domComplete	数字	紧靠浏览器将当前文档的当前文档准备就绪状态设置为完成之前的时间。此时，子资源（例如图像）的加载已经完成。这包括下载 CSS 和同步 JavaScript 等阻止内容所用的时间。在 Google Chrome 的网络面板中，此属性与 loadTime 近似。	Level 1 : <pre>entryData.domComplete > 0 ? entryData.domComplete - entryData.navigationStart : 0</pre> Level 2 : entryData.domComplete
domProcessingTime	数字	响应和加载事件开始之间的总时间。	Level 1 : entryData.loadEventStart - entryData.responseEnd Level 2 : entryData.loadEventStart - entryData.responseEnd

名称	类型	描述	注意
loadEventStart	数字	紧靠当前文档 load 事件触发之前的时间。	Level 1 : $\text{entryData.loadEventStart} > 0$? $\text{entryData.loadEventStart} - \text{entryData.navigationStart} : 0$ Level 2 : entryData.loadEventStart
loadEventTime	数字	loadEventStart 与 loadEventEnd 之间的差异。在此期间，将触发等待此加载事件的其他函数或逻辑。	Level 1 : entryData.loadEventEnd - entryData.loadEventStart Level 2 : entryData.loadEventEnd - entryData.loadEventStart
duration	字符串	持续时间为页面总加载时间。其记录下载主页面及其所有同步子资源的时间，以及渲染页面的时间。稍后还会继续下载脚本等异步资源。此属性是 loadEventEnd 与 startTime 属性之间的差异。	Level 1 : entryData.loadEventEnd - entryData.navigationStart Level 2 : entryData.duration
headerSize	数字	返回 transferSize 与 encodedBodySize 之间的差异。 此属性非必需。	Level 1 : 不可用 Level 2 : entryData.transferSize - entryData.encodedBodySize Level 2 : entryData.transferSize - entryData.encodedBodySize
compressionRatio	数字	encodedBodySize 和 decodedBodySize 的比率。encodedBodySize 的值是不包括 HTTP 标头的资源的压缩大小。decodedBodySize 的值是不包括 HTTP 标头的资源的解压大小。 此属性非必需。	Level 1 : 不可用。 Level 2 : $\text{entryData.encodedBodySize} > 0$? $\text{entryData.decodedBodySize} / \text{entryData.encodedBodySize} : 0$

名称	类型	描述	注意
navigationTimingLevel	数字	Navigation timing API 版本。	值：1 或 2

资源事件架构

只有在应用程序监控已激活性能遥测时，才会收集资源事件。

时间戳指标基于 [DOMHighResTimeStamp typedef](#)。对于 Level 2 API，默认情况下，所有计时都相对于 `startTime`。但是对于 Level 1 API，需从时间戳指标中减去 `navigationStart` 指标以获取相对值。所有时间戳值均以毫秒为单位。

代理生成的资源事件包含以下属性。

名称	类型	描述	注意
targetUrl	字符串	返回资源的 URL。	公式： entryData.name
initiatorType	字符串	表示启动性能资源事件的资源类型。	值："resource" 公式： entryData.initiatorType
duration	字符串	返回 <code>responseEnd</code> 与 <code>startTime</code> 属性之间的差异。 此属性非必需。	公式： entryData.duration
transferSize	数字	返回获取的资源大小（八位字节），包括响应标头字段和响应有效负载正文。 此属性非必需。	公式： entryData.transferSize
fileType	字符串	从目标 URL 模式派生的扩展。	

最大内容绘制事件架构

最大内容绘制事件包含以下属性。

只有在应用程序监控已激活性能遥测时，才会收集这些事件。

名称	描述		
值	有关更多信息，请参阅 Web 重要信息 。		

首次输入延迟事件

首次输入延迟事件包含以下属性。

只有在应用程序监控已激活性能遥测时，才会收集这些事件。

名称	描述		
值	有关更多信息，请参阅 Web 重要信息 。		

累计布局偏移事件

累计布局转移事件包含以下属性。

只有在应用程序监控已激活性能遥测时，才会收集这些事件。

名称	描述		
值	有关更多信息，请参阅 Web 重要信息 。		

HTTP 事件

HTTP 事件包含以下属性。其中将包含 `Response` 字段或 `Error` 字段，但无法二者皆包含。

只有在应用程序监控已激活 HTTP 性能遥测时，才会收集这些事件。

名称	描述
请求	请求字段包括 <code>Method</code> 字段，其中可能包含 GET、POST 等值。
响应	响应字段包含以下内容： • 状态，例如 2xx、4xx 或 5xx • 状态文本
Error	错误字段包含以下内容： • 类型 • 消息 • 文件名 • 行号 • 列号 • 堆栈跟踪

X-Ray 跟踪事件架构

只有在应用程序监控已激活 X-Ray 跟踪时，才会收集这些事件。

有关 X-Ray 跟踪事件架构的信息，请参阅 [Amazon X-Ray 分段文档](#)。

单页应用程序的路由更改计时

在传统的多页面应用程序中，当用户请求加载新内容时，用户实际上是在向服务器请求新的 HTML 页面。因此，CloudWatch RUM Web 客户端使用常规性能 API 指标来捕获加载时间。

但是，单页 Web 应用程序使用 JavaScript 和 Ajax 来更新界面，而无需从服务器加载新页面。浏览器计时 API 不会记录单页更新，而是使用路由更改计时。

CloudWatch RUM 支持监控来自服务器的整页加载和单页更新，但有以下区别：

- 对于路由更改计时，没有浏览器提供的指标，例如 `tlsTime`、`timeToFirstByte` 等等。
- 对于路由更改计时，`initiatorType` 字段将为 `route_change`。

CloudWatch RUM Web 客户端监听可能导致路由变更的用户交互，当记录此类用户交互时，Web 客户端会记录一个时间戳。如果满足以下两个条件，则将开始路由更改计时：

- 浏览器历史记录 API（浏览器前进和后退按钮除外）用于执行路由更改。
- 路由更改检测时间与最新用户交互时间戳之间的差异小于 1000 ms。这样可以避免数据偏斜。

然后，一旦路由更改计时开始，如果没有正在进行的 AJAX 请求和 DOM 更改，则该计时结束。然后，最新完成的活动的活动的时间戳将用作完成时间戳。

如果有持续的 AJAX 请求或 DOM 变更超过 10 秒（默认），则路由更改计时将超时。在这种情况下，CloudWatch RUM Web 客户端将不再记录此路由更改的计时。

因此，路由更改事件的持续时间按以下公式计算：

$$(\text{time of latest completed activity}) - (\text{latest user interaction timestamp})$$

管理使用 CloudWatch RUM 的应用程序

使用这些章节中的步骤管理应用程序对 CloudWatch RUM 的使用。

如何查找已生成的代码段？

要查找已为应用程序生成的 CloudWatch RUM 代码段，请按照以下步骤操作。

查找已生成的代码段

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Application monitoring（应用程序监控）、RUM。
3. 选择 List view（列表视图）。
4. 在应用程序监控的名称旁边，选择 View JavaScript（查看 JavaScript）。
5. 在 JavaScript Snippet（JavaScript 代码段）窗格中，选择 Copy to clipboard（复制到剪贴板）。

编辑应用程序

要更改应用程序监控的设置，请按以下步骤操作。您可以更改除应用程序监控名称之外的任何设置。

编辑应用程序对 CloudWatch RUM 的使用方式

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Application monitoring（应用程序监控）、RUM。
3. 选择 List view（列表视图）。
4. 选择应用程序名称旁边的按钮，然后选择 Actions（操作）、Edit（编辑）。
5. 更改除应用程序监控名称之外的任何设置。有关设置的更多信息，请参阅 [步骤 2：创建应用程序监控 \(p. 327\)](#)。

- 完成后，选择保存。

更改设置会更改代码段。现在，您必须将更新后的代码段粘贴到应用程序中。

- 在创建 JavaScript 代码段之后，选择 Copy to clipboard (复制到剪贴板) 或 Download (下载)，然后选择 Done (完成)。

要使用新设置开启监控，请将代码段插入应用程序中。在 `<body>` 元素或任何其他 `<script>` 标签之前，将代码段插入到应用程序的 `<head>` 元素中。

停止使用 CloudWatch RUM 或删除应用程序监控

要停止在应用程序中使用 CloudWatch RUM，请从应用程序代码中删除 RUM 生成的代码段。

要删除 RUM 应用程序监控，请按以下步骤操作。

删除应用程序监控

- 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
- 在导航窗格中，选择 Application monitoring (应用程序监控)、RUM。
- 选择 List view (列表视图)。
- 选择应用程序名称旁边的按钮，然后选择 Actions (操作)、Delete (删除)。
- 在确认框中，输入 **Delete**，然后选择 Delete (删除)。
- 如果您尚未执行此操作，请从应用程序代码中删除 CloudWatch RUM 代码段。

CloudWatch RUM 配额

CloudWatch RUM 具有以下配额。

资源	默认配额
应用程序监控	每个账户 20 您可以请求提高配额。
RUM 摄取率	每秒 50 个 PutRumEvents 请求 (TPS)。 您可以请求提高配额。

CloudWatch RUM 故障排除

本章节包含的提示可以帮助您对 CloudWatch RUM 进行故障排除。

应用程序没有数据

首先，请确保代码段已正确插入应用程序中。有关更多信息，请参阅[步骤 4：将代码段插入应用程序 \(p. 329\)](#)。

如果这不是问题所在，则可能是没有流量传入应用程序。通过与用户相同的方式访问应用程序来生成一些流量。

已停止记录应用程序的数据

应用程序可能经更新，现在不再包含 CloudWatch RUM 代码段。检查应用程序代码。

另一种可能性是，有人可能已经更新了代码段，但后来没有将更新的代码段插入应用程序中。按照 [如何查找已生成的代码段？](#) (p. 350) 中的说明查找当前正确的代码段，然后将其与粘贴到应用程序中的代码段进行比较。

洞察力和操作可视性

本部分中的主题介绍了 CloudWatch 的功能，这些功能可以帮助您获得对 Amazon 资源的操作了解。

主题

- [使用 Container Insights \(p. 353\)](#)
- [使用 Contributor Insights 分析高基数数据 \(p. 497\)](#)
- [Amazon CloudWatch Application Insights \(p. 508\)](#)

使用 Container Insights

使用 CloudWatch Container Insights 可以从容器化应用程序和微服务中收集、聚合和汇总指标与日志。Container Insights 可用于 Amazon EC2 上的 Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Kubernetes Service (Amazon EKS) 和 Kubernetes 平台。Container Insights 支持从部署在 Fargate 上的集群中收集针对 Amazon ECS 和 Amazon EKS 的指标。

CloudWatch 会自动收集许多资源（例如 CPU、内存、磁盘和网络）的指标。Container Insights 还提供诊断信息（如容器重新启动失败），以帮助您查明问题并快速解决问题。您还可以设置 Container Insights 收集的指标的 CloudWatch 告警。

Container Insights 使用[嵌入式指标格式 \(p. 754\)](#)将数据收集为性能日志事件。这些性能日志事件是使用结构化 JSON 架构的条目，该架构允许批量提取和存储高基数数据。从该数据中，CloudWatch 在集群、节点、pod、任务和服务级别创建聚合指标以作为 CloudWatch 指标。Container Insights 收集的指标在 CloudWatch 自动控制面板中可用，也可在 CloudWatch 控制台的 Metrics（指标）部分查看。

CloudWatch 不会根据日志数据自动创建所有可能的指标，以帮助您管理 Container Insights 成本。但是，您可以通过使用 CloudWatch Logs Insights 分析原始性能日志事件来查看其他指标和其他精细级别。

Container Insights 收集的指标按自定义指标收费。有关 CloudWatch 定价的更多信息，请参阅[Amazon CloudWatch 定价](#)。

在 Amazon EKS 和 Kubernetes 上，Container Insights 使用 CloudWatch 代理的容器化版本在集群中查找所有运行的容器。然后，它在每个性能堆栈层收集性能数据。

Container Insights 支持使用客户主密钥 (CMK) 对其收集的日志和指标进行加密。要启用此加密，必须手动为接收 Container Insights 数据的日志组启用 KMS 加密。这将导致 Container Insights 使用提供的 CMK 加密这些数据。仅支持对称 CMK。请勿使用非对称 CMK 加密日志组。

有关更多信息，请参阅[使用 Amazon KMS 加密 CloudWatch Logs 中的日志数据](#)。

支持的平台

Container Insights 可用于 Amazon EC2 实例上的 Amazon Elastic Container Service、Amazon Elastic Kubernetes Service 和 Kubernetes 平台。

- 对于 Amazon ECS，Container Insights 同时在 Linux 和 Windows Server 实例上的集群、任务和服务级别收集指标。它在 Linux 实例上只收集实例级别的指标。

对于 Amazon ECS，网络指标仅适用于 bridge 网络模式和 awsvpc 网络模式下的容器。它们不适用于 host 网络模式下的容器。

- 对于 Amazon Elastic Kubernetes Service 和 Amazon EC2 实例上的 Kubernetes 平台，仅在 Linux 实例上支持 Container Insights。

CloudWatch 代理容器镜像

亚马逊在 Amazon Elastic Container Registry 上提供 CloudWatch 代理容器镜像。有关更多信息，请参阅 Amazon ECR 上的 [cloudwatch-agent](#)。

支持的区域

以下区域支持适用于 Amazon ECS 的 Container Insights：

- 美国东部（弗吉尼亚州北部）
- 美国东部（俄亥俄州）
- 美国西部（加利福尼亚北部）
- 美国西部（俄勒冈）
- 非洲（开普敦）
- 亚太地区（孟买）
- 亚太地区（香港）
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- 亚太地区（新加坡）
- 亚太地区（东京）
- 亚太地区（雅加达）
- 亚太地区（悉尼）
- 加拿大（中部）
- 欧洲（法兰克福）
- 欧洲（爱尔兰）
- 欧洲（伦敦）
- 欧洲（巴黎）
- Europe (Stockholm)
- Middle East (Bahrain)
- 南美洲（圣保罗）
- Amazon GovCloud（美国东部）
- Amazon GovCloud（美国西部）
- China (Beijing)
- China (Ningxia)

Amazon EKS 和 Kubernetes 的支持区域

以下区域支持适用于 Amazon EKS 和 Kubernetes 的 Container Insights：

- 美国东部（弗吉尼亚州北部）

- 美国东部 (俄亥俄州)
- 美国西部 (加利福尼亚北部)
- 美国西部 (俄勒冈州)
- 亚太地区 (香港)
- 亚太地区 (孟买)
- 亚太地区 (首尔)
- 亚太地区 (新加坡)
- 亚太地区 (悉尼)
- 亚太区域 (东京)
- 加拿大 (中部)
- China (Beijing)
- China (Ningxia)
- 欧洲 (法兰克福)
- 欧洲 (爱尔兰)
- 欧洲 (伦敦)
- 欧洲 (巴黎)
- Europe (Stockholm)
- Middle East (Bahrain)
- 南美洲 (圣保罗)
- Amazon GovCloud (美国东部)
- Amazon GovCloud (美国西部)

设置 Container Insights

对于 Amazon ECS、Amazon EKS 和 Kubernetes , Container Insights 设置过程不同。

主题

- [在 Amazon ECS 上设置 Container Insights \(p. 355\)](#)
- [在 Amazon EKS 和 Kubernetes 上设置 Container Insights \(p. 367\)](#)

在 Amazon ECS 上设置 Container Insights

您可以使用以下一个或两个选项在 Amazon ECS 集群上启用 Container Insights :

- 使用 Amazon Web Services Management Console或 Amazon CLI 开始收集集群级别、任务级别和服务级别指标。
- 将 CloudWatch 代理部署为 DaemonSet 来开始收集 Amazon EC2 实例上托管的集群的实例级别指标。

主题

- [在 Amazon ECS 上针对集群级别和服务级别指标设置 Container Insights \(p. 356\)](#)
- [使用 Amazon Distro for OpenTelemetry 在 Amazon ECS 上设置 Container Insights \(p. 357\)](#)
- [部署 CloudWatch 代理以收集 Amazon ECS 上的 EC2 实例级别指标 \(p. 359\)](#)
- [部署 Amazon Distro for OpenTelemetry 以收集 Amazon ECS 集群上的 EC2 实例级指标 \(p. 365\)](#)

- [设置 Firelens 以向 CloudWatch Logs 发送日志 \(p. 366\)](#)

在 Amazon ECS 上针对集群级别和服务级别指标设置 Container Insights

可以在新的和现有的 Amazon ECS 集群上启用 Container Insights。Container Insights 在集群、任务和服务级别收集指标。对于现有集群，使用 Amazon CLI。对于新集群，使用 Amazon ECS 控制台或 Amazon CLI。

如果您在 Amazon EC2 实例上使用 Amazon ECS，并且您要收集 Container Insights 中的网络 and 存储指标，则必须使用包含 Amazon ECS 代理版本 1.29 的 AMI 启动该实例。有关更新代理版本的信息，请参阅[更新 Amazon ECS 容器代理](#)

您可以使用 Amazon CLI 设置账户级别权限，以便为您账户中创建的任何新 Amazon ECS 集群启用 Container Insights。为此，请输入以下命令。

```
aws ecs put-account-setting --name "containerInsights" --value "enabled"
```

在现有 Amazon ECS 集群上设置 Container Insights

要在现有 Amazon ECS 集群上启用 Container Insights，请输入以下命令。您必须运行版本 1.16.200 或更高版本的 Amazon CLI，才能正常运行以下命令。

```
aws ecs update-cluster-settings --cluster myECScluster --settings  
name=containerInsights,value=enabled
```

在新的 Amazon ECS 集群上设置 Container Insights

可通过两种方式在新的 Amazon ECS 集群上启用 Container Insights。您可以配置 Amazon ECS，以便默认情况下为所有新集群启用 Container Insights。否则，您可以在创建新集群时启用它。

使用 Amazon Web Services Management Console

默认情况下，您可以在所有新集群上启用 Container Insights，也可以在创建单个集群时对其启用 Container Insights。

默认情况下在所有新集群上启用 Container Insights

1. 在 <https://console.aws.amazon.com/ecs/> 处打开 Amazon ECS 控制台。
2. 在导航窗格中，选择 Account Settings (账户设置)。
3. 选中页面底部的复选框以默认启用 Container Insights。

如果您在默认情况下未使用上述过程在所有新集群上启用 Container Insights，则使用以下步骤创建一个启用了 Container Insights 的集群。

创建启用了 Container Insights 的集群

1. 在 <https://console.aws.amazon.com/ecs/> 处打开 Amazon ECS 控制台。
2. 在导航窗格中，选择 Clusters。
3. 选择创建集群。
4. 在下一页上，执行以下操作：
 - a. 为您的集群命名。
 - b. 如果您没有 VPC，请选中复选框以创建一个。您可以使用 VPC 的默认值。

- c. 填写所有其他所需信息，包括实例类型。
- d. 选择 Enabled Container Insights (已启用 Container Insights)。
- e. 选择 Create (创建)。

现在，您可以在集群中创建任务定义、运行任务和启动服务。有关更多信息，请参阅以下内容：

- [创建任务定义](#)
- [正在运行的任务](#)
- [创建服务](#)

使用 Amazon CLI 在新的 Amazon ECS 集群上设置 Container Insights

要在默认情况下在所有新集群上启用 Container Insights，请输入以下命令。

```
aws ecs put-account-setting --name "containerInsights" --value "enabled"
```

如果您在默认情况下未使用上述命令在所有新集群上启用 Container Insights，请输入以下命令来创建一个启用了 Container Insights 的新集群。您必须运行版本 1.16.200 或更高版本的 Amazon CLI，才能正常运行以下命令。

```
aws ecs create-cluster --cluster-name myECScluster --settings  
"name=containerInsights,value=enabled"
```

在 Amazon ECS 集群上禁用 Container Insights

要在现有 Amazon ECS 集群上禁用 Container Insights，请输入以下命令。

```
aws ecs update-cluster-settings --cluster myECScluster --settings  
name=containerInsights,value=disabled
```

使用 Amazon Distro for OpenTelemetry 在 Amazon ECS 上设置 Container Insights

如果需要使用 Amazon Distro for OpenTelemetry 在 Amazon ECS 集群上设置 CloudWatch Container Insights，请使用此部分。有关 Amazon Distro for Open Telemetry 的更多信息，请参阅 [Amazon Distro for OpenTelemetry](#)。

这些步骤假设您已具有正在运行 Amazon ECS 的集群。有关将 Amazon Distro for Open Telemetry 与 Amazon ECS 结合使用以及为此设置 Amazon ECS 集群的更多信息，请参阅在 [Amazon Elastic Container Service](#) 中设置 [Amazon Distro for OpenTelemetry Collector](#)。

步骤 1：创建任务角色

第一步是在集群中创建 Amazon OpenTelemetry Collector 将使用的任务角色。

为 Amazon Distro for OpenTelemetry 创建任务角色

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中选择策略，然后选择创建策略。
3. 选择 JSON 选项卡，然后复制以下策略：

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents",
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:DescribeLogStreams",
      "logs:DescribeLogGroups",
      "ssm:GetParameters"
    ],
    "Resource": "*"
  }
]
```

4. 选择 Review policy (查看策略)。
5. 对于名称, 输入 **AWSDistroOpenTelemetryPolicy**, 然后选择 Create policy (创建策略)。
6. 在左侧的导航窗格中, 选择 Roles (角色), 然后选择 Create role (创建角色)。
7. 在服务列表中, 选择 Elastic Container Service (Elastic Container 服务)。
8. 在页面下方, 选择 Elastic Container Service Task (Elastic Container 服务任务), 然后选择 Next: Permissions (下一步: 权限)。
9. 在策略列表中, 搜索 AWSDistroOpenTelemetryPolicy。
10. 选中 AWSDistroOpenTelemetryPolicy 旁的复选框。
11. 选择 Next: Tags (下一步: 标签), 然后选择 Next: Review (下一步: 审核)。
12. 对于 Role name (角色名称), 输入 **AWSOpenTelemetryTaskRole**, 然后选择 Create role (创建角色)。

步骤 2: 创建任务执行角色

下一步是为 Amazon OpenTelemetry Collector 创建任务执行角色。

为 Amazon Distro for OpenTelemetry 创建任务执行角色

1. 通过以下网址打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
2. 在左侧的导航窗格中, 选择 Roles (角色), 然后选择 Create role (创建角色)。
3. 在服务列表中, 选择 Elastic Container Service (Elastic Container 服务)。
4. 在页面下方, 选择 Elastic Container Service Task (Elastic Container 服务任务), 然后选择 Next: Permissions (下一步: 权限)。
5. 在策略列表中, 搜索 AmazonECSTaskExecutionRolePolicy, 然后选中 AmazonECSTaskExecutionRolePolicy 旁的复选框。
6. 在策略列表中, 搜索 CloudWatchLogsFullAccess, 然后选中 CloudWatchLogsFullAccess 旁的复选框。
7. 在策略列表中, 搜索 AmazonSSMReadOnlyAccess, 然后选中 AmazonSSMReadOnlyAccess 旁的复选框。
8. 依次选择 Next: Tags (下一步: 标签) 和 Next: Review (下一步: 查看)。
9. 对于 Role name (角色名称), 输入 **AWSOpenTelemetryTaskExecutionRole**, 然后选择 Create role (创建角色)。

步骤 3: 创建任务定义

下一步是创建任务定义。

为 Amazon Distro for OpenTelemetry 创建任务定义

1. 在 <https://console.aws.amazon.com/ecs/> 处打开 Amazon ECS 控制台。
2. 在左侧的导航窗格中，依次选择 Task Definitions (任务定义) 和 Create new Task Definition (创建新任务定义)。
3. 选择 FARGATE 或 EC2，然后选择 Next step (下一步)。
4. 输入任务定义名称，如 **aws-otel**。
5. 对于 Task Role (任务角色)，选择您之前创建的 AWSOpenTelemetryTaskRole。
6. 对于 Task execution role (任务执行角色)，选择您之前创建的 AWSOpenTelemetryTaskExecutionRole。
7. 填写 Task memory (任务内存) 和 Task CPU (任务 CPU)。
8. 在 Container Definitions (容器定义) 下，选择 Add container (添加容器)。
9. 对于 Container name (容器名称)，输入 **aws-otel-collector**。对于 Image (镜像)，输入 **public.ecr.aws/aws-observability/aws-otel-collector**。
10. 在 ENVIRONMENT (环境) 下，对于 Command (命令) 输入 **--config=/etc/ecs/container-insights/otel-task-metrics-config.yaml**

此 YAML 文件包含在 Docker 镜像中，并包含使用容器指标的配置。

11. 如果使用的是 EC2 启动类型，请为 TCP 输入 55680 的端口映射。
12. 完成添加容器的步骤。

有关将 Amazon OpenTelemetry 收集器与 Amazon ECS 结合使用的更多信息，请参阅在 [Amazon Elastic Container Service](#) 中设置 [Amazon Distro for OpenTelemetry Collector](#)。

步骤 4：运行任务

最后一步是运行您创建的任务。

运行 Amazon Distro for OpenTelemetry 的任务

1. 在 <https://console.aws.amazon.com/ecs/> 处打开 Amazon ECS 控制台。
 2. 在左侧导航窗格中，选择 Task Definitions (任务定义)，然后选择您刚刚创建的任务。
 3. 选择 Actions，Run Task。
- 接下来，您可以在 CloudWatch 控制台中检查新指标。
4. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
 5. 在左侧导航窗格中，选择 Metrics。

您会看到 ECS/ContainerInsights 命名空间。选择该命名空间，然后您会看到八个指标。

部署 CloudWatch 代理以收集 Amazon ECS 上的 EC2 实例级别指标

要部署 CloudWatch 代理以从 EC2 实例上托管的 Amazon ECS 集群中收集实例级别指标，请使用具有默认配置的快速启动设置，或手动安装该代理以便对其进行自定义。

这两种方法都要求您至少部署了一个具有 EC2 启动类型的 Amazon ECS 集群。这些方法还假设您已安装 Amazon CLI。此外，要在以下过程中运行这些命令，您必须登录到具有 IAMFullAccess 和 AmazonECS_FullAccess 策略的账户或角色。

主题

- [使用 Amazon CloudFormation 进行快速设置 \(p. 360\)](#)

- [手动和自定义设置 \(p. 361\)](#)

使用 Amazon CloudFormation 进行快速设置

要使用快速设置，请输入以下命令来使用 Amazon CloudFormation 安装代理。将 `cluster-name` 和 `cluster-region` 分别替换为您的 Amazon ECS 集群的名称和区域。

此命令将创建 IAM 角色，即 CWAgentECSTaskRole 和 CWAgentECSExecutionRole。如果您的账户中已有这些角色，请在输入命令时使用 `ParameterKey=CreateIAMRoles,ParameterValue=False` 而非 `ParameterKey=CreateIAMRoles,ParameterValue=True`。否则，此命令将失败。

Note

默认情况下，以下设置步骤以匿名用户身份从 Docker Hub 拉取容器镜像。这种拉取可能会受到速率限制。有关更多信息，请参阅[CloudWatch 代理容器镜像 \(p. 354\)](#)。

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation create-stack --stack-name CWAgentECS-${ClusterName}-${Region} \
  --template-body https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
  container-insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/
  cwagent-ecs-instance-metric/cloudformation-quickstart/cwagent-ecs-instance-metric-cfn.json
  \
  --parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
    ParameterKey=CreateIAMRoles,ParameterValue=True \
  --capabilities CAPABILITY_NAMED_IAM \
  --region ${Region}
```

(备选方案) 使用您自己的 IAM 角色

如果您要使用自己的自定义 ECS 任务角色和 ECS 任务执行角色，而不是使用 CWAgentECSTaskRole 和 CWAgentECSExecutionRole 角色，请先确保要用作 ECS 任务角色的角色已附加 CloudWatchAgentServerPolicy。此外，请确保要用作 ECS 任务执行角色的角色同时附加了 CloudWatchAgentServerPolicy 和 AmazonECSTaskExecutionRolePolicy 策略。然后，输入以下命令。在该命令中，将 `task-role-arn` 替换为您的自定义 ECS 任务角色的 ARN，并将 `execution-role-arn` 替换为您的自定义 ECS 任务执行角色的 ARN。

```
ClusterName=cluster-name
Region=cluster-region
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
aws cloudformation create-stack --stack-name CWAgentECS-${ClusterName}-${Region} \
  --template-body https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
  container-insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/
  cwagent-ecs-instance-metric/cloudformation-quickstart/cwagent-ecs-instance-metric-cfn.json
  \
  --parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
    ParameterKey=TaskRoleArn,ParameterValue=${TaskRoleArn} \
    ParameterKey=ExecutionRoleArn,ParameterValue=${ExecutionRoleArn} \
  --capabilities CAPABILITY_NAMED_IAM \
  --region ${Region}
```

对快速设置进行故障排除

要查看 Amazon CloudFormation 堆栈的状态，请输入以下命令。

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stacks --stack-name CWAgentECS-${ClusterName}-${Region} --region
  $Region
```

如果您看到的 `StackStatus` 不是 `CREATE_COMPLETE` 或 `CREATE_IN_PROGRESS`，请查看堆栈事件以查找错误。输入以下命令。

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack-events --stack-name CWAgentECS-$ClusterName-$Region --region $Region
```

要查看 `cwagent` 守护进程服务的状态，请输入以下命令。在输出中，您应看到 `deployment` 部分中的 `runningCount` 等于 `desiredCount`。如果二者不相等，请查看输出中的 `failures` 部分。

```
ClusterName=cluster-name
Region=cluster-region
aws ecs describe-services --services cwagent-daemon-service --cluster $ClusterName --region $Region
```

您还可以使用 CloudWatch Logs 控制台来查看代理日志。查找 `/ecs/ecs-cwagent-daemon-service` 日志组。

删除 CloudWatch 代理的 Amazon CloudFormation 堆栈

如果您需要删除 Amazon CloudFormation 堆栈，请输入以下命令。

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation delete-stack --stack-name CWAgentECS-${ClusterName}-${Region} --region ${Region}
```

手动和自定义设置

执行此部分中的步骤，手动部署 CloudWatch 代理以从 EC2 实例上托管的 Amazon ECS 集群收集实例级别指标。

必要的 IAM 角色和策略

需要两个 IAM 角色。如果这两个角色不存在，则必须创建它们。有关这些角色的更多信息，请参阅[任务的 IAM 角色](#)和[Amazon ECS 任务执行角色](#)。

- 一个 ECS 任务角色，可供 CloudWatch 代理用来发布指标。如果此角色已存在，则必须确保它附加了 `CloudWatchAgentServerPolicy` 策略。
- 一个 ECS 任务执行角色，可供 Amazon ECS 代理用来启动 CloudWatch 代理。如果此角色已存在，则必须确保它附加了 `AmazonECSTaskExecutionRolePolicy` 和 `CloudWatchAgentServerPolicy` 策略。

如果您还没有这些角色，则可使用以下命令来创建它们并附加必要的策略。第一条命令将创建 ECS 任务角色。

```
aws iam create-role --role-name CWAgentECSTaskRole \
  --assume-role-policy-document '{"Version": "\2012-10-17\","Statement": [{"Sid": "\","Effect": "\Allow\","Principal": {"Service": "\ecs-tasks.amazonaws.com\"}, {"Action": "\sts:AssumeRole\"}]}'
```

在输入上一条命令后，将命令输出中的 `Arn` 的值记为“`TaskRoleArn`”。稍后，您在创建任务定义时将需要使用它。然后，输入以下命令来附加必要的策略。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
```

```
--role-name CWAgentECSTaskRole
```

下一条命令将创建 ECS 任务执行角色。

```
aws iam create-role --role-name CWAgentECSExecutionRole \
  --assume-role-policy-document '{"Version": "2012-10-17", "Statement": [{ "Sid":
  "\", "Effect": "Allow", "Principal": { "Service": "ecs-tasks.amazonaws.com"},
  "Action": "sts:AssumeRole"}]}'
```

在输入上一条命令后，将命令输出中的 Arn 的值记为“ExecutionRoleArn”。稍后，您在创建任务定义时将需要使用它。然后，输入以下命令来附加必要的策略。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
  --role-name CWAgentECSExecutionRole

aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy \
  --role-name CWAgentECSExecutionRole
```

创建任务定义并启动守护进程服务

创建一个任务定义并使用它将 CloudWatch 代理作为守护进程服务启动。要创建任务定义，请输入以下命令。在前面的行中，将占位符替换为部署的实际值。**logs-region** 为 CloudWatch Logs 所在的区域，**cluster-region** 为集群所在的区域。**task-role-arn** 为您使用的 ECS 任务角色的 Arn，**execution-role-arn** 为 ECS 任务执行角色的 Arn。

```
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
AWSLogsRegion=logs-region
Region=cluster-region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/
latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-ecs-instance-
metric/cwagent-ecs-instance-metric.json \
  | sed "s|{{task-role-arn}}|${TaskRoleArn}|;s|{{execution-role-arn}}|
${ExecutionRoleArn}|;s|{{awslogs-region}}|${AWSLogsRegion}|" \
  | xargs -0 aws ecs register-task-definition --region ${Region} --cli-input-json
```

然后，运行以下命令来启动守护程序服务。将 **cluster-name** 和 **cluster-region** 分别替换为您的 Amazon ECS 集群的名称和区域。

```
ClusterName=cluster-name
Region=cluster-region
aws ecs create-service \
  --cluster ${ClusterName} \
  --service-name cwagent-daemon-service \
  --task-definition ecs-cwagent-daemon-service \
  --scheduling-strategy DAEMON \
  --region ${Region}
```

如果您看到的错误消息为 An error occurred (InvalidParameterException) when calling the CreateService operation: Creation of service was not idempotent，则您已创建名为 cwagent-daemon-service 的守护程序服务。您必须先删除该服务，并使用以下命令作为示例。

```
ClusterName=cluster-name
Region=cluster-region
aws ecs delete-service \
  --cluster ${ClusterName} \
  --service cwagent-daemon-service \
```

```
--region ${Region} \
--force
```

(可选) 高级配置

(可选) 您可以使用 SSM 来指定 EC2 实例上托管的 Amazon ECS 集群中 CloudWatch 代理的其他配置选项。这些选项如下所示：

- `metrics_collection_interval` – CloudWatch 代理收集指标的频率 (以秒为单位)。默认值为 60。范围为 1–172000。
- `endpoint_override` – (可选) 指定要将日志发送到的其他端点。如果您从 VPC 的集群中发布并希望将日志数据传输到 VPC 终端节点，则可能需要执行该操作。

`endpoint_override` 的值必须是表示 URL 的字符串。

- `force_flush_interval` – 以秒为单位指定日志在发送到服务器之前保留在内存缓冲区中的最大时间量。无论此字段的设置如何，如果缓冲区中的日志大小达到 1 MB，日志会立即发送到服务器。默认值为 5 秒。
- `region` – 默认情况下，代理将指标发布到 Amazon ECS 容器实例所在的同一区域。要覆盖此设置，您可以在此处指定其他区域。例如，`"region" : "us-east-1"`

以下是自定义配置的示例：

```
{
  "agent": {
    "region": "us-east-1"
  },
  "logs": {
    "metrics_collected": {
      "ecs": {
        "metrics_collection_interval": 30
      }
    },
    "force_flush_interval": 5
  }
}
```

在 Amazon ECS 容器中自定义 CloudWatch 代理配置

1. 确保将 `AmazonSSMReadOnlyAccess` 策略附加到 Amazon ECS 任务执行角色。可输入以下命令来执行此操作。此示例假定您的 Amazon ECS 任务执行角色是 `CWAgentECSExecutionRole`。如果您使用其他角色，请在以下命令中替换该角色名称。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonSSMReadOnlyAccess \
    --role-name CWAgentECSExecutionRole
```

2. 创建与上一示例类似的自定义配置文件。将此文件命名为 `/tmp/ecs-cwagent-daemon-config.json`。
3. 运行以下命令以将此配置放入 Parameter Store 中。将 `cluster-region` 替换为 Amazon ECS 集群的区域。要运行此命令，您必须登录到具有 `AmazonSSMFullAccess` 策略的用户或角色。

```
Region=cluster-region
aws ssm put-parameter \
  --name "ecs-cwagent-daemon-service" \
  --type "String" \
  --value "`cat /tmp/ecs-cwagent-daemon-config.json`" \
  --region $Region
```

4. 将任务定义文件下载到本地文件，例如 `/tmp/cwagent-ecs-instance-metric.json`

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-ecs-instance-metric/cwagent-ecs-instance-metric.json -o /tmp/cwagent-ecs-instance-metric.json
```

5. 修改任务定义文件。删除以下部分：

```
"environment": [
    {
        "name": "USE_DEFAULT_CONFIG",
        "value": "True"
    }
],
```

将此部分替换为以下内容：

```
"secrets": [
    {
        "name": "CW_CONFIG_CONTENT",
        "valueFrom": "ecs-cwagent-daemon-service"
    }
],
```

6. 通过执行以下步骤，将代理作为守护程序服务重新启动：

- a. 运行以下命令。

```
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
AWSLogsRegion=logs-region
Region=cluster-region
cat /tmp/cwagent-ecs-instance-metric.json \
| sed "s|{{task-role-arn}}|${TaskRoleArn}|;s|{{execution-role-arn}}|${ExecutionRoleArn}|;s|{{awslogs-region}}|${AWSLogsRegion}|" \
| xargs -0 aws ecs register-task-definition --region ${Region} --cli-input-json
```

- b. 运行以下命令来启动守护程序服务。将 *cluster-name* 和 *cluster-region* 分别替换为您的 Amazon ECS 集群的名称和区域。

```
ClusterName=cluster-name
Region=cluster-region
aws ecs create-service \
    --cluster ${ClusterName} \
    --service-name cwagent-daemon-service \
    --task-definition ecs-cwagent-daemon-service \
    --scheduling-strategy DAEMON \
    --region ${Region}
```

如果您看到的错误消息为 An error occurred (InvalidParameterException) when calling the CreateService operation: Creation of service was not idempotent, 则您已创建名为 `cwagent-daemon-service` 的守护程序服务。您必须先删除该服务，并使用以下命令作为示例。

```
ClusterName=cluster-name
Region=Region
aws ecs delete-service \
    --cluster ${ClusterName} \
    --service cwagent-daemon-service \
```

```
--region ${Region} \  
--force
```

部署 Amazon Distro for OpenTelemetry 以收集 Amazon ECS 集群上的 EC2 实例级指标

使用此部分中的步骤使用 Amazon Distro for OpenTelemetry 来收集 Amazon ECS 集群上的 EC2 实例级指标。有关 Amazon Distro for OpenTelemetry 的更多信息，请参阅 [Amazon Distro for OpenTelemetry](#)。

这些步骤假设您已具有正在运行 Amazon ECS 的集群。此集群必须使用 EC2 启动类型进行部署。有关将 Amazon Distro for Open Telemetry 与 Amazon ECS 结合使用以及为此设置 Amazon ECS 集群的更多信息，请参阅 [设置 Amazon Elastic Container Service 中的 Amazon Distro for OpenTelemetry Collector](#) 以了解 ECS EC2 实例级指标。

主题

- [使用 Amazon CloudFormation 进行快速设置 \(p. 365\)](#)
- [手动和自定义设置 \(p. 366\)](#)

使用 Amazon CloudFormation 进行快速设置

下载 Amazon CloudFormation 模板文件，用于在 EC2 上为 Amazon ECS 安装 Amazon Distro for OpenTelemetry 收集器。运行以下 curl 命令。

```
curl -O https://raw.githubusercontent.com/aws-observability/aws-otel-collector/main/  
deployment-template/ecs/aws-otel-ec2-instance-metrics-daemon-deployment-cfn.yaml
```

下载模板文件后，将其打开并将 `PATH_TO_CloudFormation_TEMPLATE` 替换为您保存模板文件的路径。然后导出以下参数并运行 Amazon CloudFormation 命令，如下命令所示。

- Cluster_Name– Amazon ECS 集群名称
- AWS_Region– 将发送数据的区域
- PATH_TO_CloudFormation_TEMPLATE– 保存 Amazon CloudFormation 模板文件的路径。
- command– 要启用 Amazon Distro for OpenTelemetry 收集器来收集 Amazon EC2 上 Amazon ECS 的实例级指标，您必须指定 `--config=/etc/ecs/otel-instance-metrics-config.yaml` 作为此参数。

```
ClusterName=Cluster_Name  
Region=AWS_Region  
command=--config=/etc/ecs/otel-instance-metrics-config.yaml  
aws cloudformation create-stack --stack-name AOCECS-${ClusterName}-${Region} \  
--template-body file://PATH_TO_CloudFormation_TEMPLATE \  
--parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \  
ParameterKey=CreateIAMRoles,ParameterValue=True \  
ParameterKey=command,ParameterValue=${command} \  
--capabilities CAPABILITY_NAMED_IAM \  
--region ${Region}
```

运行此命令后，使用 Amazon ECS 控制台查看任务是否正在运行。

对快速设置进行故障排除

要查看 Amazon CloudFormation 堆栈的状态，请输入以下命令。

```
ClusterName=cluster-name
```



```
Region=cluster-region  
aws cloudformation describe-stack --stack-name AOCECS-ClusterName-$Region --region $Region
```

如果 StackStatus 的值不是 CREATE_COMPLETE 或 CREATE_IN_PROGRESS，请查看堆栈事件以查找错误。输入以下命令。

```
ClusterName=cluster-name  
Region=cluster-region  
aws cloudformation describe-stack-events --stack-name AOCECS-ClusterName-$Region --region  
$Region
```

要查看 AOCECS 守护进程服务的状态，请输入以下命令。在输出中，您应看到部署部分中的 desiredCount 等于 runningCount。如果二者不相等，请查看输出中的故障部分。

```
ClusterName=cluster-name  
Region=cluster-region  
aws ecs describe-services --services AOCECS-daemon-service --cluster ClusterName --region  
$Region
```

您还可以使用 CloudWatch Logs 控制台来查看代理日志。查找 /aws/ecs/containerinsights/{ClusterName}/performance 日志组。

手动和自定义设置

执行此部分中的步骤，手动部署 Amazon Distro for OpenTelemetry 以从 Amazon EC2 实例上托管的 Amazon ECS 集群收集实例级别指标。

步骤 1：必要的角色和策略

需要两个 IAM 角色。如果这两个角色不存在，则必须创建它们。有关这些角色的更多信息，请参阅[创建 IAM policy](#)和[创建 IAM 角色](#)。

步骤 2：创建任务定义

创建一个任务定义并使用它将 Amazon Distro for OpenTelemetry 作为守护进程服务启动。

要使用任务定义模板创建任务定义，请按照[使用 Amazon OTel Collector 为 EC2 实例创建 ECS EC2 任务定义](#)中的说明进行操作。

要使用 Amazon ECS 控制台创建任务定义，请按照[通过 Amazon 控制台为 Amazon ECS EC2 实例指标创建任务定义来安装 Amazon OTel Collector](#)中的说明进行操作。

步骤 3：启动守护进程服务

要将 Amazon Distro for OpenTelemetry 作为守护进程服务启动，请按照[使用守护进程服务在 Amazon Elastic Container Service \(Amazon ECS\) 上运行您的任务](#)中的说明进行操作。

(可选) 高级配置

(可选) 您可以使用 SSM 来指定 Amazon EC2 实例上托管的 Amazon ECS 集群中 Amazon Distro for OpenTelemetry 的其他配置选项。有关创建配置文件的更多信息，请参阅[自定义 OpenTelemetry 配置](#)。有关在配置文件中可使用的选项的详细信息，请参阅[Amazon Container Insights Receiver](#)。

设置 FireLens 以向 CloudWatch Logs 发送日志

FireLens for Amazon ECS 使您能够使用任务定义参数将日志路由到 Amazon CloudWatch Logs，以进行日志存储和分析。FireLens 与 [Fluent Bit](#) 和 [Fluentd](#) 配合使用。我们提供 Amazon for Fluent Bit 镜像，您也可以使用自己的 Fluent Bit 或 Fluentd 镜像。支持通过 Amazon SDK、Amazon CLI 和 Amazon Web Services Management Console 使用 FireLens 配置创建 Amazon ECS 任务定义。有关 CloudWatch Logs 的更多信息，请参阅[什么是 CloudWatch Logs ?](#)。

使用 FireLens for Amazon ECS 时有一些关键注意事项。有关更多信息，请参阅[注意事项](#)。

要查找 Amazon for Fluent Bit 镜像，请参阅[使用 Amazon for Fluent Bit 镜像](#)。

要创建使用 FireLens 配置的任务定义，请参阅[创建使用 FireLens 配置的任务定义](#)。

示例

以下任务定义示例演示如何指定用于将日志转发到 CloudWatch Logs 日志组的日志配置。如需了解更多信息，请参阅 Amazon CloudWatch Logs 用户指南中的[什么是 Amazon CloudWatch Logs ?](#)

在日志配置选项中，指定日志组名称及其所在的区域。要让 Fluent Bit 代表您创建日志组，请指定 "auto_create_group": "true"。您还可以将任务 ID 指定为有助于筛选的日志流前缀。有关更多信息，请参阅[适用于 CloudWatch Logs 的 Fluent Bit 插件](#)。

```
{
  "family": "firelens-example-cloudwatch",
  "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
  "containerDefinitions": [
    {
      "essential": true,
      "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-bit:latest",
      "name": "log_router",
      "firelensConfiguration": {
        "type": "fluentbit"
      },
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "firelens-container",
          "awslogs-region": "us-west-2",
          "awslogs-create-group": "true",
          "awslogs-stream-prefix": "firelens"
        }
      },
      "memoryReservation": 50
    },
    {
      "essential": true,
      "image": "nginx",
      "name": "app",
      "logConfiguration": {
        "logDriver": "awsfirelens",
        "options": {
          "Name": "cloudwatch",
          "region": "us-west-2",
          "log_key": "log",
          "log_group_name": "/aws/ecs/containerinsights/
$(ecs_cluster)/application",
          "auto_create_group": "true",
          "log_stream_name": "$(ecs_task_id)"
        }
      },
      "memoryReservation": 100
    }
  ]
}
```

在 Amazon EKS 和 Kubernetes 上设置 Container Insights

在 Amazon EKS 或 Kubernetes 上设置 Container Insights 的整个过程如下所示：

1. 验证您是否满足所需的先决条件。

2. 将 CloudWatch 代理或 Amazon Distro for OpenTelemetry 设置为集群上的 DaemonSet，以将指标发送到 CloudWatch。

设置 Fluent Bit 或 FluentD 以将日志发送到 CloudWatch Logs。

如果您正在使用 CloudWatch 代理，您可以作为快速入门设置的一部分立即执行这些步骤，也可以单独执行这些步骤。

3. (可选) 设置 Amazon EKS 控制面板日志记录。
4. (可选) 在集群上将 CloudWatch 代理设置为 StatsD 端点以将 StatsD 指标发送到 CloudWatch。
5. (可选) 启用 App Mesh Envoy 访问日志。

主题

- [验证 先决条件 \(p. 368\)](#)
- [使用 CloudWatch 代理 \(p. 369\)](#)
- [使用 Amazon Distro for OpenTelemetry \(p. 375\)](#)
- [将日志发送到 CloudWatch Logs \(p. 376\)](#)
- [在 Amazon EKS 和 Kubernetes 上更新或删除 Container Insights \(p. 389\)](#)

验证 先决条件

在 Amazon EKS 或 Kubernetes 上安装 Container Insights 之前，请验证以下内容。无论您是使用 CloudWatch 代理还是 Amazon Distro for OpenTelemetry 在 Amazon EKS 集群上设置 Container Insights，这些先决条件都适用。

- 在其中一个支持适用于 Amazon EKS 和 Kubernetes 的 Container Insights 的区域中，您具有正常工作的 Amazon EKS 集群或 Kubernetes 集群并附加了节点。有关支持的区域列表，请参阅[使用 Container Insights \(p. 353\)](#)。
- 您已安装并正在运行 kubectl。有关更多信息，请参阅 Amazon EKS 用户指南 中的[安装 kubectl](#)。
- 如果您使用在 Amazon 上运行的 Kubernetes，而不是使用 Amazon EKS，则还需要满足以下先决条件：
 - 确保您的 Kubernetes 集群已启用基于角色的访问控制 (RBAC)。有关更多信息，请参阅《Kubernetes 参考》中的[使用 RBAC 授权](#)。
 - 您的 kubelet 已启用 Webhook 授权模式。有关更多信息，请参阅《Kubernetes 参考》中的[Kubelet 身份验证/授权](#)。
 - 您的容器运行时是 Docker。

您还必须授予 IAM 权限，以使 Amazon EKS Worker 节点能够向 CloudWatch 发送指标和日志。有两种方式可执行此操作：

- 将策略附加到 Worker 节点的 IAM 角色。这同时适用于 Amazon EKS 集群和其他 Kubernetes 集群。
- 对集群的服务账户使用 IAM 角色，并将策略附加到此角色。这仅适用于 Amazon EKS 集群。

第一个选项授予 CloudWatch 对整个节点的权限，而对服务账户使用 IAM 角色仅授予 CloudWatch 对相应守护程序集 Pod 的访问权限。

将策略附加到 Worker 节点的 IAM 角色

按照以下步骤将策略附加到 Worker 节点的 IAM 角色。这同时适用于 Amazon EKS 外部的 Amazon EKS 集群和 Kubernetes 集群。

将所需的策略附加到 Worker 节点的 IAM 角色

1. 通过以下网址打开 Amazon EC2 控制台：<https://console.aws.amazon.com/ec2/>。

2. 选择其中的一个 Worker 节点实例，然后在描述中选择 IAM 角色。
3. 在 IAM 角色页面上，选择 Attach policies (附加策略)。
4. 在策略列表中，选中 CloudWatchAgentServerPolicy 旁边的复选框。如有必要，请使用搜索框查找该策略。
5. 选择 Attach policies (附上策略)。

如果您在 Amazon EKS 外部运行 Kubernetes 集群，则可能无法将 IAM 角色附加到 Worker 节点。如果没有，您必须先将一个 IAM 角色附加到实例，然后按照前面步骤中所述添加策略。有关将角色附加到实例的更多信息，请参阅适用于 Windows 实例的 Amazon EC2 用户指南中的[将 IAM 角色附加到实例](#)。

如果您在 Amazon EKS 外部运行 Kubernetes 集群，并且希望在指标中收集 EBS 卷 ID，则必须向附加到该实例的 IAM 角色添加另一个策略。添加以下内容作为内联策略。有关更多信息，请参阅 IAM 用户指南中的[添加和删除 IAM 身份权限](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:DescribeVolumes"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

使用 IAM 服务账户角色

此方法仅适用于 Amazon EKS 集群。

使用 IAM 服务账户角色向 CloudWatch 授予权限

1. 为集群上的服务账户启用 IAM 角色 (如果尚未执行此操作)。有关更多信息，请参阅[为集群上的服务账户启用 IAM 角色](#)。
2. 如果尚未执行此操作，将服务账户配置为使用 IAM 角色。有关更多信息，请参阅[配置 Kubernetes 服务账户以代入 IAM 角色](#)。

创建角色时，除了您为角色创建的策略外，还将 CloudWatchAgentServerPolicy IAM policy 附加到角色。

3. 将 IAM 角色与集群中的服务账户相关联 (如果尚未执行此操作)。有关更多信息，请参阅[配置 Kubernetes 服务账户以代入 IAM 角色](#)。

使用 CloudWatch 代理

按照这些部分中的说明使用 CloudWatch 代理在 Amazon EKS 集群上设置 Container Insights。

主题

- [Amazon EKS 和 Kubernetes 上的 Container Insights 的快速入门设置 \(p. 369\)](#)
- [设置 CloudWatch 代理以收集集群指标 \(p. 372\)](#)

Amazon EKS 和 Kubernetes 上的 Container Insights 的快速入门设置

要完成 Container Insights 的设置，您可以按照本节中的快速入门说明进行操作。

Important

在完成本节中的步骤之前，您必须已对 IAM 权限等先决条件进行验证。有关更多信息，请参阅 [验证先决条件 \(p. 368\)](#)。

或者，您可以按照以下两节中的说明操作：[设置 CloudWatch 代理以收集集群指标 \(p. 372\)](#)和 [将日志发送到 CloudWatch Logs \(p. 376\)](#)。这些章节提供有关 CloudWatch 代理如何与 Amazon EKS 和 Kubernetes 结合使用的更多配置详细信息，但需要您执行更多安装步骤。

Note

Amazon 现已推出 Fluent Bit 作为 Container Insights 的默认日志解决方案，其性能显著提高。我们建议您使用 Fluent Bit 而不是 Fluentd。

使用 CloudWatch 代理和 Fluent Bit 快速入门

Fluent Bit 有两种配置：优化版本和更类似于 FluentD 的体验的版本。快速入门配置使用优化版本。有关 FluentD 兼容配置的更多详细信息，请参阅 [将 Fluent Bit 设置为 DaemonSet 以将日志发送到 CloudWatch Logs \(p. 376\)](#)。

要使用快速入门部署 Container Insights，请输入以下命令。

Note

默认情况下，以下设置步骤以匿名用户身份从 Docker Hub 拉取容器镜像。这种拉取可能会受到速率限制。有关更多信息，请参阅[CloudWatch 代理容器镜像 \(p. 354\)](#)。

```
ClusterName=<my-cluster-name>
RegionName=<my-cluster-region>
FluentBitHttpPort='2020'
FluentBitReadFromHead='Off'
[[ ${FluentBitReadFromHead} = 'On' ]] && FluentBitReadFromTail='Off' ||
  FluentBitReadFromTail='On'
[[ -z ${FluentBitHttpPort} ]] && FluentBitHttpServer='Off' || FluentBitHttpServer='On'
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-
mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluent-
bit-quickstart.yaml | sed 's/{{cluster_name}}/'${ClusterName}';s/
{{region_name}}/'${RegionName}';s/{{http_server_toggle}}/'${FluentBitHttpServer}';s/
{{http_server_port}}/'${FluentBitHttpPort}';s/
{{read_from_head}}/'${FluentBitReadFromHead}';s/
{{read_from_tail}}/'${FluentBitReadFromTail}';s/' | kubectl apply -f -
```

在此命令中，**my-cluster-name** 是 Amazon EKS 或 Kubernetes 集群的名称，**my-cluster-region** 是在其中发布日志的区域的名称。我们建议您使用在其中部署集群的同一个区域来降低 Amazon 出站数据传输成本。

例如，要在名为 MyCluster 的集群上部署 Container Insights 并将日志和指标发布到美国西部（俄勒冈），请输入以下命令。

```
ClusterName='MyCluster'
LogRegion='us-west-2'
FluentBitHttpPort='2020'
FluentBitReadFromHead='Off'
[[ ${FluentBitReadFromHead} = 'On' ]] && FluentBitReadFromTail='Off' ||
  FluentBitReadFromTail='On'
[[ -z ${FluentBitHttpPort} ]] && FluentBitHttpServer='Off' || FluentBitHttpServer='On'
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-
```

```
mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluent-bit-quickstart.yaml | sed 's/{{cluster_name}}/${ClusterName}"/;s/{{region_name}}/${LogRegion}"/;s/{{http_server_toggle}}/"${FluentBitHttpServer}"/;s/{{http_server_port}}/"${FluentBitHttpPort}"/;s/{{read_from_head}}/"${FluentBitReadFromHead}"/;s/{{read_from_tail}}/"${FluentBitReadFromTail}"/' | kubectl apply -f -
```

从 Fluentd 迁移

如果您已配置 Fluentd 并希望迁移到 Fluent Bit，则必须在安装 Fluent Bit 后删除 FluentD pod。可以使用以下命令删除 FluentD。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/fluentd/fluentd.yaml | kubectl delete -f -
kubectl delete configmap cluster-info -n amazon-cloudwatch
```

删除 Container Insights

如果您想使用快速入门设置删除 Container Insights，请输入以下命令。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluent-bit-quickstart.yaml | sed 's/{{cluster_name}}/${ClusterName}"/;s/{{region_name}}/${LogRegion}"/;s/{{http_server_toggle}}/"${FluentBitHttpServer}"/;s/{{http_server_port}}/"${FluentBitHttpPort}"/;s/{{read_from_head}}/"${FluentBitReadFromHead}"/;s/{{read_from_tail}}/"${FluentBitReadFromTail}"/' | kubectl delete -f -
```

使用 CloudWatch 代理和 Fluentd 快速入门

如果您已在 Kubernetes 集群中使用 Fluentd，并希望将其扩展为 Container Insights 的日志解决方案，我们会为您提供 FluentD 配置。

要使用快速入门部署 CloudWatch 代理和 Fluentd，请使用以下命令。以下设置包含社群支持的 FluentD 容器镜像。只要符合 FluentD 镜像要求，您就可以将该镜像替换为您自己的 FluentD 镜像。有关更多信息，请参阅（可选）将 FluentD 设置为 DaemonSet 以将日志发送到 CloudWatch Logs (p. 381)。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/{{cluster_name}}/cluster-name;/s/{{region_name}}/cluster-region/" | kubectl apply -f -
```

在此命令中，**cluster-name** 是 Amazon EKS 或 Kubernetes 集群的名称，**cluster-region** 是在其中发布日志的区域的名称。我们建议您使用在其中部署集群的同一个区域来降低 Amazon 出站数据传输成本。

例如，要在名为 MyCluster 的集群上部署 Container Insights 并将日志和指标发布到美国西部（俄勒冈），请输入以下命令。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/{{cluster_name}}/MyCluster;/s/{{region_name}}/us-west-2/" | kubectl apply -f -
```

删除 Container Insights

如果您想使用快速入门设置删除 Container Insights，请输入以下命令。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/{{cluster_name}}/cluster-name/;s/{{region_name}}/cluster-region/" | kubectl delete -f -
```

设置 CloudWatch 代理以收集集群指标

要设置 Container Insights 以收集指标，您可以按照[Amazon EKS 和 Kubernetes 上的 Container Insights 的快速入门设置 \(p. 369\)](#)中的步骤操作，也可以按照本节中的步骤操作。在以下步骤中，您设置 CloudWatch 代理以便能够从集群中收集指标。

步骤 1：为 CloudWatch 创建命名空间

使用以下步骤为 CloudWatch 创建名为 amazon-cloudwatch 的 Kubernetes 命名空间。如果已创建该命名空间，您可以跳过此步骤。

为 CloudWatch 创建命名空间

- 输入以下命令。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

步骤 2：在集群中创建服务账户

如果您还没有服务账户，请使用以下步骤为 CloudWatch 代理创建一个服务账户。

为 CloudWatch 代理创建服务账户

- 输入以下命令。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-serviceaccount.yaml
```

如果您没有按照前面的步骤进行操作，而是已具有要使用的 CloudWatch 代理的服务账户，则必须确保该账户具有以下规则。此外，在 Container Insights 安装的其余步骤中，您必须使用该服务账户的名称，而不是 cloudwatch-agent。

```
rules:
- apiGroups: [""]
  resources: ["pods", "nodes", "endpoints"]
  verbs: ["watch", "list"]
- apiGroups: [""]
  resources: ["nodes/proxy"]
  verbs: ["get"]
- apiGroups: [""]
  resources: ["nodes/stats", "configmaps", "events"]
  verbs: ["create"]
- apiGroups: [""]
  resources: ["configmaps"]
  resourceName: ["cwagent-clusterleader"]
  verbs: ["get", "update"]
```



```
- nonResourceURLs: ["/metrics"]  
verbs: ["get"]
```

步骤 3：为 CloudWatch 代理创建 ConfigMap

可以使用以下步骤为 CloudWatch 代理创建 ConfigMap。

为 CloudWatch 代理创建 ConfigMap

1. 运行以下命令以将 ConfigMap YAML 下载到 kubectl 客户端主机中：

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-  
insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-  
insights-monitoring/cwagent/cwagent-configmap.yaml
```

2. 编辑下载的 YAML 文件，如下所示：

- cluster_name – 在 kubernetes 部分中，将 {{cluster-name}} 替换为您的集群的名称。删除 {{}} 字符。或者，如果使用的是 Amazon EKS 集群，您可以删除 "cluster_name" 字段和值。如果这样做，CloudWatch 代理将从 Amazon EC2 标签中检测集群名称。
- 3. （可选）根据您的监控要求，对 ConfigMap 进行进一步的更改，如下所示：
 - metrics_collection_interval – 在 kubernetes 部分中，您可以指定代理收集指标的频率。默认值为 60 秒。kubelet 中的默认 cadvisor 收集间隔为 15 秒，因此，请不要将该值设置为小于 15 秒。
 - endpoint_override – 在 logs 部分中，如果要覆盖默认端点，您可以指定 CloudWatch Logs 端点。如果您从 VPC 的集群中发布并希望将数据传输到 VPC 终端节点，则可能需要执行该操作。
 - force_flush_interval – 在 logs 部分中，您可以指定在将日志事件发布到 CloudWatch Logs 之前批量处理这些事件的间隔。默认值为 5 秒。
 - region – 默认情况下，代理将指标发布到 Worker 节点所在的区域。要覆盖该区域，您可以在 agent 部分中添加 region 字段：例如，"region": "us-west-2"。
 - statsd 部分 – 如果希望 CloudWatch Logs 代理还在集群的每个 Worker 节点中作为 StatsD 侦听器运行，您可以将 statsd 部分添加到 metrics 部分中，如下示例中所示。有关该部分的其他 StatsD 选项的信息，请参阅 [使用 StatsD 检索自定义指标 \(p. 717\)](#)。

```
"metrics": {  
  "metrics_collected": {  
    "statsd": {  
      "service_address": ":8125"  
    }  
  }  
}
```

JSON 部分的完整示例如下所示。

```
{  
  "agent": {  
    "region": "us-east-1"  
  },  
  "logs": {  
    "metrics_collected": {  
      "kubernetes": {  
        "cluster_name": "MyCluster",  
        "metrics_collection_interval": 60  
      }  
    },  
    "force_flush_interval": 5,  
    "endpoint_override": "logs.us-east-1.amazonaws.com"  
  },  
}
```

```
"metrics": {
  "metrics_collected": {
    "statsd": {
      "service_address": ":8125"
    }
  }
}
```

4. 运行以下命令以在集群中创建 ConfigMap。

```
kubectl apply -f cwagent-configmap.yaml
```

步骤 4：将 CloudWatch 代理部署为 DaemonSet

要完成 CloudWatch 代理安装并开始收集容器指标，请使用以下步骤。

将 CloudWatch 代理部署为 DaemonSet

1.
 - 如果您不想在集群上使用 StatsD，请输入以下命令。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
```

- 如果您想要使用 StatsD，请执行以下步骤：
 - a. 运行以下命令以将 DaemonSet YAML 下载到 kubectl 客户端主机中。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
```

- b. 取消注释 port 文件中的 cwagent-daemonset.yaml 部分，如下所示：

```
ports:
- containerPort: 8125
  hostPort: 8125
  protocol: UDP
```

- c. 运行以下命令以在集群中部署 CloudWatch 代理。

```
kubectl apply -f cwagent-daemonset.yaml
```

2. 运行以下命令以验证是否部署了该代理。

```
kubectl get pods -n amazon-cloudwatch
```

在完成后，CloudWatch 代理创建一个名为 `/aws/containerinsights/Cluster_Name/performance` 的日志组，并将性能日志事件发送到该日志组。如果还将该代理设置为 StatsD 侦听器，该代理还会在端口 8125（具有计划运行应用程序 pod 的节点的 IP 地址）上侦听 StatsD 指标。

问题排查

如果未正确部署该代理，请尝试执行以下操作：

- 运行以下命令以获取 pod 列表。

```
kubectl get pods -n amazon-cloudwatch
```

- 运行以下命令并在输出底部检查事件。

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

- 运行以下命令以检查日志。

```
kubectl logs pod-name -n amazon-cloudwatch
```

使用 Amazon Distro for OpenTelemetry

您可以通过使用 Amazon Distro for OpenTelemetry 收集器设置 Container Insights 从 Amazon EKS 集群收集指标。有关 Amazon Distro for OpenTelemetry 的更多信息，请参阅 [Amazon Distro for OpenTelemetry](#)。

如何设置 Container Insights 取决于集群托管在 Amazon EC2 实例上还是 Amazon Fargate (Fargate) 上。

托管在 Amazon EC2 上的 Amazon EKS 集群

如果您尚未执行此操作，请确保您已满足先决条件，例如必要的 IAM 角色。有关更多信息，请参阅[验证先决条件](#) (p. 368)。

Amazon 提供 Helm Chart，您可以用它来设置对 Amazon EC2 上的 Amazon Elastic Kubernetes Service 的监控。此监控使用 Amazon Distro for OpenTelemetry (ADOT) 收集器用于指标，使用 Fluent Bit 用于日志。因此，Helm Chart 对使用 Amazon EC2 上的 Amazon EKS 并希望收集指标和日志以发送到 CloudWatch Container Insights 的客户非常有用。有关此 Helm Chart 的更多信息，请参阅[发送到 Amazon CloudWatch Container Insights 的 EC2 上 EKS 的指标和日志的 ADOT Helm chart](#)。

或者，您也可以使用本部分剩余内容中的说明。

首先，通过输入以下命令将 Amazon Distro for OpenTelemetry 收集器部署为 DaemonSet。

```
curl https://raw.githubusercontent.com/aws-observability/aws-otel-collector/main/
deployment-template/eks/otel-container-insights-infra.yaml |
kubectl apply -f -
```

要确认该收集器正在运行，请输入以下命令。

```
kubectl get pods -l name=aws-otel-eks-ci -n aws-otel-eks
```

如果此命令的输出包含处于 Running 状态的多个 Pod，则收集器正在运行并从集群收集指标。收集器会创建一个名为 `aws/containerinsights/cluster-name/performance` 的日志组并将性能日志事件发送给它。

有关如何在 CloudWatch 中查看 Container Insights 指标的信息，请参阅 [查看 Container Insights 指标](#) (p. 390)。

Amazon 还在 GitHub 上为此情况提供了文档。如果要自定义 Container Insights 发布的指标和日志，请参阅 <https://aws-otel.github.io/docs/getting-started/container-insights/eks-infra>。

托管在 Fargate 上的 Amazon EKS 集群

有关如何配置和部署 ADOT 收集器以从部署到 Fargate 上的 Amazon EKS 集群的工作负载收集系统指标并将其发送到 CloudWatch Container Insights 的说明，请参阅 Amazon Distro for OpenTelemetry 文档中的 [Container Insights EKS Fargate](#)。

将日志发送到 CloudWatch Logs

要将日志从您的容器发送到 Amazon CloudWatch Logs，您可以使用 Fluent Bit 或 Fluentd。有关更多信息，请参阅 [Fluent Bit](#) 和 [Fluentd](#)。

如果您尚未使用 Fluentd，出于以下原因，我们建议您使用 Fluent Bit：

- 与 FluentD 相比，Fluent Bit 具有更小的资源占用空间，并且在内存和 CPU 使用率方面具有更高的资源效率。有关更详细比较，请参阅 [Fluent Bit 与 Fluentd 性能比较 \(p. 376\)](#)。
- Fluent Bit 镜像由 Amazon 开发和维护。这使得 Amazon 能够采用新的 Fluent Bit 镜像功能并更快地响应问题。

主题

- [Fluent Bit 与 Fluentd 性能比较 \(p. 376\)](#)
- [将 Fluent Bit 设置为 DaemonSet 以将日志发送到 CloudWatch Logs \(p. 376\)](#)
- (可选) [将 FluentD 设置为 DaemonSet 以将日志发送到 CloudWatch Logs \(p. 381\)](#)
- (可选) [设置 Amazon EKS 控制面板日志记录 \(p. 386\)](#)
- (可选) [启用 App Mesh Envoy 访问日志 \(p. 386\)](#)
- (可选) [为大型集群启用 Use_Kubelet 功能 \(p. 386\)](#)

Fluent Bit 与 Fluentd 性能比较

下表显示了 Fluent Bit 在内存和 CPU 使用率方面优于 FluentD 的性能优势。以下数字仅供参考，可能会因环境而异。

每秒日志数	Fluentd CPU 使用率	Fluent Bit CPU 使用率与 Fluentd 兼容配置	具有优化配置的 Fluent Bit CPU 使用率
100	0.35 vCPU	0.02 vCPU	0.02 vCPU
1000	0.32 vCPU	0.14 vCPU	0.11 vCPU
5000	0.85 vCPU	0.48 vCPU	0.30 vCPU
10000	0.94 vCPU	0.60 vCPU	0.39 vCPU

每秒日志数	Fluentd 内存使用量	Fluent Bit 内存使用量与 Fluentd 兼容配置	优化配置的 Fluent Bit 内 存使用量
100	153MB	46MB	37MB
1000	270MB	45MB	40MB
5000	320MB	55MB	45MB
10000	375MB	92MB	75MB

将 Fluent Bit 设置为 DaemonSet 以将日志发送到 CloudWatch Logs

以下部分帮助您部署 Fluent Bit 以将日志从容器发送到 CloudWatch Logs。

主题

- [您在使用 Fluentd 时出现的差异 \(p. 377\)](#)

- [设置 Fluent Bit \(p. 377\)](#)
- [多行日志支持 \(p. 379\)](#)
- [\(可选 \) 从 Fluent Bit 减少日志卷 \(p. 380\)](#)
- [问题排查 \(p. 381\)](#)
- [控制面板 \(p. 381\)](#)

您在使用 Fluentd 时出现的差异

如果您已经使用 Fluentd 将日志从容器发送到 CloudWatch Logs，请阅读本节以了解 Fluentd 和 Fluent Bit 之间的区别。如果您尚未将 Fluentd 与 Container Insights 结合使用，则可以跳至 [设置 Fluent Bit \(p. 377\)](#)。

我们为 Fluent Bit 提供了两种默认配置：

- Fluent Bit 优化配置 – 符合 Fluent Bit 最佳实践的配置。
- Fluentd 兼容配置 – 尽可能与 Fluentd 行为保持一致的配置。

下面的列表详细说明了 Fluentd 和每个 Fluent Bit 配置之间的差异。

- 日志流名称的差异 – 如果您使用 Fluent Bit 优化配置，日志流名称将有所不同。

在 `/aws/containerinsights/Cluster_Name/application` 下

- Fluent Bit 优化配置将日志发送到 `kubernetes-nodeName-application.var.log.containers.kubernetes-podName_kubernetes-namespace_kubernetes-container-name-kubernetes-containerID`
- Fluentd 将日志发送到 `kubernetes-podName_kubernetes-namespace_kubernetes-containerName_kubernetes-containerID`

在 `/aws/containerinsights/Cluster_Name/host` 下

- Fluent Bit 优化配置将日志发送到 `kubernetes-nodeName.host-log-file`
- Fluentd 将日志发送到 `host-log-file-Kubernetes-NodePrivateIp`

在 `/aws/containerinsights/Cluster_Name/dataplane` 下

- Fluent Bit 优化配置将日志发送到 `kubernetes-nodeName.dataplaneServiceLog`
- Fluentd 将日志发送到 `dataplaneServiceLog-Kubernetes-nodeName`
- Container Insights 写入的 kube-proxy 和 aws-node 日志文件位于不同的位置。在 FluentD 配置中，它们位于 `/aws/containerinsights/Cluster_Name/application`。在 Fluent Bit 优化配置中，它们位于 `/aws/containerinsights/Cluster_Name/dataplane`。
- Fluent Bit 和 Fluentd 中大部分元数据如 `pod_name` 和 `namespace_name` 是相同的，但以下是不同的。
 - Fluent Bit 优化配置使用 `docker_id`，而 Fluentd 使用 `Docker.container_id`。
 - 两种 Fluent Bit 配置均不使用以下元数据。它们仅存在于 Fluentd 中：`container_image_id`、`master_url`、`namespace_id` 和 `namespace_labels`。

设置 Fluent Bit

要设置 Fluent Bit 以从容器中收集日志，您可以按照 [Amazon EKS 和 Kubernetes 上的 Container Insights 的快速入门设置 \(p. 369\)](#) 中的步骤操作，也可以按照本节中的步骤操作。

无论使用哪种方法，附加到集群节点的 IAM 角色必须具有足够的权限。有关运行 Amazon EKS 集群所需权限的更多信息，请参阅 Amazon EKS 用户指南中的 [Amazon EKS IAM 策略、角色和权限](#)。

在以下步骤中，您将 Fluent Bit 设置为 daemonSet 以将日志发送到 CloudWatch Logs。在完成该步骤时，Fluent Bit 将创建以下日志组（如果尚不存在）。

日志组名称	日志源
/aws/containerinsights/ <i>Cluster_Name</i> /application	/var/log/containers 中的所有日志文件
/aws/containerinsights/ <i>Cluster_Name</i> /host	/var/log/dmesg、/var/log/secure 和 /var/log/messages 中的日志
/aws/containerinsights/ <i>Cluster_Name</i> /dataplane	kubelet.service、kubeproxy.service 和 docker.service 的 /var/log/journal 中的日志。

安装 Fluent Bit 以将日志从容器发送到 CloudWatch Logs

1. 如果您还没有名为 amazon-cloudwatch 的命名空间，请通过输入以下命令创建一个：

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

2. 运行以下命令以创建一个名为 cluster-info 的 ConfigMap，该 ConfigMap 以集群名称和要向其发送日志的区域命名。将 *cluster-name* 和 *cluster-region* 分别替换为您的集群的名称和区域。

```
ClusterName=cluster-name
RegionName=cluster-region
FluentBitHttpPort='2020'
FluentBitReadFromHead='Off'
[[ ${FluentBitReadFromHead} = 'On' ]] && FluentBitReadFromTail='Off' ||
  FluentBitReadFromTail='On'
[[ -z ${FluentBitHttpPort} ]] && FluentBitHttpServer='Off' || FluentBitHttpServer='On'
kubectl create configmap fluent-bit-cluster-info \
--from-literal=cluster.name=${ClusterName} \
--from-literal=http.server=${FluentBitHttpServer} \
--from-literal=http.port=${FluentBitHttpPort} \
--from-literal=read.head=${FluentBitReadFromHead} \
--from-literal=read.tail=${FluentBitReadFromTail} \
--from-literal=logs.region=${RegionName} -n amazon-cloudwatch
```

在此命令中，默认情况下，用于监控插件指标的 FluentBitHttpServer 处于启用状态。要将其关闭，请将命令中的第三行更改为命令中的 FluentBitHttpPort=''（空字符串）。

同样，默认情况下，Fluent Bit 从尾部读取日志文件，并在部署后仅捕获新日志。如果你想要相反的设置，请设置 FluentBitReadFromHead='On'，它将收集文件系统中的所有日志。

3. 运行以下任一命令，将 Fluent Bit daemonset 下载并部署到集群中。

- 如果需要 Fluent Bit 优化配置，请运行此命令。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-bit.yaml
```

- 如果需要更类似于 Fluentd 的 Fluent Bit 配置，请运行此命令。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-bit-compatible.yaml
```

4. 运行以下命令以验证部署。每个节点应具有一个名为 fluent-bit-* 的 pod。

```
kubectl get pods -n amazon-cloudwatch
```

上述步骤在集群中创建了以下资源：

- amazon-cloudwatch 命名空间中名为 Fluent-Bit 的服务账户。该服务账户用于运行 Fluent Bit daemonSet。有关更多信息，请参阅《Kubernetes 参考》中的[管理服务账户](#)。
- amazon-cloudwatch 命名空间中名为 Fluent-Bit-role 的集群角色。该集群角色为 Fluent-Bit 服务账户授予有关 pod 日志的 get、list 和 watch 权限。有关更多信息，请参阅《Kubernetes 参考》中的[API 概述](#)。
- amazon-cloudwatch 命名空间中的名为 Fluent-Bit-config 的 ConfigMap。该 ConfigMap 包含 Fluent Bit 使用的配置。有关更多信息，请参阅《Kubernetes 任务》文档中的[配置 Pod 以使用 ConfigMap](#)。

如果要验证 Fluent Bit 设置，请按照下列步骤操作。

验证 Fluent Bit 设置

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 确保您位于部署 Fluent Bit 的区域中。
4. 检查区域中的日志组列表。您将看到以下内容：
 - /aws/containerinsights/*Cluster_Name*/application
 - /aws/containerinsights/*Cluster_Name*/host
 - /aws/containerinsights/*Cluster_Name*/dataplane
5. 导航到其中一个日志组，并检查日志流的 Last Event Time (上次事件时间)。如果相对于您部署 Fluent Bit 的时间，它是最近的，则会验证设置。

创建 /dataplane 日志组时，可能会稍有延迟。这是正常的，因为只有在 Fluent Bit 开始为该日志组发送日志时才会创建这些日志组。

多行日志支持

默认情况下，多行日志条目启动器是任何没有空格的字符。这意味着，以不具有空格的字符开头的所有日志行都被视为新的多行日志条目。

如果您自己的应用程序日志使用不同的多行启动器，则可以通过在 Fluent-Bit.yaml 文件中进行两项更改来支持它们。

首先，通过将日志文件的路径名添加到 Fluent-Bit.yaml 的 containers 部分的 exclude_path 字段中，将它们从默认输入中排除。以下是示例。

```
[INPUT]
  Name      tail
  Tag       application.*
  Exclude_Path  full_pathname_of_log_file*, full_pathname_of_log_file2*
  Path      /var/log/containers/*.log
```

接下来，将日志文件的数据块添加到 Fluent-Bit.yaml 文件中。请参阅下面的 cloudwatch-agent 日志配置示例，该示例使用时间戳正则表达式作为多行启动器。

```
application-log.conf: |
```



```
[INPUT]
  Name          tail
  Tag           application.*
  Path          /var/log/containers/cloudwatch-agent*
  Docker_Mode   On
  Docker_Mode_Flush 5
  Docker_Mode_Parser cwagent_firstline
  Parser        docker
  DB            /fluent-bit/state/flb_cwagent.db
  Mem_Buf_Limit 5MB
  Skip_Long_Lines On
  Refresh_Interval 10

parsers.conf: |
  [PARSER]
    Name          cwagent_firstline
    Format        regex
    Regex         (?<log>(?"log":")\d{4}[\/-]\d{1,2}[\/-]\d{1,2}[ T]\d{2}:\d{2}:\d{2}(?!\\.)*)(?"!\\".*(?<stream>(?"stream":")*.*)"*(?"<time>\d{4}-\d{1,2}-\d{1,2}T\d{2}:\d{2}:\d{2}\.\w*).*?(?={})
    Time_Key      time
    Time_Format   %Y-%m-%dT%H:%M:%S.%LZ
```

(可选) 从 Fluent Bit 减少日志卷

默认情况下，我们将 Fluent Bit 应用程序日志和 Kubernetes 元数据发送到 CloudWatch。如果您要减少发送到 CloudWatch 的数据量，可以停止将这些数据源中的一个或两个发送到 CloudWatch。

要停止 Fluent Bit 应用程序日志，请从 `Fluent-Bit.yaml` 文件中删除以下部分。

```
[INPUT]
  Name          tail
  Tag           application.*
  Path          /var/log/containers/fluent-bit*
  Parser        docker
  DB            /fluent-bit/state/flb_log.db
  Mem_Buf_Limit 5MB
  Skip_Long_Lines On
  Refresh_Interval 10
```

要取消将 Kubernetes 元数据附加到发送到 CloudWatch 的日志事件，请在 `Fluent-Bit.yaml` 文件的 `application-log.conf` 部分中添加以下筛选条件。

```
application-log.conf: |
  [FILTER]
    Name          nest
    Match         application.*
    Operation      lift
    Nested_under   kubernetes
    Add_prefix     Kube.

  [FILTER]
    Name          modify
    Match         application.*
    Remove        Kube.<Metadata_1>
    Remove        Kube.<Metadata_2>
    Remove        Kube.<Metadata_3>

  [FILTER]
    Name          nest
    Match         application.*
    Operation      nest
    Wildcard       Kube.*
```

Nested_under	kubernetes
Remove_prefix	Kube.

问题排查

如果您没有看到这些日志组并且查看的是正确区域，请检查 Fluent Bit daemonSet pod 日志以查找错误。

运行以下命令，并确保状态为 Running。

```
kubectl get pods -n amazon-cloudwatch
```

如果日志具有与 IAM 权限相关的错误，请检查附加到集群节点的 IAM 角色。有关运行 Amazon EKS 集群所需权限的更多信息，请参阅 Amazon EKS 用户指南中的 [Amazon EKS IAM 策略、角色和权限](#)。

如果 pod 状态为 CreateContainerConfigError，请运行以下命令以获取确切的错误。

```
kubectl describe pod pod_name -n amazon-cloudwatch
```

控制面板

您可以创建一个控制面板来监控每个正在运行的插件的指标。您可以查看输入和输出字节以及记录处理率以及输出错误和重试/失败率的数据。要查看这些指标，您需要为 Amazon EKS 和 Kubernetes 集群安装带有 Prometheus 指标集合的 CloudWatch 代理。有关如何设置控制面板的更多信息，请参阅 [在 Amazon EKS 和 Kubernetes 集群上安装带有 Prometheus 指标收集功能的 CloudWatch 代理](#) (p. 455)。

Note

在设置此控制面板之前，您必须为 Prometheus 指标设置 Container Insights。有关更多信息，请参阅 [Container Insights Prometheus 指标监控](#) (p. 418)。

为 Fluent Bit Prometheus 指标创建控制面板

1. 创建环境变量，替换以下行中右侧的值以匹配您的部署。

```
DASHBOARD_NAME=your_cw_dashboard_name
REGION_NAME=your_metric_region_such_as_us-west-1
CLUSTER_NAME=your_kubernetes_cluster_name
```

2. 通过运行以下命令来创建控制面板。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/sample_cloudwatch_dashboards/fluent-bit/cw_dashboard_fluent_bit.json \
| sed "s/{YOUR_AWS_REGION}/{REGION_NAME}/g" \
| sed "s/{YOUR_CLUSTER_NAME}/{CLUSTER_NAME}/g" \
| xargs -0 aws cloudwatch put-dashboard --dashboard-name ${DASHBOARD_NAME} --dashboard-body
```

(可选) 将 FluentD 设置为 DaemonSet 以将日志发送到 CloudWatch Logs

Warning

Container Insights 对 FluentD 的支持目前处于维护模式，这意味着 Amazon 不会为 FluentD 提供任何进一步的更新，我们计划在不久的将来将其弃用。此外，Container Insights 的当前 FluentD 配置使用的是旧版本的 FluentD 镜像 `fluent/fluentd-kubernetes-daemonset:v1.7.3-debian-cloudwatch-1.0`，该版本没有最新的改进和安全补丁。有关开源社群支持的最新 FluentD 镜像，请参阅 [fluentd-kubernetes-daemonset](#)。

如果可能，我们强烈建议您迁移以将 FluentBit 与 Container Insights 结合使用。使用 FluentBit 作为 Container Insights 的日志转发器，可显著提高性能。
有关更多信息，请参阅 [将 Fluent Bit 设置为 DaemonSet 以将日志发送到 CloudWatch Logs](#) (p. 376) 和 [您在使用 Fluentd 时出现的差异](#) (p. 377)。

要设置 FluentD 以从容器中收集日志，您可以按照 [Amazon EKS 和 Kubernetes 上的 Container Insights 的快速入门设置](#) (p. 369) 中的步骤操作，也可以按照本节中的步骤操作。在以下步骤中，您将 FluentD 设置为 DaemonSet 以将日志发送到 CloudWatch Logs。在完成该步骤时，FluentD 将创建以下日志组（如果尚不存在）。

日志组名称	日志源
/aws/containerinsights/ <i>Cluster_Name</i> /application	/var/log/containers 中的所有日志文件
/aws/containerinsights/ <i>Cluster_Name</i> /host	/var/log/dmesg、/var/log/secure 和 /var/log/messages 中的日志
/aws/containerinsights/ <i>Cluster_Name</i> /dataplane	kubelet.service、kubeproxy.service 和 docker.service 的 /var/log/journal 中的日志。

步骤 1：为 CloudWatch 创建命名空间

使用以下步骤为 CloudWatch 创建名为 amazon-cloudwatch 的 Kubernetes 命名空间。如果已创建该命名空间，您可以跳过此步骤。

为 CloudWatch 创建命名空间

- 输入以下命令。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

步骤 2：安装 FluentD

下载 FluentD 以开始执行该过程。在完成这些步骤时，部署将在集群上创建以下资源：

- amazon-cloudwatch 命名空间中名为 fluentd 的服务账户。该服务账户用于运行 FluentD DaemonSet。有关更多信息，请参阅《Kubernetes 参考》中的[管理服务账户](#)。
- amazon-cloudwatch 命名空间中名为 fluentd 的集群角色。该集群角色为 fluentd 服务账户授予有关 pod 日志的 get、list 和 watch 权限。有关更多信息，请参阅《Kubernetes 参考》中的[API 概述](#)。
- amazon-cloudwatch 命名空间中的名为 fluentd-config 的 ConfigMap。该 ConfigMap 包含由 FluentD 使用的配置。有关更多信息，请参阅《Kubernetes 任务》文档中的[配置 Pod 以使用 ConfigMap](#)。

安装 FluentD

- 使用集群名称和日志将发送到的 Amazon 区域创建一个名为 cluster-info 的 ConfigMap。运行以下命令，并使用您的集群和区域名称更新占位符。

```
kubectl create configmap cluster-info \
--from-literal=cluster.name=cluster_name \
--from-literal=logs.region=region_name -n amazon-cloudwatch
```

- 运行以下命令将 FluentD DaemonSet 下载并部署到集群中。确保您使用的是具有正确架构的容器镜像。示例清单仅适用于 x86 实例，并将输入 CrashLoopBackOff (如果您的集群中有高级 RISC 机器 (ARM) 实例)。FluentD daemonSet 没有官方的多架构 Docker 镜像，不允许您对多个底层镜像使用一个标签，并让容器运行时拉取正确标签。FluentD ARM 镜像使用带有 arm64 后缀的不同标签。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/fluentd/fluentd.yaml
```

Note

由于最近进行了更改，以优化 FluentD 配置并最大限度地减少 FluentD API 请求对 Kubernetes API 端点的影响，因此默认情况下，Kubernetes 筛选条件的“Watch (监视)”选项已禁用。有关更多详细信息，请参阅 [fluent-plugin-kubernetes_metadata_filter](#)。

- 运行以下命令以验证部署。每个节点应具有一个名为 fluentd-cloudwatch-* 的 pod。

```
kubectl get pods -n amazon-cloudwatch
```

步骤 3：验证 FluentD 设置

要验证您的 FluentD 设置，请使用以下步骤。

验证 Container Insights 的 FluentD 设置

- 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
- 在导航窗格中，选择 Log groups (日志组)。确保您位于将 FluentD 部署到您的容器的区域中。

在该区域上的日志组列表中，您将会看到以下内容：

- /aws/containerinsights/**Cluster_Name**/application
- /aws/containerinsights/**Cluster_Name**/host
- /aws/containerinsights/**Cluster_Name**/dataplane

如果看到这些日志组，则已验证 FluentD 设置。

多行日志支持

2019 年 8 月 19 日，我们为 FluentD 收集的日志添加了多行日志支持。

默认情况下，多行日志条目启动器是任何没有空格的字符。这意味着，以不具有空格的字符开头的所有日志行都被视为新的多行日志条目。

如果您自己的应用程序日志使用不同的多行启动器，则可以通过在 fluentd.yaml 文件中进行两项更改来支持它们。

首先，通过将日志文件的路径名添加到 fluentd.yaml 的 containers 部分的 exclude_path 字段中，将它们从默认的多行支持中排除。以下是示例。

```
<source>
  @type tail
  @id in_tail_container_logs
  @label @containers
  path /var/log/containers/*.log
  exclude_path ["full_pathname_of_log_file*", "full_pathname_of_log_file2*"]
```

接下来，将日志文件的数据块添加到 `fluentd.yaml` 文件中。以下示例用于 CloudWatch 代理的日志文件，该文件使用时间戳正则表达式作为多行启动器。您可以复制此数据块并将其添加到 `fluentd.yaml`。更改指示的行以反映您要使用的应用程序日志文件名和多行启动器。

```
<source>
  @type tail
  @id in_tail_cwagent_logs
  @label @cwagentlogs
  path /var/log/containers/cloudwatch-agent*
  pos_file /var/log/cloudwatch-agent.log.pos
  tag *
  read_from_head true
<parse>
  @type json
  time_format %Y-%m-%dT%H:%M:%S.%NZ
</parse>
</source>
```

```
<label @cwagentlogs>
  <filter *>
    @type kubernetes_metadata
    @id filter_kube_metadata_cwagent
  </filter>

  <filter *>
    @type record_transformer
    @id filter_cwagent_stream_transformer
    <record>
      stream_name ${tag_parts[3]}
    </record>
  </filter>

  <filter *>
    @type concat
    key log
    multiline_start_regexp /\d{4}[-/]\d{1,2}[-/]\d{1,2}/
    separator ""
    flush_interval 5
    timeout_label @NORMAL
  </filter>

  <match *>
    @type relabel
    @label @NORMAL
  </match>
</label>
```

(可选) 从 FluentD 减少日志卷

默认情况下，我们将 FluentD 应用程序日志和 Kubernetes 元数据发送到 CloudWatch。如果您要减少发送到 CloudWatch 的数据量，可以停止将这些数据源中的一个或两个发送到 CloudWatch。

要停止 FluentD 应用程序日志，请从 `fluentd.yaml` 文件中删除以下部分。

```
<source>
  @type tail
  @id in_tail_fluentd_logs
  @label @fluentdlogs
  path /var/log/containers/fluentd*
  pos_file /var/log/fluentd.log.pos
```

```
tag *
read_from_head true
<parse>
  @type json
  time_format %Y-%m-%dT%H:%M:%S.%NZ
</parse>
</source>

<label @fluentdlogs>
  <filter **>
    @type kubernetes_metadata
    @id filter_kube_metadata_fluentd
  </filter>

  <filter **>
    @type record_transformer
    @id filter_fluentd_stream_transformer
    <record>
      stream_name ${tag_parts[3]}
    </record>
  </filter>

  <match **>
    @type relabel
    @label @NORMAL
  </match>
</label>
```

要取消将 Kubernetes 元数据附加到发送到 CloudWatch 的日志事件，请在 `fluentd.yaml` 文件的 `record_transformer` 部分中添加一行。在要删除此元数据的日志源中，添加以下行。

```
remove_keys $.kubernetes.pod_id, $.kubernetes.master_url, $.kubernetes.container_image_id,
$.kubernetes.namespace_id
```

例如：

```
<filter **>
  @type record_transformer
  @id filter_containers_stream_transformer
  <record>
    stream_name ${tag_parts[3]}
  </record>
  remove_keys $.kubernetes.pod_id, $.kubernetes.master_url,
$.kubernetes.container_image_id, $.kubernetes.namespace_id
</filter>
```

问题排查

如果您没有看到这些日志组并且查看的是正确区域，请检查 FluentD DaemonSet pod 日志以查找错误。

运行以下命令，并确保状态为 `Running`。

```
kubectl get pods -n amazon-cloudwatch
```

在上一命令的结果中，记下以 `fluentd-cloudwatch` 开头的 pod 名称。在以下命令中使用该 pod 名称。

```
kubectl logs pod_name -n amazon-cloudwatch
```

如果日志具有与 IAM 权限相关的错误，请检查附加到集群节点的 IAM 角色。有关运行 Amazon EKS 集群所需权限的更多信息，请参阅 Amazon EKS 用户指南中的 [Amazon EKS IAM 策略、角色和权限](#)。

如果 pod 状态为 `CreateContainerConfigError`，请运行以下命令以获取确切的错误。

```
kubectl describe pod pod_name -n amazon-cloudwatch
```

如果 pod 状态为 `CrashLoopBackOff`，请确保 Fluentd 容器镜像的架构与您安装 Fluentd 时的节点相同。如果您的集群同时具有 x86 和 ARM64 节点，您可以使用 `kubernetes.io/arch` 标签将镜像放置在正确的节点上。有关更多信息，请参阅 kubernetes.io/arch。

(可选) 设置 Amazon EKS 控制面板日志记录

如果使用的是 Amazon EKS，您可以选择启用 Amazon EKS 控制面板日志记录，以直接从 Amazon EKS 控制面板中向 CloudWatch Logs 提供审计和诊断日志。有关更多信息，请参阅 [Amazon EKS 控制面板日志记录](#)。

(可选) 启用 App Mesh Envoy 访问日志

您可以设置 Container Insights FluentD 以将 App Mesh Envoy 访问日志发送到 CloudWatch Logs。有关更多信息，请参阅 [日志记录](#)。

将 Envoy 访问日志发送到 CloudWatch Logs

1. 在集群中设置 FluentD。有关更多信息，请参阅 [\(可选\) 将 FluentD 设置为 DaemonSet 以将日志发送到 CloudWatch Logs \(p. 381\)](#)。
2. 为您的虚拟节点配置 Envoy 访问日志。有关说明，请参阅 [日志记录](#)。请确保在每个虚拟节点中将日志路径配置为 `/dev/stdout`。

完成后，Envoy 访问日志将发送到 `/aws/containerinsights/Cluster_Name/application` 日志组。

(可选) 为大型集群启用 Use_Kubelet 功能

默认情况下，在 FluentBit Kubernetes 插件中禁用 Use_Kubelet 功能。启用此功能可以减少 API 服务器的流量，并缓解 API 服务器成为瓶颈的问题。我们建议您为大型集群启用此功能。

要启用 Use_Kubelet，请先将节点和节点/代理权限添加到 clusterRole config 中。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: fluent-bit-role
rules:
  - nonResourceURLs:
    - /metrics
    verbs:
    - get
  - apiGroups: [""]
    resources:
    - namespaces
    - pods
    - pods/logs
    - nodes
    - nodes/proxy
    verbs: ["get", "list", "watch"]
```

在 DaemonSet 配置中，此功能需要主机网络访问权限。适用于 `amazon/aws-for-fluent-bit` 的镜像版本应为 2.12.0 或更高版本，或者 `fluent bit` 镜像版本应为 1.7.2 或更高版本。

```
apiVersion: apps/v1
kind: DaemonSet
```



```

metadata:
  name: fluent-bit
  namespace: amazon-cloudwatch
  labels:
    k8s-app: fluent-bit
    version: v1
    kubernetes.io/cluster-service: "true"
spec:
  selector:
    matchLabels:
      k8s-app: fluent-bit
  template:
    metadata:
      labels:
        k8s-app: fluent-bit
        version: v1
        kubernetes.io/cluster-service: "true"
    spec:
      containers:
      - name: fluent-bit
        image: amazon/aws-for-fluent-bit:2.19.0
        imagePullPolicy: Always
        env:
          - name: AWS_REGION
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: logs.region
          - name: CLUSTER_NAME
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: cluster.name
          - name: HTTP_SERVER
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: http.server
          - name: HTTP_PORT
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: http.port
          - name: READ_FROM_HEAD
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: read.head
          - name: READ_FROM_TAIL
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: read.tail
          - name: HOST_NAME
            valueFrom:
              fieldRef:
                fieldPath: spec.nodeName
          - name: HOSTNAME
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: metadata.name
          - name: CI_VERSION
            value: "k8s/1.3.8"
      resources:
        limits:

```

```

        memory: 200Mi
      requests:
        cpu: 500m
        memory: 100Mi
    volumeMounts:
    # Please don't change below read-only permissions
    - name: fluentbitstate
      mountPath: /var/fluent-bit/state
    - name: varlog
      mountPath: /var/log
      readOnly: true
    - name: varlibdockercontainers
      mountPath: /var/lib/docker/containers
      readOnly: true
    - name: fluent-bit-config
      mountPath: /fluent-bit/etc/
    - name: runlogjournal
      mountPath: /run/log/journal
      readOnly: true
    - name: dmesg
      mountPath: /var/log/dmesg
      readOnly: true
    terminationGracePeriodSeconds: 10
    hostNetwork: true
    dnsPolicy: ClusterFirstWithHostNet
    volumes:
    - name: fluentbitstate
      hostPath:
        path: /var/fluent-bit/state
    - name: varlog
      hostPath:
        path: /var/log
    - name: varlibdockercontainers
      hostPath:
        path: /var/lib/docker/containers
    - name: fluent-bit-config
      configMap:
        name: fluent-bit-config
    - name: runlogjournal
      hostPath:
        path: /run/log/journal
    - name: dmesg
      hostPath:
        path: /var/log/dmesg
    serviceAccountName: fluent-bit
    tolerations:
    - key: node-role.kubernetes.io/master
      operator: Exists
      effect: NoSchedule
    - operator: "Exists"
      effect: "NoExecute"
    - operator: "Exists"
      effect: "NoSchedule"

```

Kubernetes 插件配置应该类似于以下内容：

```

[FILTER]
    Name          kubernetes
    Match          application.*
    Kube_URL       https://kubernetes.default.svc:443
    Kube_Tag_Prefix application.var.log.containers.
    Merge_Log      On
    Merge_Log_Key  log_processed
    K8S-Logging.Parser On
    K8S-Logging.Exclude Off

```

Labels	Off
Annotations	Off
Use_Kubelet	On
Kubelet_Port	10250
Buffer_Size	0

在 Amazon EKS 和 Kubernetes 上更新或删除 Container Insights

使用这些章节中的步骤更新 CloudWatch 代理容器镜像，或从 Amazon EKS 或 Kubernetes 集群中删除 Container Insights。

主题

- [更新 CloudWatch 代理容器镜像 \(p. 389\)](#)
- [删除 Container Insights 的 CloudWatch 代理和 FluentD \(p. 390\)](#)

更新 CloudWatch 代理容器镜像

如果需要将容器映像更新到最新版本，请使用本节中的步骤。

更新您的容器映像

1. 输入以下命令，应用最新的 `cwagent-serviceaccount.yaml` 文件。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-serviceaccount.yaml
```

2. 此步骤仅适用于从 2019 年 8 月 20 日发布的 1.226589.0 之前的版本升级其容器化 CloudWatch 代理的客户。

在 Configmap 文件 `cwagentconfig` 中，将关键字 `structuredlogs` 更改为 `logs`

- a. 首先，输入以下命令在编辑模式下打开现有的 `cwagentconfig`。

```
kubectl edit cm cwagentconfig -n amazon-cloudwatch
```

在文件中，如果您看到关键字 `structuredlogs`，请将其更改为 `logs`

- b. 输入 **wq** 以保存该文件并退出编辑模式。
3. 输入以下命令，应用最新的 `cwagent-daemonset.yaml` 文件。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
```

您可以在更改 `cwagent-configmap.yaml` 中的配置时随时实现 CloudWatch 代理 DaemonSet 的滚动更新。要实现滚动更新，您必须确保 `cwagent-daemonset.yaml` 文件中的 `.spec.template` 部分已相应更改。否则，Kubernetes 将 DaemonSet 视为未更改。通常的做法是将 ConfigMap 的哈希值添加到 `.spec.template.metadata.annotations.configHash` 中，如下示例中所示。

```
yq w -i cwagent-daemonset.yaml spec.template.metadata.annotations.configHash $(kubectl get cm/cwagentconfig -n amazon-cloudwatch -o yaml | sha256sum)
```

这会在 `cwagent-daemonset.yaml` 文件中添加哈希值，如下示例中所示。

```
spec:
```

```
selector:
  matchLabels:
    name: cloudwatch-agent
template:
  metadata:
    labels:
      name: cloudwatch-agent
    annotations:
      configHash: 88915de4cf9c3551a8dc74c0137a3e83569d28c71044b0359c2578d2e0461825
```

然后，如果运行以下命令，则会选择新配置。

```
kubectl apply -f cwagent-daemonset.yaml
```

有关 yq 的更多信息，请参阅 [yq](#)。

删除 Container Insights 的 CloudWatch 代理和 FluentD

要删除与 CloudWatch 代理和 Fluentd 相关的所有资源，请输入以下命令。在此命令中，**Cluster** 是 Amazon EKS 或 Kubernetes 集群的名称，**Region** 是在其中发布日志的区域的名称。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/{{cluster_name}}/Cluster_Name;/s/{{region_name}}/Region/" | kubectl delete -f -
```

查看 Container Insights 指标

在设置了 Container Insights 并开始收集指标后，您可以在 CloudWatch 控制台中查看这些指标。

要在控制面板上显示 Container Insights 指标，您必须完成 Container Insights 设置。有关更多信息，请参阅 [设置 Container Insights \(p. 355\)](#)。

此过程说明如何查看 Container Insights 根据收集的日志数据自动生成的指标。本节的其余部分介绍了如何进一步深入了解您的数据并使用 CloudWatch Logs Insights 查看更多精细级别的指标。

查看 Container Insights 指标

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择性能监控。
3. 使用靠近顶部的下拉框选择要查看的资源类型以及特定资源。

您可以设置 Container Insights 收集的任何指标的 CloudWatch 告警。有关更多信息，请参阅 [使用 Amazon CloudWatch 告警 \(p. 151\)](#)

Note

如果您已设置 CloudWatch Application Insights 来监控容器化应用程序，Application Insights 控制面板将显示在 Container Insights 控制面板下方。如果尚未启用 Application Insights，可以通过在 Container Insights 控制面板中的性能视图下方选择 Auto-configure Application Insights (自动配置 Application Insights) 将其启用。

有关 Application Insights 和容器化应用程序的更多信息，请参阅 [为 Amazon ECS 和 Amazon EKS 资源监控启用 Application Insights \(p. 523\)](#)。

查看排名靠前的贡献者

对于 Container Insights 性能监控中的某些视图，您还可以通过内存或 CPU 或最近的活动资源查看排名靠前的贡献者。当您在靠近页面顶部的下拉框中选择以下任一控制面板时，此功能可用：

- ECS 服务
- ECS 任务
- EKS 命名空间
- EKS 服务
- EKS Pods

当您查看这些类型的任一资源时，页面底部会显示一个最初按 CPU 使用率排序的表格。您可以将其更改为按内存使用情况或最近的活动进行排序。要查看有关表格中某行的详细信息，可以选中该行旁边的复选框，然后选择 Actions (操作)，然后选择 Actions (操作) 菜单中的任一选项。

使用 CloudWatch Logs Insights 查看 Container Insights 数据

Container Insights 通过使用[嵌入式指标格式 \(p. 754\)](#)的性能日志事件来收集指标。日志存储在 CloudWatch Logs 中。CloudWatch 从日志中自动生成多个指标，您可以在 CloudWatch 控制台中查看这些指标。您还可以对使用 CloudWatch Logs Insights 查询收集到的性能数据进行更深入的分析。

有关 CloudWatch Logs Insights 的更多信息，请参阅[使用 CloudWatch Logs Insights 分析日志数据](#)。有关您可以在查询中使用的日志字段的更多信息，请参阅[Amazon EKS 和 Kubernetes 的 Container Insights 性能日志事件 \(p. 402\)](#)。

使用 CloudWatch Logs Insights 查询容器指标数据

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Insights。

屏幕顶部附近是查询编辑器。当您首次打开 CloudWatch Logs Insights 时，此框包含一个默认查询，该查询将返回 20 个最新的日志事件。

3. 在查询编辑器上方的框中，选择要查询的 Container Insights 日志组之一。要成功执行以下示例查询，日志组名称必须以 performance 结尾。

当您选择日志组时，CloudWatch Logs Insights 会自动检测日志组数据中的字段，并将其显示在右侧窗格中的 Discovered fields (发现的字段) 中。它还显示此日志组中的日志事件随时间变化的条形图。该条形图显示与您的查询和时间范围匹配的日志组中的事件分布情况，而不仅仅是表中显示的事件。

4. 在查询编辑器中，将默认查询替换为以下查询，然后选择运行查询。

```
STATS avg(node_cpu_utilization) as avg_node_cpu_utilization by NodeName
| SORT avg_node_cpu_utilization DESC
```

该查询显示一个节点列表，它按平均节点 CPU 使用率进行排序。

5. 要尝试使用另一个示例，请将该查询替换为另一个查询，然后选择运行查询。本页稍后将列出更多示例查询。

```
STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by PodName
| SORT avg_number_of_container_restarts DESC
```

该查询显示一个 pod 列表，它按平均容器重新启动次数进行排序。

6. 如果要尝试使用另一个查询，您可以使用屏幕右侧的列表中的包含字段。有关查询语法的更多信息，请参阅[CloudWatch Logs Insights 查询语法](#)。

查看资源列表

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Resources (资源)。

3. 默认视图是由 Container Insights 监控的资源列表，以及您在这些资源上设置的警报。要查看资源的可视映射，请选择地图视图。
4. 在地图视图中，您可以将指针悬停在地图中的任何资源上，查看有关该资源的基本指标。您可以选择任意资源来查看有关该资源的详细图形。

使用案例：查看 Amazon ECS 容器中的任务级指标

以下示例说明了如何使用 CloudWatch Logs Insights 深入了解 Container Insights 日志。有关更多示例，请参阅[博客推出适用于 Amazon ECS 的 Amazon CloudWatch Container Insights](#)。

Container Insights 不会在精细的任务级别自动生成指标。以下查询显示 CPU 和内存使用率的任务级指标。

```
stats avg(CpuUtilized) as CPU, avg(MemoryUtilized) as Mem by TaskId, ContainerName
| sort Mem, CPU desc
```

Container Insights 的其他示例查询

pod 列表，按平均容器重新启动次数排序

```
STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by PodName
| SORT avg_number_of_container_restarts DESC
```

请求的 Pod 与运行的 pod

```
fields @timestamp, @message
| sort @timestamp desc
| filter Type="Pod"
| stats min(pod_number_of_containers) as requested, min(pod_number_of_running_containers)
  as running, ceil(avg(pod_number_of_containers-pod_number_of_running_containers)) as
  pods_missing by kubernetes.pod_name
| sort pods_missing desc
```

集群节点故障计数

```
stats avg(cluster_failed_node_count) as CountOfNodeFailures
| filter Type="Cluster"
| sort @timestamp desc
```

按容器名称的应用程序日志错误

```
stats count() as countoferrors by kubernetes.container_name
| filter stream="stderr"
| sort countoferrors desc
```

按容器名称的磁盘使用量

```
stats floor(avg(container_filesystem_usage/1024)) as container_filesystem_usage_avg_kb by
  InstanceId, kubernetes.container_name, device
| filter Type="ContainerFS"
| sort container_filesystem_usage_avg_kb desc
```

按容器名称的 CPU 使用率

```
stats pct(container_cpu_usage_total, 50) as CPUPercMedian by kubernetes.container_name
```

| filter Type="Container"

Container Insights 收集的指标

Container Insights 收集一组针对 Amazon ECS 和 Amazon Fargate 的指标，以及另一组针对 Amazon EKS 和 Kubernetes 的指标。

主题

- [Amazon ECS Container Insights 指标 \(p. 393\)](#)
- [Amazon EKS 和 Kubernetes Container Insights 指标 \(p. 396\)](#)

Amazon ECS Container Insights 指标

下表列出了 Container Insights 收集的针对 Amazon ECS 的指标和维度。这些指标位于 ECS/ContainerInsights 命名空间中。有关更多信息，请参阅[指标 \(p. 3\)](#)。

如果您在控制台中未看到任何 Container Insights 指标，请确保已完成 Container Insights 的设置。在完全设置 Container Insights 之前，指标不会显示。有关更多信息，请参阅[设置 Container Insights \(p. 355\)](#)。

当您使用 Container Insights 收集以下指标时，这些指标将作为自定义指标收费。有关 CloudWatch 定价的更多信息，请参阅[Amazon CloudWatch 定价](#)。Amazon ECS 还会向 CloudWatch 自动发送几个免费指标。有关更多信息，请参阅[可用指标与维度](#)。

完成在[Amazon ECS 上针对集群级别和服务级别指标设置 Container Insights \(p. 356\)](#)中的步骤后，以下指标将可用

指标名称	Dimensions	描述
ContainerInstanceCount	ClusterName	注册到集群的运行 Amazon ECS 代理的 EC2 实例的数目。 单位：计数
CpuUtilized	TaskDefinitionFamily、ClusterName ServiceName、ClusterName ClusterName	任务在您使用的维度集所指定的资源中使用的 CPU 单元。 仅针对在任务定义中具有已定义的 CPU 保留的任务收集此指标。
CpuReserved	TaskDefinitionFamily、ClusterName ServiceName、ClusterName ClusterName	任务在您使用的维度集所指定的资源中预留的 CPU 单元。 仅针对在任务定义中具有已定义的 CPU 保留的任务收集此指标。
DeploymentCount	ServiceName、ClusterName	Amazon ECS 服务中的部署数。
DesiredTaskCount	ServiceName、ClusterName	Amazon ECS 服务所需的任务数。 单位：计数

指标名称	Dimensions	描述
MemoryUtilized	TaskDefinitionFamily、ClusterName ServiceName、ClusterName ClusterName	任务在您使用的维度集所指定的资源中使用的内存。 仅针对在任务定义中具有已定义的内存保留的任务收集此指标。
MemoryReserved	TaskDefinitionFamily、ClusterName ServiceName、ClusterName ClusterName	任务在您使用的维度集所指定的资源中预留的内存。 仅针对在任务定义中具有已定义的内存保留的任务收集此指标。
NetworkRxBytes	TaskDefinitionFamily、ClusterName ServiceName、ClusterName ClusterName	您使用的维度所指定的资源接收的字节数。 此指标仅适用于使用 awsvpc 或 bridge 网络模式的任务中的容器。 单位：字节/秒
NetworkTxBytes	TaskDefinitionFamily、ClusterName ServiceName、ClusterName ClusterName	您使用的维度所指定的资源传输的字节数。 此指标仅适用于使用 awsvpc 或 bridge 网络模式的任务中的容器。 单位：字节/秒
PendingTaskCount	ServiceName、ClusterName	当前处于 PENDING 状态的 task 的数量。 单位：计数
RunningTaskCount	ServiceName、ClusterName	当前处于 RUNNING 状态的 task 的数量。 单位：计数
ServiceCount	ClusterName	集群中的服务数。 单位：计数
StorageReadBytes	TaskDefinitionFamily、ClusterName ServiceName、ClusterName ClusterName	从您使用的维度所指定的资源中的存储读取的字节数。 单位：字节

指标名称	Dimensions	描述
StorageWriteBytes	TaskDefinitionFamily、ClusterName ServiceName、ClusterName ClusterName	写入到您使用的维度所指定的资源中的存储的字节数。 单位：字节
TaskCount	ClusterName	正在集群中运行的任务数。 单位：计数
TaskSetCount	ServiceName、ClusterName	服务中设置的任务数。 单位：计数

完成部署 CloudWatch 代理以收集 Amazon ECS 上的 EC2 实例级别指标 (p. 359) 中的步骤后，以下指标将可用

指标名称	Dimensions	描述
instance_cpu_limit	ClusterName	可分配给集群中的单个 EC2 实例的 CPU 单元的最大数目。
instance_cpu_reserved_capacity	ClusterName InstanceId、ContainerInstanceId、ClusterName	集群中单个 EC2 实例上当前预留的 CPU 的百分比。
instance_cpu_usage_total	ClusterName	集群中单个 EC2 实例上正在使用的 CPU 单元的数目。
instance_cpu_utilization	ClusterName InstanceId、ContainerInstanceId、ClusterName	集群中单个 EC2 实例上正在使用的 CPU 单元的总百分比。
instance_filesystem_utilization	ClusterName InstanceId、ContainerInstanceId、ClusterName	集群中单个 EC2 实例上正在使用的文件系统容量的总百分比。
instance_memory_limit	ClusterName	可分配给此集群中单个 EC2 实例的最大内存量 (以字节为单位)。
instance_memory_reserved_capacity	ClusterName InstanceId、ContainerInstanceId、ClusterName	集群中单个 EC2 实例上当前预留的内存的百分比。
instance_memory_utilization	ClusterName InstanceId、ContainerInstanceId、ClusterName	集群中单个 EC2 实例上正在使用的内存的总百分比。
instance_memory_working_set	ClusterName	集群中单个 EC2 实例上正在使用的内存量 (以字节为单位)。

指标名称	Dimensions	描述
instance_network_total_bytes	ClusterName	集群中单个 EC2 实例通过网络传输和接收的每秒总字节数。
instance_number_of_running_tasks	ClusterName	集群中单个 EC2 实例上正在运行的任务的数目。

Amazon EKS 和 Kubernetes Container Insights 指标

下表列出了 Container Insights 收集的针对 Amazon EKS 和 Kubernetes 的指标和维度。这些指标位于 ContainerInsights 命名空间中。有关更多信息，请参阅[指标 \(p. 3\)](#)。

如果您在控制台中未看到任何 Container Insights 指标，请确保已完成 Container Insights 的设置。在完全设置 Container Insights 之前，指标不会显示。有关更多信息，请参阅[设置 Container Insights \(p. 355\)](#)。

当您使用 Container Insights 收集以下指标时，这些指标将作为自定义指标收费。有关 CloudWatch 定价的更多信息，请参阅[Amazon CloudWatch 定价](#)。

指标名称	Dimensions	描述
cluster_failed_node_count	ClusterName	集群中失败的工作线程节点的数目。如果节点遭受任何节点条件的影响，则该节点被视为失败。有关更多信息，请参阅 Kubernetes 文档中的 条件 。
cluster_node_count	ClusterName	集群中工作线程节点的总数。
namespace_number_of_running_pods	Namespace, ClusterName	您使用的维度所指定的资源中每个命名空间运行的 pod 的数目。
node_cpu_limit	ClusterName	可以分配给此集群中单个节点的 CPU 单元的最大数目。
node_cpu_reserved_capacity	NodeName, ClusterName, InstanceId ClusterName	为节点组件保留的 CPU 单元的百分比，例如 kubelet、kube-proxy 和 Docker。
node_cpu_usage_total	ClusterName	集群中节点上正在使用的 CPU 单元的数目。
node_cpu_utilization	NodeName, ClusterName, InstanceId ClusterName	集群中节点上正在使用的 CPU 单元的总百分比。
node_filesystem_utilization	NodeName, ClusterName, InstanceId ClusterName	集群中节点上正在使用的文件系统容量的总百分比。

指标名称	Dimensions	描述
node_memory_limit	ClusterName	可以分配给此集群中单个节点的最大内存量 (以字节为单位)。
node_memory_reserved	NodeName, ClusterName, InstanceId ClusterName	集群中节点上当前正在使用的内存百分比。
node_memory_utilization	NodeName, ClusterName, InstanceId ClusterName	一个或多个节点当前正在使用的内存百分比。 node_memory_utilization 的计算方法为 node_memory_working_set/ node_memory_limit。 它是节点内存使用量相比节点内存限制的百分比。
node_memory_working_set	ClusterName	集群中节点的工作集中正在使用的内存量 (以字节为单位)。
node_network_total_bytes	NodeName, ClusterName, InstanceId ClusterName	集群中每个节点通过网络传输和接收的每秒总字节数。
node_number_of_running_containers	NodeName, ClusterName, InstanceId ClusterName	集群中每个节点的正在运行的容器数。
node_number_of_running_pods	NodeName, ClusterName, InstanceId ClusterName	集群中每个节点上运行的 pod 的数量。
pod_cpu_reserved_capacity	PodName, Namespace, ClusterName ClusterName	集群中每个 pod 预留的 CPU 容量。
pod_cpu_utilization	PodName, Namespace, ClusterName Namespace, ClusterName Service, Namespace, ClusterName ClusterName	pod 所使用的 CPU 单元的百分比。
pod_cpu_utilization_exceeded	PodName, Namespace, ClusterName Namespace, ClusterName Service, Namespace, ClusterName ClusterName	超出 pod 限制的 pod 所使用的 CPU 单元的百分比。
pod_memory_reserved	PodName, Namespace, ClusterName ClusterName	为 pod 预留的内存的百分比。

指标名称	Dimensions	描述
pod_memory_utilization	PodName、Namespace、ClusterName Namespace、ClusterName Service、Namespace、ClusterName ClusterName	一个或多个 pod 当前正在使用的内存百分比。
pod_memory_utilization_exceeded	PodName、Namespace、ClusterName Namespace、ClusterName Service、Namespace、ClusterName ClusterName	超出 pod 限制的 pod 所使用的内存百分比。
pod_number_of_container_restarts	PodName、Namespace、ClusterName	一个 pod 中容器重新启动的总次数。
pod_network_rx_bytes	PodName、Namespace、ClusterName Namespace、ClusterName Service、Namespace、ClusterName ClusterName	pod 通过网络每秒接收的字节数。
pod_network_tx_bytes	PodName、Namespace、ClusterName Namespace、ClusterName Service、Namespace、ClusterName ClusterName	pod 通过网络每秒传输的字节数。
service_number_of_running_pods	Service、Namespace、ClusterName ClusterName	运行集群中的服务的 pod 的数量。

Container Insights 性能日志参考

本节包含有关 Container Insights 如何使用性能日志事件来收集指标的参考信息。

主题

- [Amazon ECS 的 Container Insights 性能日志事件 \(p. 398\)](#)
- [Amazon EKS 和 Kubernetes 的 Container Insights 性能日志事件 \(p. 402\)](#)
- [Amazon EKS 和 Kubernetes 的性能日志事件中的相关字段 \(p. 413\)](#)

Amazon ECS 的 Container Insights 性能日志事件

以下是 Container Insights 从 Amazon ECS 收集的性能日志事件的实例。

类型：容器

```
{
```

```
"Version": "0",
"Type": "Container",
"ContainerName": "sleep",
"TaskId": "7ac4dfba69214411b4783a3b8189c9ba",
"TaskDefinitionFamily": "sleep360",
"TaskDefinitionRevision": "1",
"ContainerInstanceId": "0d7650e6dec34c1a9200f72098071e8f",
"EC2InstanceId": "i-0c470579dbcd2f3",
"ClusterName": "MyCluster",
"Image": "busybox",
"ContainerKnownStatus": "RUNNING",
"Timestamp": 1623963900000,
"CpuUtilized": 0.0,
"CpuReserved": 10.0,
"MemoryUtilized": 0,
"MemoryReserved": 10,
"StorageReadBytes": 0,
"StorageWriteBytes": 0,
"NetworkRxBytes": 0,
"NetworkRxDropped": 0,
"NetworkRxErrors": 0,
"NetworkRxPackets": 14,
"NetworkTxBytes": 0,
"NetworkTxDropped": 0,
"NetworkTxErrors": 0,
"NetworkTxPackets": 0
}
```

类型：任务

```
{
  "Version": "0",
  "Type": "Task",
  "TaskId": "7ac4dfba69214411b4783a3b8189c9ba",
  "TaskDefinitionFamily": "sleep360",
  "TaskDefinitionRevision": "1",
  "ContainerInstanceId": "0d7650e6dec34c1a9200f72098071e8f",
  "EC2InstanceId": "i-0c470579dbcd2f3",
  "ClusterName": "MyCluster",
  "AccountID": "637146863587",
  "Region": "us-west-2",
  "AvailabilityZone": "us-west-2b",
  "KnownStatus": "RUNNING",
  "LaunchType": "EC2",
  "PullStartedAt": 1623963608201,
  "PullStoppedAt": 1623963610065,
  "CreatedAt": 1623963607094,
  "StartedAt": 1623963610382,
  "Timestamp": 1623963900000,
  "CpuUtilized": 0.0,
  "CpuReserved": 10.0,
  "MemoryUtilized": 0,
  "MemoryReserved": 10,
  "StorageReadBytes": 0,
  "StorageWriteBytes": 0,
  "NetworkRxBytes": 0,
  "NetworkRxDropped": 0,
  "NetworkRxErrors": 0,
  "NetworkRxPackets": 14,
  "NetworkTxBytes": 0,
  "NetworkTxDropped": 0,
  "NetworkTxErrors": 0,
  "NetworkTxPackets": 0,
  "CloudWatchMetrics": [{
    "Namespace": "ECS/ContainerInsights",
```

```
"Metrics": [{
  "Name": "CpuUtilized",
  "Unit": "None"
},
{
  "Name": "CpuReserved",
  "Unit": "None"
},
{
  "Name": "MemoryUtilized",
  "Unit": "Megabytes"
},
{
  "Name": "MemoryReserved",
  "Unit": "Megabytes"
},
{
  "Name": "StorageReadBytes",
  "Unit": "Bytes/Second"
},
{
  "Name": "StorageWriteBytes",
  "Unit": "Bytes/Second"
},
{
  "Name": "NetworkRxBytes",
  "Unit": "Bytes/Second"
},
{
  "Name": "NetworkTxBytes",
  "Unit": "Bytes/Second"
}
],
"Dimensions": [
  [
    "ClusterName"
  ],
  [
    "ClusterName",
    "TaskDefinitionFamily"
  ]
]
}]
}
```

类型：服务

```
{
  "Version": "0",
  "Type": "Service",
  "ServiceName": "myCIService",
  "ClusterName": "myCICluster",
  "Timestamp": 1561586460000,
  "DesiredTaskCount": 2,
  "RunningTaskCount": 2,
  "PendingTaskCount": 0,
  "DeploymentCount": 1,
  "TaskSetCount": 0,
  "CloudWatchMetrics": [
    {
      "Namespace": "ECS/ContainerInsights",
      "Metrics": [
        {
          "Name": "DesiredTaskCount",
          "Unit": "Count"
        }
      ]
    }
  ]
}
```



```
        },
        {
            "Name": "RunningTaskCount",
            "Unit": "Count"
        },
        {
            "Name": "PendingTaskCount",
            "Unit": "Count"
        },
        {
            "Name": "DeploymentCount",
            "Unit": "Count"
        },
        {
            "Name": "TaskSetCount",
            "Unit": "Count"
        }
    ],
    "Dimensions": [
        [
            "ServiceName",
            "ClusterName"
        ]
    ]
}
]
```

类型：集群

```
{
    "Version": "0",
    "Type": "Cluster",
    "ClusterName": "myCICluster",
    "Timestamp": 1561587300000,
    "TaskCount": 5,
    "ContainerInstanceCount": 5,
    "ServiceCount": 2,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "TaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "ContainerInstanceCount",
                    "Unit": "Count"
                },
                {
                    "Name": "ServiceCount",
                    "Unit": "Count"
                }
            ]
        },
        "Dimensions": [
            [
                "ClusterName"
            ]
        ]
    ]
}
```

Amazon EKS 和 Kubernetes 的 Container Insights 性能日志事件

以下是 Container Insights 从 Amazon EKS 和 Kubernetes 集群中收集的性能日志事件的示例。

类型 : Node

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "node_network_total_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "node_cpu_reserved_capacity"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_reserved_capacity"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_pods"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_containers"
        }
      ],
      "Dimensions": [
        "NodeName",
        "InstanceId",
        "ClusterName"
      ],
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "node_network_total_bytes"
        }
      ],
    }
  ]
}
```

```

    {
      "Unit": "Percent",
      "Name": "node_cpu_reserved_capacity"
    },
    {
      "Unit": "Percent",
      "Name": "node_memory_reserved_capacity"
    },
    {
      "Unit": "Count",
      "Name": "node_number_of_running_pods"
    },
    {
      "Unit": "Count",
      "Name": "node_number_of_running_containers"
    },
    {
      "Name": "node_cpu_usage_total"
    },
    {
      "Name": "node_cpu_limit"
    },
    {
      "Unit": "Bytes",
      "Name": "node_memory_working_set"
    },
    {
      "Unit": "Bytes",
      "Name": "node_memory_limit"
    }
  ],
  "Dimensions": [
    [
      "ClusterName"
    ]
  ],
  "Namespace": "ContainerInsights"
}
],
"ClusterName": "myCICluster",
"InstanceId": "i-1234567890123456",
"InstanceType": "t3.xlarge",
"NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
"Sources": [
  "cadvisor",
  "/proc",
  "pod",
  "calculated"
],
"Timestamp": "1567096682364",
"Type": "Node",
"Version": "0",
"kubernetes": {
  "host": "ip-192-168-75-26.us-west-2.compute.internal"
},
"node_cpu_limit": 4000,
"node_cpu_request": 1130,
"node_cpu_reserved_capacity": 28.249999999999996,
"node_cpu_usage_system": 33.794636630852764,
"node_cpu_usage_total": 136.47852169244098,
"node_cpu_usage_user": 71.67075111567326,
"node_cpu_utilization": 3.4119630423110245,
"node_memory_cache": 3103297536,
"node_memory_failcnt": 0,
"node_memory_hierarchical_pgfault": 0,
"node_memory_hierarchical_pgmajfault": 0,

```

```
"node_memory_limit": 16624865280,
"node_memory_mapped_file": 406646784,
"node_memory_max_usage": 4230746112,
"node_memory_pgfault": 0,
"node_memory_pgmajfault": 0,
"node_memory_request": 1115684864,
"node_memory_reserved_capacity": 6.7109407818311055,
"node_memory_rss": 798146560,
"node_memory_swap": 0,
"node_memory_usage": 3901444096,
"node_memory_utilization": 6.601302600149552,
"node_memory_working_set": 1097457664,
"node_network_rx_bytes": 35918.392817386324,
"node_network_rx_dropped": 0,
"node_network_rx_errors": 0,
"node_network_rx_packets": 157.67565245448117,
"node_network_total_bytes": 68264.20276554905,
"node_network_tx_bytes": 32345.80994816272,
"node_network_tx_dropped": 0,
"node_network_tx_errors": 0,
"node_network_tx_packets": 154.21455923431654,
"node_number_of_running_containers": 16,
"node_number_of_running_pods": 13
}
```

类型 : NodeFS

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_filesystem_utilization"
        }
      ],
      "Dimensions": [
        [
          "NodeName",
          "InstanceId",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097939726",
  "Type": "NodeFS",
  "Version": "0",
  "device": "/dev/nvme0n1p1",
  "fstype": "vfs",
}
```

```
"kubernetes": {
  "host": "ip-192-168-75-26.us-west-2.compute.internal"
},
"node_filesystem_available": 17298395136,
"node_filesystem_capacity": 21462233088,
"node_filesystem_inodes": 10484720,
"node_filesystem_inodes_free": 10367158,
"node_filesystem_usage": 4163837952,
"node_filesystem_utilization": 19.400767547940255
}
```

类型 : NodeDiskIO

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor"
  ],
  "Timestamp": "1567096928131",
  "Type": "NodeDiskIO",
  "Version": "0",
  "device": "/dev/nvme0n1",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_diskio_io_service_bytes_async": 9750.505814277016,
  "node_diskio_io_service_bytes_read": 0,
  "node_diskio_io_service_bytes_sync": 230.6174506688036,
  "node_diskio_io_service_bytes_total": 9981.123264945818,
  "node_diskio_io_service_bytes_write": 9981.123264945818,
  "node_diskio_io_serviced_async": 1.153087253344018,
  "node_diskio_io_serviced_read": 0,
  "node_diskio_io_serviced_sync": 0.03603397666700056,
  "node_diskio_io_serviced_total": 1.1891212300110185,
  "node_diskio_io_serviced_write": 1.1891212300110185
}
```

类型 : NodeNet

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567096928131",
  "Type": "NodeNet",
  "Version": "0",
  "interface": "eni972f6bfa9a0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
}
```

```

"node_interface_network_rx_bytes": 3163.008420864309,
"node_interface_network_rx_dropped": 0,
"node_interface_network_rx_errors": 0,
"node_interface_network_rx_packets": 16.575629266820258,
"node_interface_network_total_bytes": 3518.3935157426017,
"node_interface_network_tx_bytes": 355.385094878293,
"node_interface_network_tx_dropped": 0,
"node_interface_network_tx_errors": 0,
"node_interface_network_tx_packets": 3.9997714100370625
}

```

类型 : Pod

```

{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "pod_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "pod_network_rx_bytes"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "pod_network_tx_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "pod_cpu_utilization_over_pod_limit"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_utilization_over_pod_limit"
        }
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ],
        [
          "Service",
          "Namespace",
          "ClusterName"
        ],
        [
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    },
  ],
}

```

```
{
  "Metrics": [
    {
      "Unit": "Percent",
      "Name": "pod_cpu_reserved_capacity"
    },
    {
      "Unit": "Percent",
      "Name": "pod_memory_reserved_capacity"
    }
  ],
  "Dimensions": [
    [
      "PodName",
      "Namespace",
      "ClusterName"
    ],
    [
      "ClusterName"
    ]
  ],
  "Namespace": "ContainerInsights"
},
{
  "Metrics": [
    {
      "Unit": "Count",
      "Name": "pod_number_of_container_restarts"
    }
  ],
  "Dimensions": [
    [
      "PodName",
      "Namespace",
      "ClusterName"
    ]
  ],
  "Namespace": "ContainerInsights"
}
],
"ClusterName": "myCICluster",
"InstanceId": "i-1234567890123456",
"InstanceType": "t3.xlarge",
"Namespace": "amazon-cloudwatch",
"NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
"PodName": "cloudwatch-agent-statsd",
"Service": "cloudwatch-agent-statsd",
"Sources": [
  "cadvisor",
  "pod",
  "calculated"
],
"Timestamp": "1567097351092",
"Type": "Pod",
"Version": "0",
"kubernetes": {
  "host": "ip-192-168-75-26.us-west-2.compute.internal",
  "labels": {
    "app": "cloudwatch-agent-statsd",
    "pod-template-hash": "df44f855f"
  },
  "namespace_name": "amazon-cloudwatch",
  "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
  "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
  "pod_owners": [
    {
```



```

        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
    }
],
    "service_name": "cloudwatch-agent-statsd"
},
    "pod_cpu_limit": 200,
    "pod_cpu_request": 200,
    "pod_cpu_reserved_capacity": 5,
    "pod_cpu_usage_system": 1.4504841104992765,
    "pod_cpu_usage_total": 5.817016867430125,
    "pod_cpu_usage_user": 1.1281543081661038,
    "pod_cpu_utilization": 0.14542542168575312,
    "pod_cpu_utilization_over_pod_limit": 2.9085084337150624,
    "pod_memory_cache": 8192,
    "pod_memory_failcnt": 0,
    "pod_memory_hierarchical_pgfault": 0,
    "pod_memory_hierarchical_pgmajfault": 0,
    "pod_memory_limit": 104857600,
    "pod_memory_mapped_file": 0,
    "pod_memory_max_usage": 25268224,
    "pod_memory_pgfault": 0,
    "pod_memory_pgmajfault": 0,
    "pod_memory_request": 104857600,
    "pod_memory_reserved_capacity": 0.6307275170893897,
    "pod_memory_rss": 22777856,
    "pod_memory_swap": 0,
    "pod_memory_usage": 25141248,
    "pod_memory_utilization": 0.10988455961791709,
    "pod_memory_utilization_over_pod_limit": 17.421875,
    "pod_memory_working_set": 18268160,
    "pod_network_rx_bytes": 9880.697124714186,
    "pod_network_rx_dropped": 0,
    "pod_network_rx_errors": 0,
    "pod_network_rx_packets": 107.80005532263283,
    "pod_network_total_bytes": 10158.829201483635,
    "pod_network_tx_bytes": 278.13207676944796,
    "pod_network_tx_dropped": 0,
    "pod_network_tx_errors": 0,
    "pod_network_tx_packets": 1.146027574644318,
    "pod_number_of_container_restarts": 0,
    "pod_number_of_containers": 1,
    "pod_number_of_running_containers": 1,
    "pod_status": "Running"
}

```

类型 : PodNet

```

{
    "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
    "ClusterName": "myCICluster",
    "InstanceId": "i-1234567890123456",
    "InstanceType": "t3.xlarge",
    "Namespace": "amazon-cloudwatch",
    "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
    "PodName": "cloudwatch-agent-statsd",
    "Service": "cloudwatch-agent-statsd",
    "Sources": [
        "cadvisor",
        "calculated"
    ],
    "Timestamp": "1567097351092",
    "Type": "PodNet",
    "Version": "0",
}

```

```

"interface": "eth0",
"kubernetes": {
  "host": "ip-192-168-75-26.us-west-2.compute.internal",
  "labels": {
    "app": "cloudwatch-agent-statsd",
    "pod-template-hash": "df44f855f"
  },
  "namespace_name": "amazon-cloudwatch",
  "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
  "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
  "pod_owners": [
    {
      "owner_kind": "Deployment",
      "owner_name": "cloudwatch-agent-statsd"
    }
  ],
  "service_name": "cloudwatch-agent-statsd"
},
"pod_interface_network_rx_bytes": 9880.697124714186,
"pod_interface_network_rx_dropped": 0,
"pod_interface_network_rx_errors": 0,
"pod_interface_network_rx_packets": 107.80005532263283,
"pod_interface_network_total_bytes": 10158.829201483635,
"pod_interface_network_tx_bytes": 278.13207676944796,
"pod_interface_network_tx_dropped": 0,
"pod_interface_network_tx_errors": 0,
"pod_interface_network_tx_packets": 1.146027574644318
}

```

类型：容器

```

{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-NodeGroup-sample",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567097399912",
  "Type": "Container",
  "Version": "0",
  "container_cpu_limit": 200,
  "container_cpu_request": 200,
  "container_cpu_usage_system": 1.87958283771964,
  "container_cpu_usage_total": 6.159993652997942,
  "container_cpu_usage_user": 1.6707403001952357,
  "container_cpu_utilization": 0.15399984132494854,
  "container_memory_cache": 8192,
  "container_memory_failcnt": 0,
  "container_memory_hierarchical_pgfault": 0,
  "container_memory_hierarchical_pgmajfault": 0,
  "container_memory_limit": 104857600,
  "container_memory_mapped_file": 0,
  "container_memory_max_usage": 24580096,
  "container_memory_pgfault": 0,
  "container_memory_pgmajfault": 0,
  "container_memory_request": 104857600,
  "container_memory_rss": 22736896,

```

```

"container_memory_swap": 0,
"container_memory_usage": 24453120,
"container_memory_utilization": 0.10574541028701798,
"container_memory_working_set": 17580032,
"container_status": "Running",
"kubernetes": {
  "container_name": "cloudwatch-agent",
  "docker": {
    "container_id": "8967b6b37da239dfad197c9fdea3e5dfd35a8a759ec86e2e4c3f7b401e232706"
  },
  "host": "ip-192-168-75-26.us-west-2.compute.internal",
  "labels": {
    "app": "cloudwatch-agent-statsd",
    "pod-template-hash": "df44f855f"
  },
  "namespace_name": "amazon-cloudwatch",
  "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
  "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
  "pod_owners": [
    {
      "owner_kind": "Deployment",
      "owner_name": "cloudwatch-agent-statsd"
    }
  ],
  "service_name": "cloudwatch-agent-statsd"
},
"number_of_container_restarts": 0
}

```

类型：ContainerFS

```

{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097399912",
  "Type": "ContainerFS",
  "Version": "0",
  "container_filesystem_available": 0,
  "container_filesystem_capacity": 21462233088,
  "container_filesystem_usage": 24576,
  "container_filesystem_utilization": 0.0001145081217748071,
  "device": "/dev/nvme0n1p1",
  "fstype": "vfs",
  "kubernetes": {
    "container_name": "cloudwatch-agent",
    "docker": {
      "container_id": "8967b6b37da239dfad197c9fdea3e5dfd35a8a759ec86e2e4c3f7b401e232706"
    },
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    }
  },
}

```

```
"namespace_name": "amazon-cloudwatch",
"pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
"pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
"pod_owners": [
  {
    "owner_kind": "Deployment",
    "owner_name": "cloudwatch-agent-statsd"
  }
],
"service_name": "cloudwatch-agent-statsd"
}
}
```

类型：集群

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "cluster_node_count"
        },
        {
          "Unit": "Count",
          "Name": "cluster_failed_node_count"
        }
      ],
      "Dimensions": [
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097534160",
  "Type": "Cluster",
  "Version": "0",
  "cluster_failed_node_count": 0,
  "cluster_node_count": 3
}
```

类型：“ClusterService”

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "service_number_of_running_pods"
        }
      ],
      "Dimensions": [
        [
          "Service",
          "Namespace",
          "ClusterName"
        ]
      ]
    }
  ]
}
```

```

        ],
        [
            "ClusterName"
        ]
    ],
    "Namespace": "ContainerInsights"
}
],
"ClusterName": "myCICluster",
"Namespace": "amazon-cloudwatch",
"Service": "cloudwatch-agent-statsd",
"Sources": [
    "apiserver"
],
"Timestamp": "1567097534160",
"Type": "ClusterService",
"Version": "0",
"kubernetes": {
    "namespace_name": "amazon-cloudwatch",
    "service_name": "cloudwatch-agent-statsd"
},
"service_number_of_running_pods": 1
}

```

类型 : ClusterNamespace

```

{
    "CloudWatchMetrics": [
        {
            "Metrics": [
                {
                    "Unit": "Count",
                    "Name": "namespace_number_of_running_pods"
                }
            ],
            "Dimensions": [
                [
                    "Namespace",
                    "ClusterName"
                ],
                [
                    "ClusterName"
                ]
            ],
            "Namespace": "ContainerInsights"
        }
    ],
    "ClusterName": "myCICluster",
    "Namespace": "amazon-cloudwatch",
    "Sources": [
        "apiserver"
    ],
    "Timestamp": "1567097594160",
    "Type": "ClusterNamespace",
    "Version": "0",
    "kubernetes": {
        "namespace_name": "amazon-cloudwatch"
    },
    "namespace_number_of_running_pods": 7
}

```

Amazon EKS 和 Kubernetes 的性能日志事件中的相关字段

对于 Amazon EKS 和 Kubernetes，容器化的 CloudWatch 代理将数据作为性能日志事件发出。这使 CloudWatch 能够摄取和存储高基数数据。CloudWatch 使用性能日志事件中的数据在集群、节点和 pod 级别创建聚合的 CloudWatch 指标，而不会丢失详细信息。

下表列出了这些性能日志事件中与 Container Insights 指标数据收集相关的字段。您可以使用 CloudWatch Logs Insights 查询其中的任何字段以收集数据或调查问题。有关更多信息，请参阅[使用 CloudWatch Logs Insights 分析日志数据](#)。

类型	日志字段	源	公式或备注
Pod	pod_cpu_utilization	计算	公式： $\text{pod\_cpu\_usage\_total} / \text{node\_cpu\_limit}$
Pod	pod_cpu_usage_total pod_cpu_usage_total 以千分之一核心为单位报道。	cadvisor	
Pod	pod_cpu_limit	计算	公式： $\text{sum}(\text{container\_cpu\_limit})$ 如果 pod 中的任何容器未定义 CPU 限制，则该字段不会显示在日志事件中。
Pod	pod_cpu_request	计算	公式： $\text{sum}(\text{container\_cpu\_request})$ 不保证设置 container_cpu_request。仅在总和中包含设置的字段。
Pod	pod_cpu_utilization_over_pod_limit	计算	公式： $\text{pod\_cpu\_usage\_total} / \text{pod\_cpu\_limit}$
Pod	pod_cpu_reserved_capacity	计算	公式： $\text{pod\_cpu\_request} / \text{node\_cpu\_limit}$
Pod	pod_memory_utilization	计算	公式： $\text{pod\_memory\_working\_set} / \text{node\_memory\_limit}$ 它是 pod 内存使用量相比节点内存限制的百分比。
Pod	pod_memory_working_set	cadvisor	
Pod	pod_memory_limit	计算	公式： $\text{sum}(\text{container\_memory\_limit})$

类型	日志字段	源	公式或备注
			如果 pod 中的任何容器没有定义内存限制，则该字段不会显示在日志事件中。
Pod	pod_memory_request	计算	公式： $\text{sum}(\text{container\_memory\_request})$ 不保证设置 container_memory_request 仅在总和中包括设置的字段。
Pod	pod_memory_utilization_over_pod_limit	计算	公式： $\frac{\text{pod\_memory\_working\_set}}{\text{pod\_memory\_limit}}$ 如果 pod 中的任何容器没有定义内存限制，则该字段不会显示在日志事件中。
Pod	pod_memory_reserved_capacity	计算	公式： $\frac{\text{pod\_memory\_request}}{\text{node\_memory\_limit}}$
Pod	pod_network_tx_bytes	计算	公式： $\text{sum}(\text{pod\_interface\_network\_tx\_bytes})$ 该数据适用于每个 pod 的所有网络接口。CloudWatch 代理计算总数并添加指标提取规则。
Pod	pod_network_rx_bytes	计算	公式： $\text{sum}(\text{pod\_interface\_network\_rx\_bytes})$
Pod	pod_network_total_bytes	计算	公式： $\text{pod\_network\_rx\_bytes} + \text{pod\_network\_tx\_bytes}$
PodNet	pod_interface_network_rx_bytes	cadvisor	该数据是 pod 网络接口每秒接收的网络字节数。
PodNet	pod_interface_network_tx_bytes	cadvisor	该数据是 pod 网络接口每秒发送的网络字节数。
容器	container_cpu_usage_total	cadvisor	

类型	日志字段	源	公式或备注
容器	container_cpu_limit	cadvisor	不保证设置该字段。如果未设置，则不会发出该指标。
容器	container_cpu_request	cadvisor	不保证设置该字段。如果未设置，则不会发出该指标。
容器	container_memory_working_set	cadvisor	
容器	container_memory_limit	创建集群时必须进行一些设置，例如允许通过 SSH 访问 Amazon EC2 节点。	不保证设置该字段。如果未设置，则不会发出该指标。
容器	container_memory_request	创建集群时必须进行一些设置，例如允许通过 SSH 访问 Amazon EC2 节点。	不保证设置该字段。如果未设置，则不会发出该指标。
ContainerFS	container_filesystem_capacity	创建集群时必须进行一些设置，例如允许通过 SSH 访问 Amazon EC2 节点。	该数据适用于每个磁盘设备。
ContainerFS	container_filesystem_usage	创建集群时必须进行一些设置，例如允许通过 SSH 访问 Amazon EC2 节点。	该数据适用于每个磁盘设备。
ContainerFS	container_filesystem_utilization	计算	公式： $\frac{\text{container\_filesystem\_usage}}{\text{container\_filesystem\_capacity}}$ 该数据适用于每个设备名称。
节点	node_cpu_utilization	计算	公式： $\frac{\text{node\_cpu\_usage\_total}}{\text{node\_cpu\_limit}}$
节点	node_cpu_usage_total	cadvisor	
节点	node_cpu_limit	/proc	
节点	node_cpu_request	计算	公式： $\text{sum}(\text{pod\_cpu\_request})$

类型	日志字段	源	公式或备注
节点	node_cpu_reserved_capacity	计算	公式: $\text{node\_cpu\_request} / \text{node\_cpu\_limit}$
节点	node_memory_utilization	计算	公式: $\text{node\_memory\_working\_set} / \text{node\_memory\_limit}$
节点	node_memory_working_set	cadvisor	
节点	node_memory_limit	/proc	
节点	node_memory_request	计算	公式: $\text{sum}(\text{pod\_memory\_request})$
节点	node_memory_reserved_capacity	计算	公式: $\text{node\_memory\_request} / \text{node\_memory\_limit}$
节点	node_network_rx_bytes	计算	公式: $\text{sum}(\text{node\_interface\_rx\_bytes})$
节点	node_network_tx_bytes	计算	公式: $\text{sum}(\text{node\_interface\_tx\_bytes})$
节点	node_network_total_bytes	计算	公式: $\text{node\_network\_rx\_bytes} + \text{node\_network\_tx\_bytes}$
节点	node_number_of_running_pods	Pod 列表	
节点	node_number_of_running_containers	Pod 列表	
NodeNet	node_interface_network_rx_bytes	cadvisor	该数据是工作线程节点网络接口每秒接收的网络字节数。
NodeNet	node_interface_network_tx_bytes	cadvisor	该数据是工作线程节点网络接口每秒发送的网络字节数。
NodeFS	node_filesystem_capacity	cadvisor	
NodeFS	node_filesystem_usage	cadvisor	
NodeFS	node_filesystem_utilization	计算	公式: $\text{node\_filesystem\_usage} / \text{node\_filesystem\_capacity}$ 该数据适用于每个设备名称。
Cluster	cluster_failed_node_count	API 服务器	
Cluster	cluster_node_count	API 服务器	

类型	日志字段	源	公式或备注
服务	service_number_of_running_pods	API 服务器	
命名空间	namespace_number_of_running_pods	API 服务器	

指标计算示例

本节包含的示例说明了如何计算上表中的某些值。

假设您具有一个处于以下状态的集群。

```
Node1
  node_cpu_limit = 4
  node_cpu_usage_total = 3

  Pod1
    pod_cpu_usage_total = 2

    Container1
      container_cpu_limit = 1
      container_cpu_request = 1
      container_cpu_usage_total = 0.8

    Container2
      container_cpu_limit = null
      container_cpu_request = null
      container_cpu_usage_total = 1.2

  Pod2
    pod_cpu_usage_total = 0.4

    Container3
      container_cpu_limit = 1
      container_cpu_request = 0.5
      container_cpu_usage_total = 0.4

Node2
  node_cpu_limit = 8
  node_cpu_usage_total = 1.5

  Pod3
    pod_cpu_usage_total = 1

    Container4
      container_cpu_limit = 2
      container_cpu_request = 2
      container_cpu_usage_total = 1
```

下表说明了如何使用该数据计算 pod CPU 指标。

指标	公式	Pod1	Pod2	Pod3
pod_cpu_utilization	pod_cpu_usage_total / node_cpu_limit	2 / 4 = 50%	0.4 / 4 = 10%	1 / 8 = 12.5%
pod_cpu_utilization_over_pod_limit	pod_cpu_usage_total / sum(container_cpu_limit)	不适用，因为没有定义 Container2	0.4 / 1 = 40%	1 / 2 = 50%

指标	公式	Pod1	Pod2	Pod3
		的 CPU 限制		
pod_cpu_reserved_capacity	$\frac{\sum(\text{container\_cpu\_request})}{\text{node\_cpu\_limit}}$	$(1 + 0) / 4 = 25\%$	$0.5 / 4 = 12.5\%$	$2 / 8 = 25\%$

下表说明了如何使用该数据计算节点 CPU 指标。

指标	公式	Node1	Node2
node_cpu_utilization	$\frac{\text{node\_cpu\_usage\_total}}{\text{node\_cpu\_limit}}$	$3 / 4 = 75\%$	$1.5 / 8 = 18.75\%$
node_cpu_reserved_capacity	$\frac{\sum(\text{pod\_cpu\_request})}{\text{node\_cpu\_limit}}$	$1.5 / 4 = 37.5\%$	$2 / 8 = 25\%$

Container Insights Prometheus 指标监控

适用于 Prometheus 的 CloudWatch Container Insights 监控可以自动发现来自容器化系统和工作负载的 Prometheus 指标。Prometheus 是一个开源系统监控和警报工具包。有关更多信息，请参阅 Prometheus 文档中的[什么是 Prometheus？](#)

[Amazon Elastic Container Service](#)、[Amazon Elastic Kubernetes Service](#) 和在 Amazon EC2 实例上运行的 [Kubernetes](#) 集群支持发现 Prometheus 指标。收集了 Prometheus 计数器、计量表和汇总指标类型。对直方图指标的支持计划在未来的版本中推出。

对于 Amazon ECS 和 Amazon EKS 集群，EC2 和 Fargate 启动类型均受支持。Container Insights 会自动从多个工作负载收集指标，您可以将其配置为从任何工作负载收集指标。

您可以采用 Prometheus 作为开源和开放标准方法，在 CloudWatch 中摄取自定义指标。具有 Prometheus 支持的 CloudWatch 代理可以发现并收集 Prometheus 指标，以便监控应用程序性能下降和故障，并更快地进行故障排除和发出告警。这也减少了改进可观测性所需的监控工具数量。

Container Insights Prometheus 支持涉及按用量付费的指标和日志，包括收集、存储和分析。有关更多信息，请参阅 [Amazon CloudWatch 定价](#)。

某些工作负载的预构建控制面板

Container Insights Prometheus 解决方案包括针对本节中列出的常见工作负载的预构建控制面板。有关这些工作负载的示例配置，请参阅 [（可选）为 Prometheus 指标测试设置示例容器化 Amazon ECS 工作负载 \(p. 431\)](#) 和 [（可选）为 Prometheus 指标测试设置示例容器化 Amazon EKS 工作负载 \(p. 466\)](#)。

您还可以配置 Container Insights 以通过编辑代理配置文件从其他容器化服务和应用程序收集 Prometheus 指标。

带有适用于在 Amazon EC2 实例上运行的 Amazon EKS 集群和 Kubernetes 集群的预构建控制面板的工作负载：

- Amazon App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy

带有适用于 Amazon ECS 集群的预构建控制面板的工作负载：

- Amazon App Mesh
- Java/JMX
- NGINX
- NGINX Plus

在亚马逊云服务器集群上设置和配置 Prometheus 指标收集 在 Amazon ECS 集群上设置和配置 Prometheus 指标收集

要从 Amazon ECS 集群收集 Prometheus 指标，您可以使用 CloudWatch 代理作为收集器或使用 Amazon Distro for OpenTelemetry 收集器。有关使用 Amazon Distro for OpenTelemetry 收集器的信息，请参阅 <https://aws-otel.github.io/docs/getting-started/container-insights/ecs-prometheus>。

下面几节介绍如何使用 CloudWatch 代理作为收集器来检索 Prometheus 指标。您可以在运行 Amazon ECS 的集群上安装带有 Prometheus 监控功能的 CloudWatch 代理，还可以选择配置该代理以抓取其他目标。这些部分还提供用于设置示例工作负载的可选教程，以使用 Prometheus 监控进行测试。

Amazon ECS 上的 Container Insights 支持 Prometheus 指标的以下启动类型和网络模式组合：

Amazon ECS 启动类型	支持的网络模式
EC2 (Linux)	网桥、主机和 awsvpc
Fargate	awsvpc

VPC 安全组要求

Prometheus 工作负载的安全组的入口规则必须向 CloudWatch 代理打开 Prometheus 端口，以便通过私有 IP 抓取 Prometheus 指标。

CloudWatch 代理的安全组的出口规则必须允许 CloudWatch 代理通过私有 IP 连接到 Prometheus 工作负载的端口。

主题

- [在 Amazon ECS 集群上安装带有 Prometheus 指标收集功能的 CloudWatch 代理 \(p. 419\)](#)
- [抓取其他 Prometheus 源并导入这些指标 \(p. 423\)](#)
- [\(可选 \) 为 Prometheus 指标测试设置示例容器化 Amazon ECS 工作负载 \(p. 431\)](#)

在 Amazon ECS 集群上安装带有 Prometheus 指标收集功能的 CloudWatch 代理

本节介绍如何在运行 Amazon ECS 的集群中设置带有 Prometheus 监控功能的 CloudWatch 代理。执行此操作后，代理会自动抓取并导入该集群中运行的以下工作负载的指标。

- Amazon App Mesh
- Java/JMX

您还可以将代理配置为从其他 Prometheus 工作负载和源抓取及导入指标。

设置 IAM 角色

您需要两个 IAM 角色来执行 CloudWatch 代理任务定义。如果您指定 Amazon CloudFormation 堆栈中的 **CreateIAMRoles=True** 让 Container Insights 为您创建这些角色，则将使用正确的权限创建这些角色。如果要自己创建或使用现有角色，则需要以下角色和权限。

- CloudWatch 代理 ECS 任务角色 – CloudWatch 代理容器使用此角色。它必须包括 CloudWatchAgentServerPolicy 策略以及包含以下只读权限的客户管理策略：
 - `ec2:DescribeInstances`
 - `ecs:ListTasks`
 - `ecs:ListServices`
 - `ecs:DescribeContainerInstances`
 - `ecs:DescribeServices`
 - `ecs:DescribeTasks`
 - `ecs:DescribeTaskDefinition`
- CloudWatch 代理 ECS 任务执行角色 – 这是 Amazon ECS 启动和执行容器所需的角色。确保您的任务执行角色附加了 AmazonSSMReadOnlyAccess、AmazonECSTaskExecutionRolePolicy 和 CloudWatchAgentServerPolicy 策略。如果您需要存储更敏感的数据以供 Amazon ECS 使用，请参阅[指定敏感数据](#)以了解更多信息。

使用 Amazon CloudFormation 安装带有 Prometheus 监控功能的 CloudWatch 代理

您使用 Amazon CloudFormation 为 Amazon ECS 集群安装带有 Prometheus 监控功能的 CloudWatch 代理。以下列表显示了您将在 Amazon CloudFormation 模板中使用的参数。

- `ECSClusterName` – 指定目标 Amazon ECS 集群。
- `CreateIAMRoles` – 指定 **True** 为 Amazon ECS 任务角色和 Amazon ECS 任务执行角色创建新角色。指定 **False** 重新使用现有角色。
- `TaskRoleName` – 如果您为 `CreateIAMRoles` 指定 **True**，这会指定用于新 Amazon ECS 任务角色的名称。如果您为 `CreateIAMRoles` 指定 **False**，这会指定要用作 Amazon ECS 任务角色的现有角色。
- `ExecutionRoleName` – 如果您为 `CreateIAMRoles` 指定 **True**，这会指定用于新 Amazon ECS 任务执行角色的名称。如果您为 `CreateIAMRoles` 指定 **False**，这会指定要用作 Amazon ECS 任务执行角色的现有角色。
- `ECSNetworkMode` – 如果您使用 EC2 启动类型，请在此处指定网络模式。必须是 **bridge** 或 **host**。
- `ECSLaunchType` – 指定 **fargate** 或 **EC2**。
- `SecurityGroupID` – 如果 `ECSNetworkMode` 是 **awsvpc**，请在此处指定安全组 ID。
- `SubnetID` – 如果 `ECSNetworkMode` 是 **awsvpc**，请在此处指定子网 ID。

命令示例

本节包含在各种情况下安装带有 Prometheus 监控功能的 Container Insights 的 Amazon CloudFormation 示例命令。

Note

默认情况下，以下设置步骤以匿名用户身份从 Docker Hub 拉取容器镜像。这种拉取可能会受到速率限制。有关更多信息，请参阅[CloudWatch 代理容器镜像 \(p. 354\)](#)。

在桥式网络模式下为 Amazon ECS 集群创建 Amazon CloudFormation 堆栈

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_NETWORK_MODE=bridge
export CREATE_IAM_ROLES=True
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/
master/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/
cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml
```

```
aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
--template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
--parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
               ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
               ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
               ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
               ParameterKey=ExecutionRoleName,ParameterValue=${ECS_EXECUTION_ROLE_NAME} \
--capabilities CAPABILITY_NAMED_IAM \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

在主机网络模式下为 Amazon ECS 集群创建 Amazon CloudFormation 堆栈

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_NETWORK_MODE=host
export CREATE_IAM_ROLES=True
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/master/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
--template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
--parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
               ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
               ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
               ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
               ParameterKey=ExecutionRoleName,ParameterValue=${ECS_EXECUTION_ROLE_NAME} \
\
--capabilities CAPABILITY_NAMED_IAM \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

在 awsvpc 网络模式下为 Amazon ECS 集群创建 Amazon CloudFormation 堆栈

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_LAUNCH_TYPE=EC2
export CREATE_IAM_ROLES=True
export ECS_CLUSTER_SECURITY_GROUP=your_security_group_eg_sg-xxxxxxxxxx
export ECS_CLUSTER_SUBNET=your_subnet_eg_subnet-xxxxxxxxxx
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/master/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-${ECS_LAUNCH_TYPE}-awsvpc \
--template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
--parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
               ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
               ParameterKey=ECSLaunchType,ParameterValue=${ECS_LAUNCH_TYPE} \
```



```

        ParameterKey=SecurityGroupID,ParameterValue=${ECS_CLUSTER_SECURITY_GROUP}
    \
        ParameterKey=SubnetID,ParameterValue=${ECS_CLUSTER_SUBNET} \
        ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
        ParameterKey=ExecutionRoleName,ParameterValue=${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}

```

在 awsvpc 网络模式下为 Fargate 集群创建 Amazon CloudFormation 堆栈

```

export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_LAUNCH_TYPE=FARGATE
export CREATE_IAM_ROLES=True
export ECS_CLUSTER_SECURITY_GROUP=your_security_group_eg_sg-xxxxxxxxxx
export ECS_CLUSTER_SUBNET=your_subnet_eg_subnet-xxxxxxxxxx
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/
master/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/
cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-
${ECS_LAUNCH_TYPE}-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
        ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
        ParameterKey=ECSLaunchType,ParameterValue=${ECS_LAUNCH_TYPE} \
        ParameterKey=SecurityGroupID,ParameterValue=${ECS_CLUSTER_SECURITY_GROUP}
    \
        ParameterKey=SubnetID,ParameterValue=${ECS_CLUSTER_SUBNET} \
        ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
        ParameterKey=ExecutionRoleName,ParameterValue=${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}

```

Amazon CloudFormation 堆栈创建的 Amazon 资源

下表列出了 Amazon 资源，该资源在 Amazon ECS 集群上使用 Amazon CloudFormation 设置带有 Prometheus 监控功能的 Container Insights 时创建。

资源类型	资源名称	注释
AWS::SSM::Parameter	AmazonCloudWatch-CWAgentConfig- \$ECS_CLUSTER_NAME- \$ECS_LAUNCH_TYPE- \$ECS_NETWORK_MODE	这是具有默认 App Mesh 和 Java/JMX 嵌入式指标格式定义的 CloudWatch 代理。
AWS::SSM::Parameter	AmazonCloudWatch- PrometheusConfigName- \$ECS_CLUSTER_NAME- \$ECS_LAUNCH_TYPE- \$ECS_NETWORK_MODE	这是 Prometheus 抓取配置。
AWS::IAM::Role	\$ECS_TASK_ROLE_NAME。	Amazon ECS 任务角色。这仅在您为 CREATE_IAM_ROLES 指定 True 时创建。

资源类型	资源名称	注释
AWS::IAM::Role	<code>\${ECS_EXECUTION_ROLE_NAME}</code>	Amazon ECS 任务执行角色。这仅在您为 <code>CREATE_IAM_ROLES</code> 指定 <code>True</code> 时创建。
AWS::ECS::TaskDefinition	<code>agent-prometheus-\${ECS_CLUSTER_NAME}-\${ECS_LAUNCH_TYPE}-\${ECS_NETWORK_MODE}</code>	
AWS::ECS::Service	<code>agent-prometheus-replica-service-\${ECS_LAUNCH_TYPE}-\${ECS_NETWORK_MODE}</code>	

使用 Prometheus 监控功能删除 CloudWatch 代理的 Amazon CloudFormation 堆栈

要从 Amazon ECS 集群中删除 CloudWatch 代理，请输入以下命令。

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export CLOUDFORMATION_STACK_NAME=your_cloudformation_stack_name

aws cloudformation delete-stack \
--stack-name ${CLOUDFORMATION_STACK_NAME} \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

抓取其他 Prometheus 源并导入这些指标

具有 Prometheus 监控功能的 CloudWatch 代理需要两种配置来抓取 Prometheus 指标。一个用于 Prometheus 文档中 [<scrape_config>](#) 记录的标准 Prometheus 配置。另一种配置是 CloudWatch 代理配置文件。

对于 Amazon ECS 集群，配置通过 Amazon ECS 任务定义中的密钥与 Amazon Systems Manager 的 Parameter Store 集成：

- 密钥 `PROMETHEUS_CONFIG_CONTENT` 用于 Prometheus 抓取配置。
- 密钥 `CW_CONFIG_CONTENT` 用于 CloudWatch 代理配置。

要抓取其他 Prometheus 指标源并将这些指标导入 CloudWatch，您需要修改 Prometheus 抓取配置和 CloudWatch 代理配置，然后使用更新的配置重新部署代理。

VPC 安全组要求

Prometheus 工作负载的安全组的入口规则必须向 CloudWatch 代理打开 Prometheus 端口，以便通过私有 IP 抓取 Prometheus 指标。

CloudWatch 代理的安全组的出口规则必须允许 CloudWatch 代理通过私有 IP 连接到 Prometheus 工作负载的端口。

Prometheus 抓取配置

CloudWatch 代理支持标准的 Prometheus 抓取配置，如 Prometheus 文档中的 [<scrape_config>](#) 所述。您可以编辑此部分以更新此文件中已有的配置，并添加其他 Prometheus 抓取目标。默认情况下，示例配置文件包含以下全局配置行：

```
global:
  scrape_interval: 1m
```

```
scrape_timeout: 10s
```

- `scrape_interval` – 定义抓取目标的频率。
- `scrape_timeout` – 定义抓取请求超时之前的等待时间。

您还可以在作业级别为这些设置定义不同的值，以覆盖全局配置。

Prometheus 抓取任务

CloudWatch 代理 YAML 文件已配置了一些默认的抓取任务。例如，在 Amazon ECS 的 YAML 文件（例如 `cwagent-ecs-prometheus-metric-for-bridge-host.yaml`）中，默认抓取任务在 `ecs_service_discovery` 部分中配置。

```
"ecs_service_discovery": {
  "sd_frequency": "1m",
  "sd_result_file": "/tmp/cwagent-ecs-auto_sd.yaml",
  "docker_label": {
  },
  "task_definition_list": [
    {
      "sd_job_name": "ecs-appmesh-colors",
      "sd_metrics_ports": "9901",
      "sd_task_definition_arn_pattern": ".*:task-definition/.*-
ColorTeller-(white):[0-9]+",
      "sd_metrics_path": "/stats/prometheus"
    },
    {
      "sd_job_name": "ecs-appmesh-gateway",
      "sd_metrics_ports": "9901",
      "sd_task_definition_arn_pattern": ".*:task-definition/.*-
ColorGateway:[0-9]+",
      "sd_metrics_path": "/stats/prometheus"
    }
  ]
}
```

这些默认目标中的每个目标都会被抓取，并使用嵌入式指标格式在日志事件中将指标发送到 CloudWatch。有关更多信息，请参阅[采用 CloudWatch 嵌入式指标格式摄取高基数日志以生成指标 \(p. 754\)](#)。

来自 Amazon ECS 集群的日志事件存储在 `/aws/ecs/containerinsights/cluster_name/prometheus` 日志组中。

每个抓取作业都包含在此日志组中的不同日志流中。

要添加新的抓取目标，请在 YAML 文件 `ecs_service_discovery` 部分下的 `task_definition_list` 部分中添加一个新条目，然后重新启动代理。有关此过程的示例，请参阅[添加新 Prometheus 抓取目标的教程：Prometheus API 服务器指标 \(p. 463\)](#)。

Prometheus 的 CloudWatch 代理配置

CloudWatch 代理配置文件在 `metrics_collected` 下面有一个 `prometheus` 部分用于 Prometheus 抓取配置。它包含以下配置选项：

- `cluster_name` – 指定要在日志事件中添加为标签的集群名称。此字段为可选项。如果省略此项，代理可以检测 Amazon ECS 集群名称。
- `log_group_name` – 指定已抓取 Prometheus 指标的日志组名称。此字段为可选项。如果省略此项，CloudWatch 会将 `/aws/ecs/containerinsights/cluster_name/prometheus` 用于来自 Amazon ECS 集群的日志。
- `prometheus_config_path` – 指定 Prometheus 抓取配置文件路径。如果此字段的值以 `env:` 开头，Prometheus 抓取配置文件内容将从容器的环境变量中检索。请勿更改此字段。

- `ecs_service_discovery` – 是指定 Amazon ECS Prometheus 目标自动发现功能配置的部分。支持两种模式来发现 Prometheus 目标：根据容器的 docker 标签的发现或根据 Amazon ECS 任务定义 ARN 正则表达式的发现。您可以同时使用这两种模式，CloudWatch 代理将根据 `{private_ip}:{port}/{metrics_path}` 对发现的目标进行重复去除。

`ecs_service_discovery` 部分包含以下字段：

- `sd_frequency` 是发现 Prometheus 导出器的频率。指定数字和单位后缀。例如，`1m` 表示每分钟一次或 `30s` 表示每 30 秒一次。有效的单位后缀为 `ns`、`us`、`ms`、`s`、`m` 和 `h`。

此字段为可选项。默认值为 60 秒（1 分钟）。
- `sd_target_cluster` 是用于自动发现的目标 Amazon ECS 集群名称。此字段为可选项。默认名称为安装 CloudWatch 代理的 Amazon ECS 集群的名称。
- `sd_cluster_region` 是目标 Amazon ECS 集群的区域。此字段为可选项。默认区域为安装 CloudWatch 代理的 Amazon ECS 集群的区域。
- `sd_result_file` 是 Prometheus 目标结果的 YAML 文件的路径。Prometheus 抓取配置将引用此文件。
- `docker_label` 是可选部分，您可以使用它来指定基于 docker 标签的服务发现的配置。如果省略此部分，则不会使用基于 docker 标签的发现。此部分包含以下字段：
 - `sd_port_label` 是容器的 docker 标签名称，用于指定 Prometheus 指标的容器端口。默认值为 `ECS_PROMETHEUS_EXPORTER_PORT`。如果容器没有此 docker 标签，CloudWatch 代理将跳过它。
 - `sd_metrics_path_label` 是容器的 docker 标签名称，用于指定 Prometheus 指标路径。默认值为 `ECS_PROMETHEUS_METRICS_PATH`。如果容器没有此 docker 标签，代理会采用默认路径 `/metrics`。
 - `sd_job_name_label` 是容器的 docker 标签名称，用于指定 Prometheus 抓取任务名称。默认值为 `job`。如果容器没有此 docker 标签，CloudWatch 代理会使用 Prometheus 抓取配置中的任务名称。
- `task_definition_list` 是可选部分，可用于指定基于任务定义的服务发现的配置。如果省略此部分，则不会使用基于任务定义地发现。此部分包含以下字段：
 - `sd_task_definition_arn_pattern` 是用于指定要发现的 Amazon ECS 任务定义的模式。这是正则表达式。
 - `sd_metrics_ports` 列出 Prometheus 指标的 `containerPort`。用分号分隔 `containerPorts`。
 - `sd_container_name_pattern` 指定 Amazon ECS 任务容器名称。这是正则表达式。
 - `sd_metrics_path` 指定 Prometheus 指标路径。如果省略此项，代理会采用默认路径 `/metrics`。
 - `sd_job_name` 指定 Prometheus 抓取任务名称。如果省略此字段，CloudWatch 代理会使用 Prometheus 抓取配置中的任务名称。
- `service_name_list_for_tasks` 是可选部分，可用于指定基于服务名称的发现的配置。如果省略此部分，则不会使用基于服务名称的发现。此部分包含以下字段：
 - `sd_service_name_pattern` 是用于指定要在其中发现任务的 Amazon ECS 服务的模式。这是正则表达式。
 - `sd_metrics_ports` 列出 Prometheus 指标的 `containerPort`。用分号分隔多个 `containerPorts`。
 - `sd_container_name_pattern` 指定 Amazon ECS 任务容器名称。这是正则表达式。
 - `sd_metrics_path` 指定 Prometheus 指标路径。如果省略此项，代理会采用默认路径 `/metrics`。
 - `sd_job_name` 指定 Prometheus 抓取任务名称。如果省略此字段，CloudWatch 代理会使用 Prometheus 抓取配置中的任务名称。
- `metric_declaration` – 是指定要生成的采用嵌入式指标格式的日志数组的部分。默认情况下，CloudWatch 代理从中进行导入的每个 Prometheus 源都有 `metric_declaration` 部分。这些部分各包括以下字段：
 - `label_matcher` 是一个正则表达式，用于检查 `source_labels` 中列出的标签的值。匹配的指标将启用，以包含在发送到 CloudWatch 的嵌入式指标格式中。

如果您在 `source_labels` 中指定了多个标签，我们建议您不要在 `label_matcher` 的正则表达式中使用 `^` 或 `$` 字符。

-
- `source_labels` 指定由 `label_matcher` 检查的标签的值。

- `label_separator` 指定要在 `label_matcher` 行中使用的分隔符（如果指定了多个 `source_labels`）。默认为 `;`。您可以在以下示例中看到 `label_matcher` 行中使用的此默认值。
- `metric_selectors` 是一个正则表达式，用于指定要收集并发送到 CloudWatch 的指标。
- `dimensions` 是要用作每个选定指标的 CloudWatch 维度的标签列表。

请参阅以下 `metric_declaration` 示例。

```
"metric_declaration": [
  {
    "source_labels": [ "Service", "Namespace" ],
    "label_matcher": "(.*node-exporter.*|.*kube-dns.*);kube-system$",
    "dimensions": [
      [ "Service", "Namespace" ]
    ],
    "metric_selectors": [
      "^coredns_dns_request_type_count_total$"
    ]
  }
]
```

此示例配置嵌入式指标格式部分，以便在满足以下条件时作为日志事件发送：

- `Service` 的值包含 `node-exporter` 或 `kube-dns`。
- `Namespace` 的值为 `kube-system`。
- Prometheus 指标 `coredns_dns_request_type_count_total` 同时包含 `Service` 和 `Namespace` 标签。

发送的日志事件包括以下突出显示的部分：

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Name": "coredns_dns_request_type_count_total"
        }
      ],
      "Dimensions": [
        [
          "Namespace",
          "Service"
        ]
      ],
      "Namespace": "ContainerInsights/Prometheus"
    }
  ],
  "Namespace": "kube-system",
  "Service": "kube-dns",
  "coredns_dns_request_type_count_total": 2562,
  "eks_amazonaws_com_component": "kube-dns",
  "instance": "192.168.61.254:9153",
  "job": "kubernetes-service-endpoints",
  ...
}
```

Amazon ECS 集群上自动发现的详细指南

Prometheus 提供了数十种动态服务发现机制，如 `<scrape_config>` 中所述。但是，Amazon ECS 没有内置服务发现。CloudWatch 代理添加了此机制。

启用 Amazon ECS Prometheus 服务发现后，CloudWatch 代理会定期对 Amazon ECS 和 Amazon EC2 前端进行以下 API 调用，以检索目标 ECS 集群中正在运行的 ECS 任务的元数据。

```
EC2:DescribeInstances
ECS:ListTasks
ECS:ListServices
ECS:DescribeContainerInstances
ECS:DescribeServices
ECS:DescribeTasks
ECS:DescribeTaskDefinition
```

CloudWatch 代理使用元数据来扫描 ECS 集群内的 Prometheus 目标。CloudWatch 代理支持三种服务发现模式：

- 基于容器 docker 标签的服务发现
- 基于 ECS 任务定义 ARN 正则表达式的服务发现
- 基于 ECS 服务名称正则表达式的服务发现

所有模式均可同时使用。CloudWatch 代理会根据 {private_ip}:{port}/{metrics_path} 对发现的目标进行重复去除。

所有发现的目标均写入由 CloudWatch 代理容器内的 `sd_result_file` 配置字段指定的结果文件。以下为示例结果文件：

```
- targets:
- 10.6.1.95:32785
labels:
  __metrics_path__: /metrics
  ECS_PROMETHEUS_EXPORTER_PORT: "9406"
  ECS_PROMETHEUS_JOB_NAME: demo-jar-ec2-bridge-dynamic
  ECS_PROMETHEUS_METRICS_PATH: /metrics
  InstanceType: t3.medium
  LaunchType: EC2
  SubnetId: subnet-123456789012
  TaskDefinitionFamily: demo-jar-ec2-bridge-dynamic-port
  TaskGroup: family:demo-jar-ec2-bridge-dynamic-port
  TaskRevision: "7"
  VpcId: vpc-01234567890
  container_name: demo-jar-ec2-bridge-dynamic-port
  job: demo-jar-ec2-bridge-dynamic
- targets:
- 10.6.3.193:9404
labels:
  __metrics_path__: /metrics
  ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_B: "9404"
  ECS_PROMETHEUS_JOB_NAME: demo-tomcat-ec2-bridge-mapped-port
  ECS_PROMETHEUS_METRICS_PATH: /metrics
  InstanceType: t3.medium
  LaunchType: EC2
  SubnetId: subnet-123456789012
  TaskDefinitionFamily: demo-tomcat-ec2-bridge-mapped-port
  TaskGroup: family:demo-jar-tomcat-bridge-mapped-port
  TaskRevision: "12"
  VpcId: vpc-01234567890
  container_name: demo-tomcat-ec2-bridge-mapped-port
  job: demo-tomcat-ec2-bridge-mapped-port
```

您可以直接将此结果文件与基于 Prometheus 文件的服务发现集成。有关基于 Prometheus 文件的服务发现的更多信息，请参阅 [<file_sd_config>](#)。

假设将结果文件写入 `/tmp/cwagent_ecs_auto_sd.yaml`。以下 Prometheus 抓取配置将使用它。

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: cwagent-ecs-file-sd-config
    sample_limit: 10000
    file_sd_configs:
      - files: [ "/tmp/cwagent_ecs_auto_sd.yaml" ]
```

CloudWatch 代理还会为发现的目标添加以下附加标签。

- container_name
- TaskDefinitionFamily
- TaskRevision
- TaskGroup
- StartedBy
- LaunchType
- job
- __metrics_path__
- Docker 标签

当集群具有 EC2 启动类型时，会添加以下三个标签。

- InstanceType
- VpcId
- SubnetId

Note

与正则表达式 `[a-zA-Z][a-zA-Z0-9_]*` 不匹配的 Docker 标签将被筛选掉。这与 Prometheus 文档中[配置文件](#)的 `label_name` 中列出的 Prometheus 惯例相匹配。

ECS 服务发现配置示例

本节包括演示 ECS 服务发现的示例。

示例 1

```
"ecs_service_discovery": {
  "sd_frequency": "1m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
  }
}
```

此示例启用基于 docker 标签的服务发现。CloudWatch 代理将每分钟查询一次 ECS 任务的元数据，并将发现的目标写入 CloudWatch 代理容器内的 `/tmp/cwagent_ecs_auto_sd.yaml` 文件中。

`docker_label` 部分中 `sd_port_label` 的默认值为 `ECS_PROMETHEUS_EXPORTER_PORT`。如果 ECS 任务中任何正在运行的容器具有 `ECS_PROMETHEUS_EXPORTER_PORT` docker 标签，CloudWatch 代理将其值作为 `container port` 来扫描容器的所有公开端口。如果匹配，则使用映射的主机端口和容器的私有 IP，以 `private_ip:host_port` 格式来构建 Prometheus 导出器目标。

`docker_label` 部分中 `sd_metrics_path_label` 的默认值为 `ECS_PROMETHEUS_METRICS_PATH`。如果容器具有此 docker 标签，则其值将用作 `__metrics_path__`。如果容器没有此标签，则使用默认值 `/metrics`。

`docker_label` 部分中 `sd_job_name_label` 的默认值为 `job`。如果容器具有此 `docker` 标签，则其值将作为一个目标标签附加，以替换 Prometheus 配置中指定的默认任务名称。此 `docker` 标签的值用作 CloudWatch Logs 日志组中的日志流名称。

示例 2

```
"ecs_service_discovery": {
  "sd_frequency": "15s",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
    "sd_port_label": "ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_A",
    "sd_job_name_label": "ECS_PROMETHEUS_JOB_NAME"
  }
}
```

此示例启用基于 `docker` 标签的服务发现。CloudWatch 代理将每 15 秒查询一次 ECS 任务的元数据，并将发现的目标写入 CloudWatch 代理容器内的 `/tmp/cwagent_ecs_auto_sd.yaml` 文件中。带有 `ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_A` 的 `docker` 标签的容器将被扫描。`docker` 标签 `ECS_PROMETHEUS_JOB_NAME` 的值用作任务名称。

示例 3

```
"ecs_service_discovery": {
  "sd_frequency": "5m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "task_definition_list": [
    {
      "sd_job_name": "java-prometheus",
      "sd_metrics_path": "/metrics",
      "sd_metrics_ports": "9404; 9406",
      "sd_task_definition_arn_pattern": ".*:task-definition/.*/javajmx.*:[0-9]+"
    },
    {
      "sd_job_name": "envoy-prometheus",
      "sd_metrics_path": "/stats/prometheus",
      "sd_container_name_pattern": "^envoy$",
      "sd_metrics_ports": "9901",
      "sd_task_definition_arn_pattern": ".*:task-definition/.*/appmesh.*:23"
    }
  ]
}
```

此示例启用基于 ECS 任务定义 ARN 正则表达式的服务发现。CloudWatch 代理将每五分钟查询一次 ECS 任务的元数据，并将发现的目标写入 CloudWatch 代理容器内的 `/tmp/cwagent_ecs_auto_sd.yaml` 文件中。

定义了两个任务定义 ARN 正则表达式部分：

- 对于第一部分，过滤 ECS 任务定义 ARN 中的 ECS 任务以及 `javajmx`，以进行容器端口扫描。如果这些 ECS 任务中的容器在 9404 或 9406 上公开容器端口，则映射的主机端口和容器的私有 IP 会用于创建 Prometheus 导出器目标。`sd_metrics_path` 的值将 `__metrics_path__` 设置为 `/metrics`。因此 CloudWatch 代理将从 `private_ip:host_port/metrics` 中抓取 Prometheus 指标，抓取的指标发送到日志组 `/aws/ecs/containerinsights/cluster_name/prometheus` 中 CloudWatch Logs 中的 `java-prometheus` 日志流。
- 对于第二部分，过滤 ECS 任务定义 ARN 中带有 `appmesh` 的 ECS 任务以及 `:23` 的 `version`，以进行容器端口扫描。对于在 9901 上公开容器端口的、名称为 `envoy` 的容器，映射的主机端口以及容器的私有 IP 用于创建 Prometheus 导出器目标。这些 ECS 任务中的值会公开 9404 或 9406 上的容器端口，映射的主机端口以及容器的私有 IP 用于创建 Prometheus 导出器目标。`sd_metrics_path` 的值将 `__metrics_path__` 设置为 `/stats/prometheus`。因此，CloudWatch 代理将从 `private_ip:host_port/stats/prometheus` 中抓取 Prometheus 指标，并将抓取的指标发送到日

志组 `/aws/ecs/containerinsights/cluster_name/prometheus` 中的 CloudWatch Logs 中的 `envoy-prometheus` 日志流。

示例 4。

```
"ecs_service_discovery": {
  "sd_frequency": "5m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "service_name_list_for_tasks": [
    {
      "sd_job_name": "nginx-prometheus",
      "sd_metrics_path": "/metrics",
      "sd_metrics_ports": "9113",
      "sd_service_name_pattern": "^nginx-.*"
    },
    {
      "sd_job_name": "haproxy-prometheus",
      "sd_metrics_path": "/stats/metrics",
      "sd_container_name_pattern": "^haproxy$"
      "sd_metrics_ports": "8404",
      "sd_service_name_pattern": ".*haproxy-service.*"
    }
  ]
}
```

本示例启用基于 ECS 服务名称正则表达式的服务发现。CloudWatch 代理将每五分钟查询一次 ECS 服务的元数据，并将发现的目标写入 CloudWatch 代理容器内的 `/tmp/cwagent_ecs_auto_sd.yaml` 文件中。

定义了两个服务名称正则表达式部分：

- 对于第一部分，过滤与 ECS 服务关联的 ECS 任务（名称与正则表达式 `^nginx-.*` 相匹配），以进行容器端口扫描。如果这些 ECS 任务中的容器在 9113 上公开容器端口，则映射的主机端口和容器的私有 IP 将用于创建 Prometheus 导出器目标。`sd_metrics_path` 的值将 `__metrics_path__` 设置为 `/metrics`。因此，CloudWatch 代理将从 `private_ip:host_port/metrics` 中抓取 Prometheus 指标，并将抓取的指标发送到日志组 `/aws/ecs/containerinsights/cluster_name/prometheus` 中 CloudWatch Logs 中的 `nginx-prometheus` 日志流。
- 或第二部分，过滤与 ECS 服务关联的 ECS 任务（名称与正则表达式 `.*haproxy-service.*` 匹配），以进行容器端口扫描。对于在 8404 上公开容器端口、名称为 `haproxy` 的容器，映射的主机端口以及容器的私有 IP 用于创建 Prometheus 导出器目标。`sd_metrics_path` 的值将 `__metrics_path__` 设置为 `/stats/metrics`。因此，CloudWatch 代理将从 `private_ip:host_port/stats/metrics` 中抓取 Prometheus 指标，并将抓取的指标发送到日志组 `/aws/ecs/containerinsights/cluster_name/prometheus` 中 CloudWatch Logs 中的 `haproxy-prometheus` 日志流。

示例 5

```
"ecs_service_discovery": {
  "sd_frequency": "1m30s",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
    "sd_port_label": "MY_PROMETHEUS_EXPORTER_PORT_LABEL",
    "sd_metrics_path_label": "MY_PROMETHEUS_METRICS_PATH_LABEL",
    "sd_job_name_label": "MY_PROMETHEUS_METRICS_NAME_LABEL"
  }
  "task_definition_list": [
    {
      "sd_metrics_ports": "9150",
      "sd_task_definition_arn_pattern": ".*memcached.*"
    }
  ]
}
```



```
}
```

本示例启用了两种 ECS 服务发现模式。CloudWatch 代理将每 90 秒查询一次 ECS 任务的元数据，并将发现的目标写入 CloudWatch 代理容器内的 `/tmp/cwagent_ecs_auto_sd.yaml` 文件中。

对于基于 docker 的服务发现配置：

- 过滤带有 docker 标签 `MY_PROMETHEUS_EXPORTER_PORT_LABEL` 的 ECS 任务，以进行 Prometheus 端口扫描。目标 Prometheus 容器端口由 label `MY_PROMETHEUS_EXPORTER_PORT_LABEL` 的值指定。
- docker 标签 `MY_PROMETHEUS_EXPORTER_PORT_LABEL` 的值用于 `__metrics_path__`。如果容器没有此 docker 标签，则使用默认值 `/metrics`。
- docker 标签 `MY_PROMETHEUS_EXPORTER_PORT_LABEL` 的值用作任务标签。如果容器没有此 docker 标签，则使用 Prometheus 配置中定义的任务名称。

对于基于 ECS 任务定义 ARN 正则表达式的服务发现配置：

- 过滤 ECS 任务定义 ARN 中带 `memcached` 的 ECS 任务，以进行容器端口扫描。根据 `sd_metrics_ports` 定义，目标 Prometheus 容器端口是 9150。使用默认指标路径 `/metrics`。使用 Prometheus 配置中定义的任务名称。

(可选) 为 Prometheus 指标测试设置示例容器化 Amazon ECS 工作负载

要测试 CloudWatch Container Insights 中的 Prometheus 指标支持，您可设置以下一个或多个容器化工作负载。具有 Prometheus 支持的 CloudWatch 代理会自动从这些工作负载中的每一个收集指标。要查看默认情况下收集的指标，请参阅 [CloudWatch 代理收集的 Prometheus 指标 \(p. 480\)](#)。

主题

- [Amazon ECS 集群的示例 App Mesh 工作负载 \(p. 431\)](#)
- [Amazon ECS 集群的示例 Java/JMX 工作负载 \(p. 432\)](#)
- [Amazon ECS 集群的示例 NGINX 工作负载 \(p. 433\)](#)
- [Amazon ECS 集群的示例 NGINX Plus 工作负载 \(p. 438\)](#)
- [添加新 Prometheus 抓取目标的教程：Amazon ECS 上的 Memcached \(p. 445\)](#)
- [在 Amazon ECS Fargate 上抓取 Redis Prometheus 指标的教程 \(p. 450\)](#)

Amazon ECS 集群的示例 App Mesh 工作负载

要从 Amazon ECS 的示例 Prometheus 工作负载收集指标，您必须在集群中运行 Container Insights。有关安装 Container Insights 的信息，请参阅 [在 Amazon ECS 上设置 Container Insights \(p. 355\)](#)。

首先，按照本**演练**在 Amazon ECS 集群上部署示例颜色应用程序。完成后，您将在端口 9901 上公开 App Mesh Prometheus 指标。

接下来，按照以下步骤在安装颜色应用程序的同一 Amazon ECS 集群上，安装带有 Prometheus 监控功能的 CloudWatch 代理。本节中的步骤以桥式网络模式安装 CloudWatch 代理。

您在演练中设置的环境变量 `ENVIRONMENT_NAME`、`AWS_PROFILE` 和 `AWS_DEFAULT_REGION` 也将在以下步骤中使用。

安装带有 Prometheus 监控功能的 CloudWatch 代理以进行测试

1. 通过输入以下命令，下载 Amazon CloudFormation 模板。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/master/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml
```

2. 通过输入以下命令，设置网络模式。

```
export ECS_CLUSTER_NAME=${ENVIRONMENT_NAME}
export ECS_NETWORK_MODE=bridge
```

3. 输入以下命令以创建 Amazon CloudFormation 堆栈。

```
aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
  --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
  --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
    ParameterKey=CreateIAMRoles,ParameterValue=True \
    ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
    ParameterKey=TaskRoleName,ParameterValue=CWAgent-Prometheus-TaskRole-
${ECS_CLUSTER_NAME} \
    ParameterKey=ExecutionRoleName,ParameterValue=CWAgent-Prometheus-
ExecutionRole-${ECS_CLUSTER_NAME} \
  --capabilities CAPABILITY_NAMED_IAM \
  --region ${AWS_DEFAULT_REGION} \
  --profile ${AWS_PROFILE}
```

4. （可选）创建 Amazon CloudFormation 堆栈后，您会看到 CREATE_COMPLETE 消息。如果要在看到该消息之前检查状态，请输入以下命令。

```
aws cloudformation describe-stacks \
  --stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
  --query 'Stacks[0].StackStatus' \
  --region ${AWS_DEFAULT_REGION} \
  --profile ${AWS_PROFILE}
```

故障排除

演练中的步骤使用 jq 来分析 Amazon CLI 的输出结果。有关安装 jq 代理的更多信息，请参阅 [jq](#)。使用以下命令将 Amazon CLI 的默认输出格式设置为 JSON，以便 jq 可以对其进行正确分析。

```
$ aws configure
```

当响应变为 Default output format 时，输入 **json**。

使用 Prometheus 监控功能卸载 CloudWatch 代理

完成测试后，输入以下命令以通过 Amazon CloudFormation 删除堆栈来卸载 CloudWatch 代理。

```
aws cloudformation delete-stack \
  --stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
  --region ${AWS_DEFAULT_REGION} \
  --profile ${AWS_PROFILE}
```

Amazon ECS 集群的示例 Java/JMX 工作负载

JMX Exporter 是 Prometheus 的官方导出程序，可以将 JMX MBeans 作为 Prometheus 指标进行抓取和公开。有关详细信息，请参阅 [prometheus/jmx_exporter](#)。

支持 Prometheus 的 CloudWatch 代理根据 Amazon ECS 集群中的服务发现配置抓取 Java/JMX Prometheus 指标。您可以将 JMX Exporter 配置为在不同端口或 metrics_path 上公开指标。如果您更改了端口或路径，请更新 CloudWatch 代理配置中的默认 ecs_service_discovery 部分。

要从 Amazon ECS 的示例 Prometheus 工作负载收集指标，您必须在集群中运行 Container Insights。有关安装 Container Insights 的信息，请参阅 [在 Amazon ECS 上设置 Container Insights \(p. 355\)](#)。

为 Amazon ECS 集群安装 Java/JMX 示例工作负载

- 按照这些部分中的步骤创建 Docker 镜像。
 - [示例：具有 Prometheus 指标的 Java Jar 应用程序 Docker 镜像 \(p. 473\)](#)
 - [示例：具有 Prometheus 指标的 Apache Tomcat Docker 镜像 \(p. 472\)](#)
- 在 Amazon ECS 任务定义文件中指定以下两个 docker 标签。然后，您可以将任务定义作为集群中的 Amazon ECS 服务或 Amazon ECS 任务运行。
 - 将 `ECS_PROMETHEUS_EXPORTER_PORT` 设置为指向公开 Prometheus 指标的 `containerPort`。
 - 将 `Java_EMF_Metrics` 设置为 `true`。CloudWatch 代理使用此标志在日志事件中生成嵌入的指标格式。

以下是示例：

```
{
  "family": "workload-java-ec2-bridge",
  "taskRoleArn": "{{task-role-arn}}",
  "executionRoleArn": "{{execution-role-arn}}",
  "networkMode": "bridge",
  "containerDefinitions": [
    {
      "name": "tomcat-prometheus-workload-java-ec2-bridge-dynamic-port",
      "image": "your_docker_image_tag_for_tomcat_with_prometheus_metrics",
      "portMappings": [
        {
          "hostPort": 0,
          "protocol": "tcp",
          "containerPort": 9404
        }
      ],
      "dockerLabels": {
        "ECS_PROMETHEUS_EXPORTER_PORT": "9404",
        "Java_EMF_Metrics": "true"
      }
    }
  ],
  "requiresCompatibilities": [
    "EC2"
  ],
  "cpu": "256",
  "memory": "512"
}
```

Amazon CloudFormation 模板中 CloudWatch 代理的默认设置同时启用基于 docker 标签的服务发现和基于任务定义 ARN 的服务发现。要查看这些默认设置，请参阅 [CloudWatch 代理 YAML 配置文件的第 65 行](#)。带有 `ECS_PROMETHEUS_EXPORTER_PORT` 标签的容器将根据 Prometheus 抓取的指定容器端口自动发现。

CloudWatch 代理的默认设置在同一文件的第 112 行也有 `Java/JMX metric_declaration` 设置。目标容器的所有 docker 标签将作为附加标签添加到 Prometheus 指标中并发送到 CloudWatch Logs。对于带有 docker 标签 `Java_EMF_Metrics="true"` 的 Java/JMX 容器，将生成嵌入式指标格式。

Amazon ECS 集群的示例 NGINX 工作负载

NGINX Prometheus 导出器可以抓取和公开 NGINX 数据作为 Prometheus 指标。此示例将导出器与适用于 Amazon ECS 的 NGINX 反向代理服务结合使用。

有关 NGINX Prometheus 导出器的更多信息，请参阅 Github 上的 [nginx-prometheus-exporter](#)。有关 NGINX 反向代理的更多信息，请参阅 Github 上的 [ecs-nginx-reverse-proxy](#)。

支持 Prometheus 的 CloudWatch 代理根据 Amazon ECS 集群中的服务发现配置抓取 NGINX Prometheus 指标。您可以将 NGINX Prometheus Exporter 配置为在不同端口或路径上公开指标。如果您更改端口或路径，请更新 CloudWatch 代理配置文件中的 `ecs_service_discovery` 部分。

为 Amazon ECS 集群安装 NGINX 反向代理示例工作负载

按照以下步骤安装 NGINX 反向代理示例工作负载。

创建 Docker 镜像

为 NGINX 反向代理示例工作负载创建 Docker 镜像

1. 从 NGINX 反向代理存储库下载以下文件夹：<https://github.com/aws-labs/ecs-nginx-reverse-proxy/tree/master/reverse-proxy/>。
2. 查找 `app` 目录并从该目录构建镜像：

```
docker build -t web-server-app ./path-to-app-directory
```

3. 为 NGINX 构建自定义镜像。首先，创建一个包含以下两个文件的目录：

- 示例 Dockerfile：

```
FROM nginx
COPY nginx.conf /etc/nginx/nginx.conf
```

- 一个 `nginx.conf` 文件，修改自 <https://github.com/aws-labs/ecs-nginx-reverse-proxy/tree/master/reverse-proxy/>：

```
events {
    worker_connections 768;
}

http {
    # Nginx will handle gzip compression of responses from the app server
    gzip on;
    gzip_proxied any;
    gzip_types text/plain application/json;
    gzip_min_length 1000;

    server{
        listen 8080;
        location /stub_status {
            stub_status on;
        }
    }

    server {
        listen 80;

        # Nginx will reject anything not matching /api
        location /api {
            # Reject requests with unsupported HTTP method
            if ($request_method !~ ^(GET|POST|HEAD|OPTIONS|PUT|DELETE)#) {
                return 405;
            }

            # Only requests matching the whitelist expectations will
            # get sent to the application server
        }
    }
}
```

```
proxy_pass http://app:3000;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection 'upgrade';
proxy_set_header Host $host;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_cache_bypass $http_upgrade;
}
}
}
```

Note

stub_status 必须在将 nginx-prometheus-exporter 配置为从中抓取指标的同端口上启用。在我们的示例任务定义中，将 nginx-prometheus-exporter 配置为从端口 8080 抓取指标。

4. 从新目录中的文件构建镜像：

```
docker build -t nginx-reverse-proxy ./path-to-your-directory
```

5. 将新镜像上传到镜像存储库以供日后使用。

创建任务定义以在 Amazon ECS 中运行 NGINX 和 web 服务器应用程序

接下来，设置任务定义。

此任务定义启用 NGINX Prometheus 指标的收集和导出。NGINX 容器跟踪来自应用程序的输入，并将该数据公开到端口 8080，如 nginx.conf 中所设置。NGINX prometheus 导出器容器抓取这些指标，并将其发布到端口 9113，以在 CloudWatch 中使用。

为 NGINX 示例 Amazon ECS 工作负载设置任务定义

1. 使用以下内容创建任务定义 JSON 文件。将 *your-customized-nginx-image* 替换为您自定义的 NGINX 镜像的镜像 URI，并将 *your-web-server-app-image* 替换为您 web 服务器应用程序镜像的镜像 URI。

```
{
  "containerDefinitions": [
    {
      "name": "nginx",
      "image": "your-customized-nginx-image",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "links": [
        "app"
      ]
    },
    {
      "name": "app",
      "image": "your-web-server-app-image",
      "memory": 256,
      "cpu": 256,
      "essential": true
    }
  ],
}
```

```
{
  "name": "nginx-prometheus-exporter",
  "image": "docker.io/nginx/nginx-prometheus-exporter:0.8.0",
  "memory": 256,
  "cpu": 256,
  "essential": true,
  "command": [
    "-nginx.scrape-uri",
    "http://nginx:8080/stub_status"
  ],
  "links": [
    "nginx"
  ],
  "portMappings": [
    {
      "containerPort": 9113,
      "protocol": "tcp"
    }
  ]
},
{
  "networkMode": "bridge",
  "placementConstraints": [],
  "family": "nginx-sample-stack"
}
```

2. 使用以下命令注册任务定义。

```
aws ecs register-task-definition --cli-input-json file://path-to-your-task-definition-  
json
```

3. 通过输入以下命令创建服务以运行任务：

确保不要更改服务名称。我们将使用配置来运行 CloudWatch 代理服务，该配置使用启动它们的服务的名称模式来搜索任务。例如，要让 CloudWatch 代理查找此命令启动的任务，您可以将 `sd_service_name_pattern` 的值指定为 `^nginx-service$`。下一部分将提供更多详细信息。

```
aws ecs create-service \
  --cluster your-cluster-name \
  --service-name nginx-service \
  --task-definition nginx-sample-stack:1 \
  --desired-count 1
```

配置 CloudWatch 代理以抓取 NGINX Prometheus 指标

最后一步是配置 CloudWatch 代理以抓取 NGINX 指标。在此示例中，CloudWatch 代理通过服务名称模式和端口 9113 发现任务，导出器在该端口公开 NGINX 的 prometheus 指标。发现任务且指标可用后，CloudWatch 代理开始将收集的指标发布到日志流 `nginx-prometheus-exporter`。

配置 CloudWatch 代理以抓取 NGINX 指标

1. 通过输入以下命令，下载必要 YAML 文件的最新版本。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-  
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-  
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml
```

2. 使用文本编辑器打开文件，然后在 `resource:CWAgentConfigSSMParameter` 部分的 `value` 密钥中查找完整的 CloudWatch 代理配置。然后，在 `ecs_service_discovery` 部分中，添加以下 `service_name_list_for_tasks` 部分。

```
"service_name_list_for_tasks": [
  {
    "sd_job_name": "nginx-prometheus-exporter",
    "sd_metrics_path": "/metrics",
    "sd_metrics_ports": "9113",
    "sd_service_name_pattern": "^nginx-service$"
  }
],
```

- 在同一文件中，在 `metric_declaration` 部分中添加以下部分以允许 NGINX 指标。请务必遵循现有的缩进模式。

```
{
  "source_labels": ["job"],
  "label_matcher": ".*nginx.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName"]],
  "metric_selectors": [
    "^nginx_.*$"
  ]
},
```

- 如果您尚未在此集群中部署 CloudWatch 代理，请跳至步骤 8。

如果您已经使用 Amazon CloudFormation 将 CloudWatch 代理部署在 Amazon ECS 集群中，您可以通过输入以下命令来创建更改集：

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
  --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
  --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
    ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
    ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
    ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
    ParameterKey=ExecutionRoleName,ParameterValue=${ECS_EXECUTION_ROLE_NAME} \
  --capabilities CAPABILITY_NAMED_IAM \
  --region $AWS_REGION \
  --change-set-name nginx-scraping-support
```

- 打开 Amazon CloudFormation 控制台，地址：<https://console.aws.amazon.com/cloudformation>。
- 查看新创建的变更集 `nginx-scraping-support`。您会看到一项应用于 `CWAgentConfigSSMParameter` 资源的更改。通过输入以下命令，运行变更集并重新启动 CloudWatch 代理任务：

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-${ECS_NETWORK_MODE} \
--region $AWS_REGION
```

- 等待大约 10 秒，然后输入以下命令。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-${ECS_NETWORK_MODE} \
```



```
--region $AWS_REGION
```

8. 如果您是首次在集群上安装带有 Prometheus 指标收集功能的 CloudWatch 代理，请输入以下命令。

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
  --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
  --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
    ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
    ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
    ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
    ParameterKey=ExecutionRoleName,ParameterValue=$ECS_EXECUTION_ROLE_NAME \
  --capabilities CAPABILITY_NAMED_IAM \
  --region $AWS_REGION
```

查看您的 NGINX 指标和日志

您现在可以查看正在收集的 NGINX 指标。

查看示例 NGINX 工作负载的指标

- 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
- 在运行集群的区域中，选择左侧导航窗格中的 Metrics (指标)。查找 ContainerInsights/Prometheus 命名空间以查看指标。
- 要查看 CloudWatch Logs 事件，请在导航窗格中选择 Log groups (日志组)。这些事件位于日志流 *nginx-prometheus-exporter* 中的日志组 */aws/containerinsights/*your_cluster_name*/prometheus* 中。

Amazon ECS 集群的示例 NGINX Plus 工作负载

NGINX Plus 是 NGINX 的商业版本。您必须拥有许可证方可使用它。有关更多信息，请参阅 [NGINX Plus](#)

NGINX Prometheus 导出器可以抓取和公开 NGINX 数据作为 Prometheus 指标。此示例将导出器与适用于 Amazon ECS 的 NGINX Plus 反向代理服务结合使用。

有关 NGINX Prometheus 导出器的更多信息，请参阅 Github 上的 [nginx-prometheus-exporter](#)。有关 NGINX 反向代理的更多信息，请参阅 Github 上的 [ecs-nginx-reverse-proxy](#)。

支持 Prometheus 的 CloudWatch 代理根据 Amazon ECS 集群中的服务发现配置抓取 NGINX Plus Prometheus 指标。您可以将 NGINX Prometheus Exporter 配置为在不同端口或路径上公开指标。如果您更改端口或路径，请更新 CloudWatch 代理配置文件中的 `ecs_service_discovery` 部分。

为 Amazon ECS 集群安装 NGINX Plus 反向代理示例工作负载

按照以下步骤安装 NGINX 反向代理示例工作负载。

创建 Docker 镜像

为 NGINX Plus 反向代理示例工作负载创建 Docker 镜像

- 从 NGINX 反向代理存储库下载以下文件夹：<https://github.com/awslabs/ecs-nginx-reverse-proxy/tree/master/reverse-proxy/>。

2. 查找 app 目录并从该目录构建镜像：

```
docker build -t web-server-app ./path-to-app-directory
```

3. 为 NGINX Plus 构建自定义镜像。在为 NGINX Plus 构建镜像之前，您需要为您的 NGINX Plus 许可获取名为 nginx-repo.key 的密钥和 SSL 证书 nginx-repo.crt。创建目录并将您的 nginx-repo.key 和 nginx-repo.crt 文件存储在其中。

在刚刚创建的目录中，创建以下两个文件：

- 使用以下内容创建示例 Dockerfile。此 docker 文件取自 https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-docker/#docker_plus_image 上提供的示例文件。我们所做的重要更改是我们加载了一个名为 nginx.conf 的单独文件，该文件将在下一步中创建。

```
FROM debian:buster-slim

LABEL maintainer="NGINX Docker Maintainers <docker-maint@nginx.com>"

# Define NGINX versions for NGINX Plus and NGINX Plus modules
# Uncomment this block and the versioned nginxPackages block in the main RUN
# instruction to install a specific release
# ENV NGINX_VERSION 21
# ENV NJS_VERSION 0.3.9
# ENV PKG_RELEASE 1-buster

# Download certificate and key from the customer portal (https://cs.nginx.com
# (https://cs.nginx.com/))
# and copy to the build context
COPY nginx-repo.crt /etc/ssl/nginx/
COPY nginx-repo.key /etc/ssl/nginx/
# COPY nginx.conf /etc/nginx/nginx.conf

RUN set -x \
# Create nginx user/group first, to be consistent throughout Docker variants
&& addgroup --system --gid 101 nginx \
&& adduser --system --disabled-login --ingroup nginx --no-create-home --home /
nonexistent --gecos "nginx user" --shell /bin/false --uid 101 nginx \
&& apt-get update \
&& apt-get install --no-install-recommends --no-install-suggests -y ca-certificates
gnupg1 \
&& \
NGINX_GPGKEY=573BFD6B3D8FBC641079A6ABABF5BD827BD9BF62; \
found=''; \
for server in \
ha.pool.sks-keyservers.net (http://ha.pool.sks-keyservers.net/) \
hkp://keyserver.ubuntu.com:80 \
hkp://p80.pool.sks-keyservers.net:80 \
pgp.mit.edu (http://pgp.mit.edu/) \
; do \
echo "Fetching GPG key $NGINX_GPGKEY from $server"; \
apt-key adv --keyserver "$server" --keyserver-options timeout=10 --recv-keys
"$NGINX_GPGKEY" && found=yes && break; \
done; \
test -z "$found" && echo >&2 "error: failed to fetch GPG key $NGINX_GPGKEY" && exit
1; \
apt-get remove --purge --auto-remove -y gnupg1 && rm -rf /var/lib/apt/lists/* \
# Install the latest release of NGINX Plus and/or NGINX Plus modules
# Uncomment individual modules if necessary
# Use versioned packages over defaults to specify a release
&& nginxPackages=" \
nginx-plus \
# nginx-plus=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-xslt \
```

```
# nginx-plus-module-xslt=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-geoip \
# nginx-plus-module-geoip=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-image-filter \
# nginx-plus-module-image-filter=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-perl \
# nginx-plus-module-perl=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-njs \
# nginx-plus-module-njs=${NGINX_VERSION}+${NJS_VERSION}-${PKG_RELEASE} \
" \
&& echo "Acquire::https::plus-pkgs.nginx.com::Verify-Peer \"true\";" >> /etc/apt/
apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::Verify-Host \"true\";" >> /etc/apt/
apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::SslCert \"/etc/ssl/nginx/nginx-repo.crt
\";" >> /etc/apt/apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::SslKey \"/etc/ssl/nginx/nginx-repo.key
\";" >> /etc/apt/apt.conf.d/90nginx \
&& printf "deb https://plus-pkgs.nginx.com/debian buster nginx-plus\n" > /etc/apt/
sources.list.d/nginx-plus.list \
&& apt-get update \
&& apt-get install --no-install-recommends --no-install-suggests -y \
$nginxPackages \
gettext-base \
curl \
&& apt-get remove --purge --auto-remove -y && rm -rf /var/lib/apt/lists/* /etc/apt/
sources.list.d/nginx-plus.list \
&& rm -rf /etc/apt/apt.conf.d/90nginx /etc/ssl/nginx

# Forward request logs to Docker log collector
RUN ln -sf /dev/stdout /var/log/nginx/access.log \
&& ln -sf /dev/stderr /var/log/nginx/error.log

COPY nginx.conf /etc/nginx/nginx.conf

EXPOSE 80

STOPSIGNAL SIGTERM

CMD ["nginx", "-g", "daemon off;"]
```

- 一个修改自 <https://github.com/awslabs/ecs-nginx-reverse-proxy/tree/master/reverse-proxy/nginx> 的 nginx.conf 文件。

```
events {
    worker_connections 768;
}

http {
    # Nginx will handle gzip compression of responses from the app server
    gzip on;
    gzip_proxied any;
    gzip_types text/plain application/json;
    gzip_min_length 1000;

    upstream backend {
        zone name 10m;
        server app:3000    weight=2;
        server app2:3000   weight=1;
    }

    server{
        listen 8080;
        location /api {
            api write=on;
        }
    }
}
```

```
}  
}  
  
match server_ok {  
    status 100-599;  
}  
  
server {  
    listen 80;  
    status_zone zone;  
    # Nginx will reject anything not matching /api  
    location /api {  
        # Reject requests with unsupported HTTP method  
        if ($request_method !~ ^(GET|POST|HEAD|OPTIONS|PUT|DELETE)$) {  
            return 405;  
        }  
  
        # Only requests matching the whitelist expectations will  
        # get sent to the application server  
        proxy_pass http://backend;  
        health_check uri=/lorem-ipsum match=server_ok;  
        proxy_http_version 1.1;  
        proxy_set_header Upgrade $http_upgrade;  
        proxy_set_header Connection 'upgrade';  
        proxy_set_header Host $host;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        proxy_cache_bypass $http_upgrade;  
    }  
}
```

4. 从新目录中的文件构建镜像：

```
docker build -t nginx-plus-reverse-proxy ./path-to-your-directory
```

5. 将新镜像上传到镜像存储库以供日后使用。

创建任务定义以在 Amazon ECS 中运行 NGINX Plus 和 web 服务器应用程序

接下来，设置任务定义。

此任务定义启用 NGINX Plus Prometheus 指标的收集和导出。NGINX 容器跟踪来自应用程序的输入，并将该数据公开到端口 8080，如 `nginx.conf` 中所设置。NGINX prometheus 导出器抓取这些指标，并将其发布到端口 9113，以在 CloudWatch 中使用。

为 NGINX 示例 Amazon ECS 工作负载设置任务定义

1. 使用以下内容创建任务定义 JSON 文件。将 `your-customized-nginx-plus-image` 替换为自定义 NGINX Plus 镜像的镜像 URI，并将 `your-web-server-app-image` 替换为 Web 服务器应用程序镜像的镜像 URI。

```
{  
  "containerDefinitions": [  
    {  
      "name": "nginx",  
      "image": "your-customized-nginx-plus-image",  
      "memory": 256,  
      "cpu": 256,  
      "essential": true,  
      "portMappings": [  
        {  
          "containerPort": 80,  

```

```
        "protocol": "tcp"
      }
    ],
    "links": [
      "app",
      "app2"
    ]
  },
  {
    "name": "app",
    "image": "your-web-server-app-image",
    "memory": 256,
    "cpu": 128,
    "essential": true
  },
  {
    "name": "app2",
    "image": "your-web-server-app-image",
    "memory": 256,
    "cpu": 128,
    "essential": true
  },
  {
    "name": "nginx-prometheus-exporter",
    "image": "docker.io/nginx/nginx-prometheus-exporter:0.8.0",
    "memory": 256,
    "cpu": 256,
    "essential": true,
    "command": [
      "-nginx.plus",
      "-nginx.scrape-uri",
      "http://nginx:8080/api"
    ]
  },
  "links": [
    "nginx"
  ],
  "portMappings": [
    {
      "containerPort": 9113,
      "protocol": "tcp"
    }
  ]
}
},
"networkMode": "bridge",
"placementConstraints": [],
"family": "nginx-plus-sample-stack"
}
```

2. 注册任务定义：

```
aws ecs register-task-definition --cli-input-json file://path-to-your-task-definition-  
json
```

3. 通过输入以下命令创建服务以运行任务：

```
aws ecs create-service \
  --cluster your-cluster-name \
  --service-name nginx-plus-service \
  --task-definition nginx-plus-sample-stack:1 \
  --desired-count 1
```

确保不要更改服务名称。我们将使用配置来运行 CloudWatch 代理服务，该配置使用启动它们的服务的名称模式来搜索任务。例如，要让 CloudWatch 代理查找此命令启动的任务，您可以将 `sd_service_name_pattern` 的值指定为 `^nginx-plus-service$`。下一部分将提供更多详细信息。

配置 CloudWatch 代理以抓取 NGINX Plus Prometheus 指标

最后一步是配置 CloudWatch 代理以抓取 NGINX 指标。在此示例中，CloudWatch 代理通过服务名称模式和端口 9113 发现任务，导出器在该端口公开 NGINX 的 prometheus 指标。发现任务且指标可用后，CloudWatch 代理开始将收集的指标发布到日志流 `nginx-prometheus-exporter`。

配置 CloudWatch 代理以抓取 NGINX 指标

1. 通过输入以下命令，下载必要 YAML 文件的最新版本。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml
```

2. 使用文本编辑器打开文件，然后在 `resource:CWAgentConfigSSMParameter` 部分的 `value` 密钥中查找完整的 CloudWatch 代理配置。然后，在 `ecs_service_discovery` 部分中，添加以下 `service_name_list_for_tasks` 部分。

```
"service_name_list_for_tasks": [
  {
    "sd_job_name": "nginx-plus-prometheus-exporter",
    "sd_metrics_path": "/metrics",
    "sd_metrics_ports": "9113",
    "sd_service_name_pattern": "^nginx-plus.*"
  }
],
```

3. 在同一个文件中，在 `metric_declaration` 的部分中添加以下部分，以允许 NGINX Plus 指标。请务必遵循现有的缩进模式。

```
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName"]],
  "metric_selectors": [
    "^nginxplus_connections_accepted$",
    "^nginxplus_connections_active$",
    "^nginxplus_connections_dropped$",
    "^nginxplus_connections_idle$",
    "^nginxplus_http_requests_total$",
    "^nginxplus_ssl_handshakes$",
    "^nginxplus_ssl_handshakes_failed$",
    "^nginxplus_up$",
    "^nginxplus_upstream_server_health_checks_fails$"
  ]
},
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName", "upstream"]],
  "metric_selectors": [
    "^nginxplus_upstream_server_response_time$"
  ]
},
{
```

```
"source_labels": ["job"],
"label_matcher": "^nginx-plus.*",
"dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName", "code"]],
"metric_selectors": [
  "^nginxplus_upstream_server_responses$",
  "^nginxplus_server_zone_responses$"
]
},
```

4. 如果您尚未在此集群中部署 CloudWatch 代理，请跳至步骤 8。

如果您已经使用 Amazon CloudFormation 将 CloudWatch 代理部署在 Amazon ECS 集群中，您可以通过输入以下命令来创建更改集：

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
  --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
  --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=${ECS_EXECUTION_ROLE_NAME} \
  --capabilities CAPABILITY_NAMED_IAM \
  --region $AWS_REGION \
  --change-set-name nginx-plus-scraping-support
```

5. 打开 Amazon CloudFormation 控制台，地址：<https://console.aws.amazon.com/cloudformation>。
6. 查看新创建的变更集 nginx-plus-scraping-support。您会看到一项应用于 CWAgentConfigSSMParameter 资源的更改。通过输入以下命令，运行变更集并重新启动 CloudWatch 代理任务：

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-${ECS_NETWORK_MODE} \
--region $AWS_REGION
```

7. 等待大约 10 秒，然后输入以下命令。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-${ECS_NETWORK_MODE} \
--region $AWS_REGION
```

8. 如果您是首次在集群上安装带有 Prometheus 指标收集功能的 CloudWatch 代理，请输入以下命令。

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
```

```
--template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
--parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
               ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
               ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
               ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
               ParameterKey=ExecutionRoleName,ParameterValue=$ECS_EXECUTION_ROLE_NAME \
\
--capabilities CAPABILITY_NAMED_IAM \
--region $AWS_REGION
```

查看您的 NGINX Plus 指标和日志

您现在可以查看正在收集的 NGINX Plus 指标。

查看示例 NGINX 工作负载的指标

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在运行集群的区域中，选择左侧导航窗格中的 Metrics (指标)。查找 ContainerInsights/Prometheus 命名空间以查看指标。
3. 要查看 CloudWatch Logs 事件，请在导航窗格中选择 Log groups (日志组)。事件位于日志流 *nginx-plus-prometheus-exporter* 中的日志组 */aws/containerinsights/**your_cluster_name**/prometheus* 中。

添加新 Prometheus 抓取目标的教程：Amazon ECS 上的 Memcached

本教程提供实践介绍，关于如何在 Amazon ECS 集群上使用 EC2 启动类型抓取示例 Memcached 应用程序的 Prometheus 指标。Memcached Prometheus 导出器目标将由 CloudWatch 代理通过基于 ECS 任务定义的服务发现自动发现。

Memcached 是一个通用型分布式内存缓存系统。它通常用于通过在 RAM 中缓存数据和对象来加速动态数据库驱动的网站，以减少必须读取的外部数据源（例如数据库或 API）的次数。有关更多信息，请参阅[什么是 Memcached？](#)

[memcached_exporter](#) (Apache 许可证 2.0) 是 Prometheus 一款官方导出器。默认情况下，memcache_exporter 在 /metrics. 的端口 0.0.0.0:9150 上提供服务

本教程使用了以下两个 Docker Hub 存储库中的 Docker 镜像：

- [Memcached](#)
- [prom/memcached-exporter](#)

先决条件

要从 Amazon ECS 的示例 Prometheus 工作负载收集指标，您必须在集群中运行 Container Insights。有关安装 Container Insights 的信息，请参阅 [在 Amazon ECS 上设置 Container Insights \(p. 355\)](#)。

主题

- [设置 Amazon ECS EC2 集群环境变量 \(p. 446\)](#)
- [安装 Memcached 示例工作负载 \(p. 446\)](#)
- [配置 CloudWatch 代理以抓取 Memcached Prometheus 指标 \(p. 447\)](#)
- [查看您的 Memcached 指标 \(p. 449\)](#)

设置 Amazon ECS EC2 集群环境变量

设置 Amazon ECS EC2 集群环境变量

1. 请安装 Amazon ECS CLI (如果尚未安装)。有关更多信息, 请参阅[安装 Amazon ECS CLI](#)。
2. 设置新的 Amazon ECS 集群名称和区域。例如 :

```
ECS_CLUSTER_NAME=ecs-ec2-memcached-tutorial
AWS_DEFAULT_REGION=ca-central-1
```

3. (可选) 如果您还没有要在其中安装示例 Memcached 工作负载和 CloudWatch 代理的、带有 EC2 启动类型的 Amazon ECS 集群, 您可以通过输入以下命令来创建一个。

```
ecs-cli up --capability-iam --size 1 \
--instance-type t3.medium \
--cluster $ECS_CLUSTER_NAME \
--region $AWS_REGION
```

此命令的预期结果如下所示 :

```
WARN[0000] You will not be able to SSH into your EC2 instances without a key pair.
INFO[0000] Using recommended Amazon Linux 2 AMI with ECS Agent 1.44.4 and Docker
version 19.03.6-ce
INFO[0001] Created cluster                                cluster=ecs-ec2-memcached-
tutorial region=ca-central-1
INFO[0002] Waiting for your cluster resources to be created...
INFO[0002] Cloudformation stack status                    stackStatus=CREATE_IN_PROGRESS
INFO[0063] Cloudformation stack status                    stackStatus=CREATE_IN_PROGRESS
INFO[0124] Cloudformation stack status                    stackStatus=CREATE_IN_PROGRESS
VPC created: vpc-xxxxxxxxxxxxxxxxxxxx
Security Group created: sg-xxxxxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxxxxx
Cluster creation succeeded.
```

安装 Memcached 示例工作负载

安装公开 Prometheus 指标的示例 Memcached 工作负载

1. 通过输入以下命令, 下载 Memcached Amazon CloudFormation 模板。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/master/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/sample_traffic/memcached/memcached-traffic-sample.yaml
```

2. 通过输入以下命令, 设置要为 Memcached 创建的 IAM 角色名称。

```
MEMCACHED_ECS_TASK_ROLE_NAME=memcached-prometheus-demo-ecs-task-role-name
MEMCACHED_ECS_EXECUTION_ROLE_NAME=memcached-prometheus-demo-ecs-execution-role-name
```

3. 通过输入以下命令, 安装示例 Memcached 工作负载。此示例在 host 网络模式下安装工作负载。

```
MEMCACHED_ECS_NETWORK_MODE=host

aws cloudformation create-stack --stack-name Memcached-Prometheus-Demo-ECS-
$ECS_CLUSTER_NAME-EC2-$MEMCACHED_ECS_NETWORK_MODE \
--template-body file://memcached-traffic-sample.yaml \
--parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
```



```
        ParameterKey=ECSNetworkMode,ParameterValue=$MEMCACHED_ECS_NETWORK_MODE
    \
        ParameterKey=TaskRoleName,ParameterValue=$MEMCACHED_ECS_TASK_ROLE_NAME
    \
        ParameterKey=ExecutionRoleName,ParameterValue=
$MEMCACHED_ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

Amazon CloudFormation 堆栈创建四种资源：

- 一个 ECS 任务角色
- 一个 ECS 任务执行角色
- 一个 Memcached 任务定义
- 一项 Memcached 服务

在 Memcached 任务定义中，定义了两个容器：

- 主容器运行一个简单的 Memcached 应用程序并打开端口 11211 以供访问。
- 另一个容器运行 Redis 导出器进程以在端口 9150 上公开 Prometheus 指标。该容器将由 CloudWatch 代理发现和抓取。

配置 CloudWatch 代理以抓取 Memcached Prometheus 指标

配置 CloudWatch 代理以抓取 Memcached Prometheus 指标

1. 通过输入以下命令，下载 `cwagent-ecs-prometheus-metric-for-awsvpc.yaml` 的最新版本。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/master/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml
```

2. 使用文本编辑器打开该文件，并在 `resource:CWAgentConfigSSMParameter` 部分中的 `value` 密钥后面查找完整的 CloudWatch 代理配置。

然后，在 `ecs_service_discovery` 部分中，将以下配置添加到 `task_definition_list` 部分中。

```
{
  "sd_job_name": "ecs-memcached",
  "sd_metrics_ports": "9150",
  "sd_task_definition_arn_pattern": ".*:task-definition/memcached-prometheus-demo.*:
[0-9]+"
},
```

对于 `metric_declaration` 部分，默认设置不允许任何 Memcached 指标。添加以下部分以允许 Memcached 指标。请务必遵循现有的缩进模式。

```
{
  "source_labels": ["container_name"],
  "label_matcher": "memcached-exporter-.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily"]],
  "metric_selectors": [
    "^memcached_current_(bytes|items|connections)$",
    "^memcached_items_(reclaimed|evicted)_total$",
    "^memcached_(written|read)_bytes_total$",
```

```
    "^memcached_limit_bytes$",
    "^memcached_commands_total$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "memcached-exporter-.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "status", "command"],
  ["ClusterName", "TaskDefinitionFamily", "command"]],
  "metric_selectors": [
    "^memcached_commands_total$"
  ]
}
},
```

3. 如果 Amazon CloudFormation 已在 Amazon ECS 集群中部署 CloudWatch 代理，您可以通过输入以下命令来创建更改集。

```
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
  --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
  --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
    ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
    ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
    ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
    ParameterKey=ExecutionRoleName,ParameterValue=${ECS_EXECUTION_ROLE_NAME} \
  --capabilities CAPABILITY_NAMED_IAM \
  --region $AWS_REGION \
  --change-set-name memcached-scraping-support
```

4. 打开 Amazon CloudFormation 控制台，地址：<https://console.aws.amazon.com/cloudformation>。
5. 查看新创建的变更集 memcached-scraping-support。您会看到一项应用于 CWAgentConfigSSMParameter 资源的更改。通过输入以下命令执行变更集并重新启动 CloudWatch 代理任务。

```
aws ecs update-service --cluster ${ECS_CLUSTER_NAME} \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-${ECS_NETWORK_MODE} \
--region $AWS_REGION
```

6. 等待大约 10 秒，然后输入以下命令。

```
aws ecs update-service --cluster ${ECS_CLUSTER_NAME} \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-${ECS_NETWORK_MODE} \
--region $AWS_REGION
```

7. 如果您是首次为集群安装带有 Prometheus 指标收集功能的 CloudWatch 代理，请输入以下命令：

```
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
  --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
```

```
--parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
               ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
               ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
               ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
               ParameterKey=ExecutionRoleName,ParameterValue=$ECS_EXECUTION_ROLE_NAME \
--capabilities CAPABILITY_NAMED_IAM \
--region $AWS_REGION
```

查看您的 Memcached 指标

本教程会将以下指标发送到 CloudWatch 中的 ECS/ContainerInsights/Prometheus 命名空间。您可以使用 CloudWatch 控制台查看该命名空间中的指标。

指标名称	维度	
memcached_current_items	ClusterName、TaskDefinitionFamily	
memcached_current_connections	ClusterName、TaskDefinitionFamily	
memcached_limit_bytes	ClusterName、TaskDefinitionFamily	
memcached_current_bytes	ClusterName、TaskDefinitionFamily	
memcached_written_bytes	ClusterName、TaskDefinitionFamily	
memcached_read_bytes	ClusterName、TaskDefinitionFamily	
memcached_items_evicted	ClusterName、TaskDefinitionFamily	
memcached_items_received	ClusterName、TaskDefinitionFamily	
memcached_commands_total	ClusterName、TaskDefinitionFamily	
	ClusterName、TaskDefinitionFamily、命令	
	ClusterName、TaskDefinitionFamily、状态、命令	

Note

命令维度的值可以是 delete、get、cas、set、decr、touch、incr 或 flush。
状态维度的值可以是 hit、miss 或 badval。

您还可以为 Memcached Prometheus 指标创建 CloudWatch 控制面板。

为 Memcached Prometheus 指标创建控制面板

1. 创建环境变量，替换以下值以匹配部署。

```
DASHBOARD_NAME=your_memcached_cw_dashboard_name
ECS_TASK_DEF_FAMILY=memcached-prometheus-demo-$ECS_CLUSTER_NAME-EC2-
$MEMCACHED_ECS_NETWORK_MOD
```

2. 输入以下命令以创建控制面板。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/master/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/sample_cloudwatch_dashboards/memcached/cw_dashboard_memcached.json \
```

```
| sed "s/{{YOUR_AWS_REGION}}/$AWS_REGION/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/$ECS_CLUSTER_NAME/g" \
| sed "s/{{YOUR_TASK_DEF_FAMILY}}/$ECS_TASK_DEF_FAMILY/g" \
| xargs -0 aws cloudwatch put-dashboard --dashboard-name ${DASHBOARD_NAME} --region
$AWS_REGION --dashboard-body
```

在 Amazon ECS Fargate 上抓取 Redis Prometheus 指标的教程

本教程提供实践介绍，关于如何在 Amazon ECS Fargate 集群中抓取示例 Redis 应用程序的 Prometheus 指标。Redis Prometheus 导出器目标将由 CloudWatch 代理自动发现，并根据容器的 docker 标签提供 Prometheus 指标支持。

Redis (<https://redis.io/>) 是一个开源（获得 BSD 许可）的内存数据结构存储，用作数据库、缓存和消息代理。有关更多信息，请参阅 [redis](#)。

redis_exporter（MIT 许可证）用于在指定端口（默认值：0.0.0.0:9121）上公开 Redis prometheus 指标。有关更多信息，请参阅 [redis_exporter](#)。

本教程使用了以下两个 Docker Hub 存储库中的 Docker 镜像：

- [redis](#)
- [redis_exporter](#)

先决条件

要从 Amazon ECS 的示例 Prometheus 工作负载收集指标，您必须在集群中运行 Container Insights。有关安装 Container Insights 的信息，请参阅 [在 Amazon ECS 上设置 Container Insights \(p. 355\)](#)。

主题

- [设置 Amazon ECS Fargate 集群环境变量 \(p. 450\)](#)
- [为 Amazon ECS Fargate 集群设置网络环境变量 \(p. 451\)](#)
- [安装 Redis 示例工作负载 \(p. 451\)](#)
- [配置 CloudWatch 代理以抓取 Redis Prometheus 指标 \(p. 452\)](#)
- [查看您的 Redis 指标 \(p. 454\)](#)

设置 Amazon ECS Fargate 集群环境变量

设置 Amazon ECS Fargate 集群环境变量

1. 请安装 Amazon ECS CLI（如果尚未安装）。有关更多信息，请参阅[安装 Amazon ECS CLI](#)。
2. 设置新的 Amazon ECS 集群名称和区域。例如：

```
ECS_CLUSTER_NAME=ecs-fargate-redis-tutorial
AWS_DEFAULT_REGION=ca-central-1
```

3. （可选）如果您还没有要在其中安装示例 Redis 工作负载和 CloudWatch 代理的 Amazon ECS Fargate 集群，您可以通过输入以下命令来创建一个。

```
ecs-cli up --capability-iam \
--cluster $ECS_CLUSTER_NAME \
--launch-type FARGATE \
--region $AWS_DEFAULT_REGION
```

此命令的预期结果如下所示：

```
INFO[0000] Created cluster   cluster=ecs-fargate-redis-tutorial region=ca-central-1
INFO[0001] Waiting for your cluster resources to be created...
INFO[0001] Cloudformation stack status   stackStatus=CREATE_IN_PROGRESS
VPC created: vpc-xxxxxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxxxxx
Cluster creation succeeded.
```

为 Amazon ECS Fargate 集群设置网络环境变量

为 Amazon ECS Fargate 集群设置网络环境变量

1. 设置 Amazon ECS 集群的 VPC 和子网 ID。如果您在上一步骤中创建了新集群，您将在最终命令的结果中看到这些值。否则，请使用要与 Redis 一起使用的现有集群的 ID。

```
ECS_CLUSTER_VPC=vpc-xxxxxxxxxxxxxxxxxxxx
ECS_CLUSTER_SUBNET_1=subnet-xxxxxxxxxxxxxxxxxxxx
ECS_CLUSTER_SUBNET_2=subnet-xxxxxxxxxxxxxxxxxxxx
```

2. 在本教程中，我们将在 Amazon ECS 集群的 VPC 的默认安全组中安装 Redis 应用程序和 CloudWatch 代理。默认安全组允许同一安全组内的所有网络连接，因此 CloudWatch 代理可以抓取 Redis 容器上公开的 Prometheus 指标。在实际生产环境中，您可能希望为 Redis 应用程序和 CloudWatch 代理创建专用安全组并为其设置自定义权限。

要获取默认安全组 ID，请输入以下命令。

```
aws ec2 describe-security-groups \
--filters Name=vpc-id,Values=$ECS_CLUSTER_VPC \
--region $AWS_DEFAULT_REGION
```

然后通过输入以下命令设置 Fargate 集群默认安全组变量，将 *my-default-security-group* 替换为您在上一个命令中找到的值。

```
ECS_CLUSTER_SECURITY_GROUP=my-default-security-group
```

安装 Redis 示例工作负载

安装公开 Prometheus 指标的示例 Redis 工作负载

1. 通过输入以下命令，下载 Redis Amazon CloudFormation 模板。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/master/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/sample_traffic/redis/redis-traffic-sample.yaml
```

2. 通过输入以下命令，设置要为 Redis 创建的 IAM 角色名称。

```
REDIS_ECS_TASK_ROLE_NAME=redis-prometheus-demo-ecs-task-role-name
REDIS_ECS_EXECUTION_ROLE_NAME=redis-prometheus-demo-ecs-execution-role-name
```

3. 通过输入以下命令安装 Redis 示例工作负载。

```
aws cloudformation create-stack --stack-name Redis-Prometheus-Demo-ECS-
$ECS_CLUSTER_NAME-fargate-awsvpc \
```

```
--template-body file://redis-traffic-sample.yaml \
--parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
               ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
               ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET_1 \
               ParameterKey=TaskRoleName,ParameterValue=$REDIS_ECS_TASK_ROLE_NAME \
               ParameterKey=ExecutionRoleName,ParameterValue=
$REDIS_ECS_EXECUTION_ROLE_NAME \
--capabilities CAPABILITY_NAMED_IAM \
--region $AWS_DEFAULT_REGION
```

Amazon CloudFormation 堆栈创建四种资源：

- 一个 ECS 任务角色
- 一个 ECS 任务执行角色
- 一个 Redis 任务定义
- 一项 Redis 服务

在 Redis 任务定义中，定义了两个容器：

- 主容器运行一个简单的 Redis 应用程序并打开端口 6379 以供访问。
- 另一个容器运行 Redis 导出器进程以在端口 9121 上公开 Prometheus 指标。该容器将由 CloudWatch 代理发现和抓取。定义了以下 docker 标签，以便 CloudWatch 代理可以根据它发现该容器。

```
ECS_PROMETHEUS_EXPORTER_PORT: 9121
```

配置 CloudWatch 代理以抓取 Redis Prometheus 指标

配置 CloudWatch 代理以抓取 Redis Prometheus 指标

1. 通过输入以下命令，下载 `cwagent-ecs-prometheus-metric-for-awsvpc.yaml` 的最新版本。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/master/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml
```

2. 使用文本编辑器打开该文件，并在 `resource:CWAgentConfigSSMParameter` 部分中的 `value` 密钥后面查找完整的 CloudWatch 代理配置。

然后，在此处显示的 `ecs_service_discovery` 部分中，使用基于 `ECS_PROMETHEUS_EXPORTER_PORT` 的默认设置启用基于 `docker_label` 的服务发现，这与我们在 Redis ECS 任务定义中定义的 Docker 标签相匹配。因此，我们无需在此部分进行任何更改：

```
ecs_service_discovery": {
  "sd_frequency": "1m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  * "docker_label": {
    },*
  ...
}
```

对于 `metric_declaration` 部分，默认设置不允许任何 Redis 指标。添加以下部分以允许 Redis 指标。请务必遵循现有的缩进模式。

```
{
  "source_labels": ["container_name"],
```

```
"label_matcher": "^redis-exporter-.*$",
"dimensions": [["ClusterName", "TaskDefinitionFamily"]],
"metric_selectors": [
  "^redis_net_(in|out)put_bytes_total$",
  "^redis_(expired|evicted)_keys_total$",
  "^redis_keyspace_(hits|misses)_total$",
  "^redis_memory_used_bytes$",
  "^redis_connected_clients$"
]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "cmd"]],
  "metric_selectors": [
    "^redis_commands_total$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "db"]],
  "metric_selectors": [
    "^redis_db_keys$"
  ]
},
},
```

3. 如果 Amazon CloudFormation 已在 Amazon ECS 集群中部署 CloudWatch 代理，您可以通过输入以下命令来创建更改集。

```
ECS_LAUNCH_TYPE=FARGATE
CREATE_IAM_ROLES=True
ECS_CLUSTER_SUBNET=$ECS_CLUSTER_SUBNET_1
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
$ECS_CLUSTER_NAME-$ECS_LAUNCH_TYPE-awsvpc \
  --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
  --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
    ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
    ParameterKey=ECSLaunchType,ParameterValue=$ECS_LAUNCH_TYPE \
    ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
    ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET \
    ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
    ParameterKey=ExecutionRoleName,ParameterValue=$ECS_EXECUTION_ROLE_NAME
\
  --capabilities CAPABILITY_NAMED_IAM \
  --region ${AWS_DEFAULT_REGION} \
  --change-set-name redis-scraping-support
```

4. 打开 Amazon CloudFormation 控制台，地址：<https://console.aws.amazon.com/cloudformation>。
5. 查看新创建的变更集 redis-scraping-support。您会看到一项应用于 CWAgentConfigSSMPParameter 资源的更改。通过输入以下命令执行变更集并重新启动 CloudWatch 代理任务。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
  --desired-count 0 \
  --service cwagent-prometheus-replica-service-$ECS_LAUNCH_TYPE-awsvpc \
  --region ${AWS_DEFAULT_REGION}
```

6. 等待大约 10 秒，然后输入以下命令。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-$ECS_LAUNCH_TYPE-awsvpc \
--region ${AWS_DEFAULT_REGION}
```

7. 如果您是首次为集群安装带有 Prometheus 指标收集功能的 CloudWatch 代理，请输入以下命令：

```
ECS_LAUNCH_TYPE=FARGATE
CREATE_IAM_ROLES=True
ECS_CLUSTER_SUBNET=$ECS_CLUSTER_SUBNET_1
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-$ECS_CLUSTER_NAME-$ECS_LAUNCH_TYPE-awsvpc \
--template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
--parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
               ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
               ParameterKey=ECSLaunchType,ParameterValue=$ECS_LAUNCH_TYPE \
               ParameterKey=SecurityGroupID,ParameterValue=$ECS_CLUSTER_SECURITY_GROUP \
               ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET \
               ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
               ParameterKey=ExecutionRoleName,ParameterValue=$ECS_EXECUTION_ROLE_NAME \
--capabilities CAPABILITY_NAMED_IAM \
--region ${AWS_DEFAULT_REGION}
```

查看您的 Redis 指标

本教程会将以下指标发送到 CloudWatch 中的 ECS/ContainerInsights/Prometheus 命名空间。您可以使用 CloudWatch 控制台查看该命名空间中的指标。

指标名称	维度	
redis_net_input_bytes	ClusterName、TaskDefinitionFamily	
redis_net_output_bytes	ClusterName、TaskDefinitionFamily	
redis_expired_keys_total	ClusterName、TaskDefinitionFamily	
redis_evicted_keys_total	ClusterName、TaskDefinitionFamily	
redis_keyspace_hits	ClusterName、TaskDefinitionFamily	
redis_keyspace_misses	ClusterName、TaskDefinitionFamily	
redis_memory_used_bytes	ClusterName、TaskDefinitionFamily	
redis_connected_clients	ClusterName、TaskDefinitionFamily	
redis_commands_total	ClusterName、TaskDefinitionFamily、cmd	
redis_db_keys	ClusterName、TaskDefinitionFamily、db	

db 维度的值可以从 db0 到 db15。

为 Redis Prometheus 指标创建控制面板

- ```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/master/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_cloudwatch_dashboards/redis/cw_dashboard_redis.json \
| sed "s/{{YOUR_AWS_REGION}}/${REGION_NAME}/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/${CLUSTER_NAME}/g" \
| sed "s/{{YOUR_NAMESPACE}}/${NAMESPACE}/g" \
```

## 455

- HAProxy
- Fluent Bit

您还可以将代理配置为抓取和导入其他 Prometheus 工作负载和源。

在按照下面这些步骤安装 CloudWatch 代理来收集 Prometheus 指标之前，您必须有在 Amazon EKS 上运行的集群或者在 Amazon EC2 实例上运行的 Kubernetes 集群。

#### VPC 安全组要求

Prometheus 工作负载的安全组的入口规则必须向 CloudWatch 代理打开 Prometheus 端口，以便通过私有 IP 抓取 Prometheus 指标。

CloudWatch 代理的安全组的出口规则必须允许 CloudWatch 代理通过私有 IP 连接到 Prometheus 工作负载的端口。

#### 主题

- [在 Amazon EKS 和 Kubernetes 集群上安装带有 Prometheus 指标收集功能的 CloudWatch 代理 \(p. 456\)](#)
- [抓取其他 Prometheus 源并导入这些指标 \(p. 459\)](#)
- [\( 可选 \) 为 Prometheus 指标测试设置示例容器化 Amazon EKS 工作负载 \(p. 466\)](#)

### 在 Amazon EKS 和 Kubernetes 集群上安装带有 Prometheus 指标收集功能的 CloudWatch 代理

本节介绍如何在运行 Amazon EKS 或 Kubernetes 的集群中设置带有 Prometheus 监控功能的 CloudWatch 代理。执行此操作后，代理会自动抓取并导入该集群中运行的以下工作负载的指标。

- Amazon App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy
- Fluent Bit

您还可以将代理配置为抓取和导入其他 Prometheus 工作负载和源。

在按照下面这些步骤安装 CloudWatch 代理来收集 Prometheus 指标之前，您必须有在 Amazon EKS 上运行的集群或者在 Amazon EC2 实例上运行的 Kubernetes 集群。

#### VPC 安全组要求

Prometheus 工作负载的安全组的入口规则必须向 CloudWatch 代理打开 Prometheus 端口，以便通过私有 IP 抓取 Prometheus 指标。

CloudWatch 代理的安全组的出口规则必须允许 CloudWatch 代理通过私有 IP 连接到 Prometheus 工作负载的端口。

#### 主题

- [设置 IAM 角色 \(p. 456\)](#)
- [安装 CloudWatch 代理以收集 Prometheus 指标 \(p. 457\)](#)

### 设置 IAM 角色

第一步是在集群中设置必要的 IAM 角色。有两种方法：

- 为服务账户设置 IAM 角色，也称为服务角色。此方法适用于 EC2 启动类型和 Fargate 启动类型。
- 将 IAM policy 添加到集群的 IAM 角色。这仅适用于 EC2 启动类型。

设置服务角色 ( EC2 启动类型和 Fargate 启动类型 )

要设置服务角色，请输入以下命令。将 *MyCluster* 替换为集群的名称。

```
eksctl create iamserviceaccount \
 --name cwagent-prometheus \
 --namespace amazon-cloudwatch \
 --cluster MyCluster \
 --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
 --approve \
 --override-existing-serviceaccounts
```

向集群的 IAM 角色添加策略 ( 仅限 EC2 启动类型 )

在集群中为 Prometheus 支持设置 IAM policy

1. 通过以下网址打开 Amazon EC2 控制台：<https://console.aws.amazon.com/ec2/>。
2. 在导航窗格中，选择实例。
3. 您需要查找集群的 IAM 角色名称的前缀。为此，请选中集群中实例名称旁边的复选框，然后选择操作、实例设置、附加/替换 IAM 角色。然后复制 IAM 角色的前缀，例如 eksctl-dev303-workshop-nodegroup。
4. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
5. 在导航窗格中，选择 Roles ( 角色 )。
6. 使用搜索框查找您在此过程前面复制的前缀，然后选择该角色。
7. 选择 Attach policies ( 附上策略 )。
8. 使用搜索框查找 CloudWatchAgentServerPolicy。选中 CloudWatchAgentServerPolicy 旁边的复选框，然后选择附加策略。

## 安装 CloudWatch 代理以收集 Prometheus 指标

您必须在集群中安装 CloudWatch 代理才能收集指标。对于 Amazon EKS 集群和 Kubernetes 集群，安装代理的方式有所不同。

删除具有 Prometheus 支持的 CloudWatch 代理的早期版本

如果您的集群中已安装了具有 Prometheus 支持的 CloudWatch 代理版本，则必须通过输入以下命令删除该版本。这仅对具有 Prometheus 支持的以前版本的代理是必要的。您无需删除启用了 Container Insights 但没有 Prometheus 支持的 CloudWatch 代理。

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

## 在具有 EC2 启动类型的 Amazon EKS 集群上安装 CloudWatch 代理

要在 Amazon EKS 集群上安装具有 Prometheus 支持的 CloudWatch 代理，请按照下列步骤操作。

在 Amazon EKS 集群上安装具有 Prometheus 支持的 CloudWatch 代理

1. 输入以下命令，检查是否已创建 amazon-cloudwatch 命名空间：

```
kubectl get namespace
```

2. 如果结果中未显示 amazon-cloudwatch，请输入以下命令来创建它：

```
kubectl create namespace amazon-cloudwatch
```

3. 要使用默认配置部署代理并使其将数据发送到所在的 Amazon 区域，请输入以下命令：

**Note**

默认情况下，以下设置步骤以匿名用户身份从 Docker Hub 拉取容器镜像。这种拉取可能会受到速率限制。有关更多信息，请参阅[CloudWatch 代理容器镜像 \(p. 354\)](#)。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
```

要让代理将数据发送到不同的区域，请按照下列步骤操作：

- a. 输入以下命令，下载代理的 YAML 文件：

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
```

- b. 使用文本编辑器打开文件，然后搜索文件的 `cwagentconfig.json` 块。
- c. 添加突出显示的行，并指定所需的区域：

```
cwagentconfig.json: |
{
 "agent": {
 "region": "us-east-2"
 },
 "logs": { ...
```

- d. 保存文件并使用更新后的文件部署代理。

```
kubectl apply -f prometheus-eks.yaml
```

## 在具有 Fargate 启动类型的 Amazon EKS 集群上安装 CloudWatch 代理

要在具有 Fargate 启动类型的 Amazon EKS 集群上安装具有 Prometheus 支持的 CloudWatch 代理，请按照下列步骤操作。

在具有 Fargate 启动类型的 Amazon EKS 集群上安装具有 Prometheus 支持的 CloudWatch 代理

1. 输入以下命令为 CloudWatch 代理创建 Fargate 配置文件，以便它可以在集群内运行。将 `MyCluster` 替换为集群的名称。

```
eksctl create fargateprofile --cluster MyCluster \
--name amazon-cloudwatch \
--namespace amazon-cloudwatch
```

2. 要安装 CloudWatch 代理，请输入以下命令。将 `MyCluster` 替换为集群的名称。此名称在存储代理收集的日志事件的日志组名称中使用，也用作代理收集的指标的维度。

将 `region####` 替换为要将指标发送到的区域的名称。例如：`us-west-1`。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml |
```

```
sed "s/{{cluster_name}}/MyCluster;/s/{{region_name}}/region/" |
kubectl apply -f -
```

## 在 Kubernetes 集群上安装 CloudWatch 代理

要在运行 Kubernetes 的集群上安装具有 Prometheus 支持的 CloudWatch 代理，请输入以下命令：

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/
latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/
prometheus-k8s.yaml |
sed "s/{{cluster_name}}/MyCluster;/s/{{region_name}}/region/" |
kubectl apply -f -
```

将 **MyCluster** 替换为集群的名称。此名称在存储代理收集的日志事件的日志组名称中使用，也用作代理收集的指标的维度。

将 **region####** 替换为要将指标发送到的 Amazon 区域的名称。例如：**us-west-1**。

## 验证代理是否正在运行

在 Amazon EKS 和 Kubernetes 集群上，您可以输入以下命令以确认代理正在运行。

```
kubectl get pod -l "app=cwagent-prometheus" -n amazon-cloudwatch
```

如果结果包含处于 Running 状态的单个 CloudWatch 代理 pod，则代理会运行并收集 Prometheus 指标。默认情况下，CloudWatch 代理每分钟会收集 App Mesh、NGINX、Memcached、Java/JMX 和 HAProxy 的指标。有关这些指标的更多信息，请参阅 [CloudWatch 代理收集的 Prometheus 指标 \(p. 480\)](#)。有关如何在 CloudWatch 中查看 Prometheus 指标的说明，请参阅 [查看 Prometheus 指标 \(p. 487\)](#)

您还可以将 CloudWatch 代理配置为从其他 Prometheus 导出程序收集指标。有关更多信息，请参阅 [抓取其他 Prometheus 源并导入这些指标 \(p. 459\)](#)。

## 抓取其他 Prometheus 源并导入这些指标

具有 Prometheus 监控功能的 CloudWatch 代理需要两种配置来抓取 Prometheus 指标。一个用于 Prometheus 文档中 [<scrape\\_config>](#) 记录的标准 Prometheus 配置。另一种配置是 CloudWatch 代理配置文件。

对于 Amazon EKS 集群，配置在 prometheus-eks.yaml（适用于 EC2 启动类型）或 prometheus-eks-fargate.yaml（适用于 Fargate 启动类型）中定义为两个 Config 映射：

- name: prometheus-config 部分包含 Prometheus 抓取的设置。
- name: prometheus-cwagentconfig 部分包含 CloudWatch 代理的配置。您可以使用此部分配置 CloudWatch 如何收集 Prometheus 指标。例如，您指定要导入 CloudWatch 的指标，并定义其维度。

对于在 Amazon EC2 实例上运行的 Kubernetes 集群，配置在 prometheus-k8s.yaml YAML 文件中定义为两个 config 映射：

- name: prometheus-config 部分包含 Prometheus 抓取的设置。
- name: prometheus-cwagentconfig 部分包含 CloudWatch 代理的配置。

要抓取其他 Prometheus 指标源并将这些指标导入 CloudWatch，您需要修改 Prometheus 抓取配置和 CloudWatch 代理配置，然后使用更新的配置重新部署代理。

## VPC 安全组要求

Prometheus 工作负载的安全组的入口规则必须向 CloudWatch 代理打开 Prometheus 端口，以便通过私有 IP 抓取 Prometheus 指标。

CloudWatch 代理的安全组的出口规则必须允许 CloudWatch 代理通过私有 IP 连接到 Prometheus 工作负载的端口。

### Prometheus 抓取配置

CloudWatch 代理支持标准的 Prometheus 抓取配置，如 Prometheus 文档中的 [<scrape\\_config>](#) 所述。您可以编辑此部分以更新此文件中已有的配置，并添加其他 Prometheus 抓取目标。默认情况下，示例配置文件包含以下全局配置行：

```
global:
 scrape_interval: 1m
 scrape_timeout: 10s
```

- `scrape_interval` – 定义抓取目标的频率。
- `scrape_timeout` – 定义抓取请求超时之前的等待时间。

您还可以在作业级别为这些设置定义不同的值，以覆盖全局配置。

### Prometheus 抓取任务

CloudWatch 代理 YAML 文件已配置了一些默认的抓取任务。例如在 `prometheus-eks.yaml` 中，在 `scrape_configs` 部分中的 `job_name` 行中配置了默认的抓取任务。在此文件中，以下默认 `kubernetes-pod-jmx` 部分会抓取 JMX Exporter 指标。

```
- job_name: 'kubernetes-pod-jmx'
 sample_limit: 10000
 metrics_path: /metrics
 kubernetes_sd_configs:
 - role: pod
 relabel_configs:
 - source_labels: [__address__]
 action: keep
 regex: '.*:9404$'
 - action: labelmap
 regex: __meta_kubernetes_pod_label_(.+)
 - action: replace
 source_labels:
 - __meta_kubernetes_namespace
 target_label: Namespace
 - source_labels: [__meta_kubernetes_pod_name]
 action: replace
 target_label: pod_name
 - action: replace
 source_labels:
 - __meta_kubernetes_pod_container_name
 target_label: container_name
 - action: replace
 source_labels:
 - __meta_kubernetes_pod_controller_name
 target_label: pod_controller_name
 - action: replace
 source_labels:
 - __meta_kubernetes_pod_controller_kind
 target_label: pod_controller_kind
 - action: replace
 source_labels:
 - __meta_kubernetes_pod_phase
```



```
target_label: pod_phase
```

这些默认目标中的每个目标都会被抓取，并使用嵌入式指标格式在日志事件中将指标发送到 CloudWatch。有关更多信息，请参阅[采用 CloudWatch 嵌入式指标格式摄取高基数日志以生成指标 \(p. 754\)](#)。

来自 Amazon EKS 和 Kubernetes 集群的日志事件存储在 CloudWatch Logs 的 `/aws/containerinsights/cluster_name/prometheus` 日志组中。来自 Amazon ECS 集群的日志事件存储在 `/aws/ecs/containerinsights/cluster_name/prometheus` 日志组中。

每个抓取作业都包含在此日志组中的不同日志流中。例如，为 App Mesh 定义了 Prometheus 抓取任务 `kubernetes-pod-appmesh-envoy`。所有来自 Amazon EKS 和 Kubernetes 集群的 App Mesh Prometheus 指标都发送到名为 `/aws/containerinsights/cluster_name>prometheus/kubernetes-pod-appmesh-envoy/` 的日志流。

要添加新的抓取目标，请将新的 `job_name` 部分添加到 YAML 文件的 `scrape_configs` 部分，然后重新启动代理。有关此过程的示例，请参阅[添加新 Prometheus 抓取目标的教程：Prometheus API 服务器指标 \(p. 463\)](#)。

## Prometheus 的 CloudWatch 代理配置

CloudWatch 代理配置文件在 `metrics_collected` 下面有一个 `prometheus` 部分用于 Prometheus 抓取配置。它包含以下配置选项：

- `cluster_name` – 指定要在日志事件中添加为标签的集群名称。此字段为可选项。如果省略它，代理可以检测到 Amazon EKS 或 Kubernetes 集群名称。
- `log_group_name` – 指定已抓取 Prometheus 指标的日志组名称。此字段为可选项。如果省略它，CloudWatch 会将 `/aws/containerinsights/cluster_name/prometheus` 用于来自 Amazon EKS 和 Kubernetes 集群的日志。
- `prometheus_config_path` – 指定 Prometheus 抓取配置文件路径。如果此字段的值以 `env:` 开头，Prometheus 抓取配置文件内容将从容器的环境变量中检索。请勿更改此字段。
- `ecs_service_discovery` – 是指定 Amazon ECS Prometheus 服务发现配置的部分。有关更多信息，请参阅[Amazon ECS 集群上自动发现的详细指南 \(p. 426\)](#)。

`ecs_service_discovery` 部分包含以下字段：

- `sd_frequency` 是发现 Prometheus 导出器的频率。指定数字和单位后缀。例如，`1m` 表示每分钟一次或 `30s` 表示每 30 秒一次。有效的单位后缀为 `ns`、`us`、`ms`、`s`、`m` 和 `h`。

此字段为可选项。默认值为 60 秒（1 分钟）。

- `sd_target_cluster` 是用于自动发现的目标 Amazon ECS 集群名称。此字段为可选项。默认名称为安装 CloudWatch 代理的 Amazon ECS 集群的名称。
- `sd_cluster_region` 是目标 Amazon ECS 集群的区域。此字段为可选项。默认区域为安装 CloudWatch 代理的 Amazon ECS 集群的区域。
- `sd_result_file` 是 Prometheus 目标结果的 YAML 文件的路径。Prometheus 抓取配置将引用此文件。
- `docker_label` 是可选部分，您可以使用它来指定基于 docker 标签的服务发现的配置。如果省略此部分，则不会使用基于 docker 标签的发现。此部分包含以下字段：
  - `sd_port_label` 是容器的 docker 标签名称，用于指定 Prometheus 指标的容器端口。默认值为 `ECS_PROMETHEUS_EXPORTER_PORT`。如果容器没有此 docker 标签，CloudWatch 代理将跳过它。
  - `sd_metrics_path_label` 是容器的 docker 标签名称，用于指定 Prometheus 指标路径。默认值为 `ECS_PROMETHEUS_METRICS_PATH`。如果容器没有此 docker 标签，代理会采用默认路径 `/metrics`。
  - `sd_job_name_label` 是容器的 docker 标签名称，用于指定 Prometheus 抓取任务名称。默认值为 `job`。如果容器没有此 docker 标签，CloudWatch 代理会使用 Prometheus 抓取配置中的任务名称。
- `task_definition_list` 是可选部分，可用于指定基于任务定义的服务发现的配置。如果省略此部分，则不会使用基于任务定义的发现。此部分包含以下字段：

- `sd_task_definition_arn_pattern` 是用于指定要发现的 Amazon ECS 任务定义的模式。这是正则表达式。
- `sd_metrics_ports` 列出 Prometheus 指标的 `containerPort`。用分号分隔 `containerPorts`。
- `sd_container_name_pattern` 指定 Amazon ECS 任务容器名称。这是正则表达式。
- `sd_metrics_path` 指定 Prometheus 指标路径。如果省略此项，代理会采用默认路径 `/metrics`。
- `sd_job_name` 指定 Prometheus 抓取任务名称。如果省略此字段，CloudWatch 代理会使用 Prometheus 抓取配置中的任务名称。
- `metric_declaration` – 是指定要生成的采用嵌入式指标格式的日志数组的部分。默认情况下，CloudWatch 代理从中进行导入的每个 Prometheus 源都有 `metric_declaration` 部分。这些部分各包括以下字段：
  - `label_matcher` 是一个正则表达式，用于检查 `source_labels` 中列出的标签的值。匹配的指标将启用，以包含在发送到 CloudWatch 的嵌入式指标格式中。

如果您在 `source_labels` 中指定了多个标签，我们建议您不要在 `label_matcher` 的正则表达式中使用 `^` 或 `$` 字符。

- `source_labels` 指定由 `label_matcher` 行检查的标签的值。
- `label_separator` 指定要在 `label_matcher` 行中使用的分隔符（如果指定了多个 `source_labels`）。默认为 `;`。您可以在以下示例中看到 `label_matcher` 行中使用的此默认值。
- `metric_selectors` 是一个正则表达式，用于指定要收集并发送到 CloudWatch 的指标。
- `dimensions` 是要用作每个选定指标的 CloudWatch 维度的标签列表。

请参阅以下 `metric_declaration` 示例。

```
"metric_declaration": [
 {
 "source_labels": ["Service", "Namespace"],
 "label_matcher": "(.*node-exporter.*|.*kube-dns.*);kube-system",
 "dimensions": [
 ["Service", "Namespace"]
],
 "metric_selectors": [
 "^coredns_dns_request_type_count_total$"
]
 }
]
```

此示例配置嵌入式指标格式部分，以便在满足以下条件时作为日志事件发送：

- `Service` 的值包含 `node-exporter` 或 `kube-dns`。
- `Namespace` 的值为 `kube-system`。
- Prometheus 指标 `coredns_dns_request_type_count_total` 同时包含 `Service` 和 `Namespace` 标签。

发送的日志事件包括以下突出显示的部分：

```
{
 "CloudWatchMetrics": [
 {
 "Metrics": [
 {
 "Name": "coredns_dns_request_type_count_total"
 }
],
 "Dimensions": [
 [
 "Namespace",
```



```
 "Service"
],
 "Namespace": "ContainerInsights/Prometheus"
 }
],
 "Namespace": "kube-system",
 "Service": "kube-dns",
 "coredns_dns_request_type_count_total": 2562,
 "eks_amazonaws_com_component": "kube-dns",
 "instance": "192.168.61.254:9153",
 "job": "kubernetes-service-endpoints",
 ...
}
```

### 添加新 Prometheus 抓取目标的教程：Prometheus API 服务器指标

默认情况下，Kubernetes API 服务器会在端点上公开 Prometheus 指标。Kubernetes API 服务器抓取配置的官方示例可在 [Github](#) 上找到。

以下教程演示如何执行以下步骤以开始将 Kubernetes API 服务器指标导入到 CloudWatch 中：

- 将 Kubernetes API 服务器的 Prometheus 抓取配置添加到 CloudWatch 代理 YAML 文件。
- 在 CloudWatch 代理 YAML 文件中配置嵌入式指标格式指标定义。
- ( 可选 ) 为 Kubernetes API 服务器指标创建 CloudWatch 控制面板。

#### Note

Kubernetes API 服务器公开计量表、计数器、直方图和摘要指标。在此版本的 Prometheus 指标支持中，CloudWatch 仅导入具有计量表、计数器和汇总类型的指标。

开始在 CloudWatch 中收集 Kubernetes API 服务器 Prometheus 指标

1. 通过输入以下命令之一，下载 `prometheus-eks.yaml`、`prometheus-eks-fargate.yaml` 或 `prometheus-k8s.yaml` 文件的最新版本。

对于具有 EC2 启动类型的 Amazon EKS 集群，请输入以下命令：

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
```

对于具有 Fargate 启动类型的 Amazon EKS 集群，请输入以下命令：

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml
```

对于在 Amazon EC2 实例上运行的 Kubernetes 集群，请输入以下命令：

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml
```

2. 使用文本编辑器打开文件，找到 `prometheus-config` 部分，然后在该部分中添加以下部分。然后，保存更改：

```
Scrape config for API servers
```

```
- job_name: 'kubernetes-apiservers'
 kubernetes_sd_configs:
 - role: endpoints
 namespaces:
 names:
 - default
 scheme: https
 tls_config:
 ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
 insecure_skip_verify: true
 bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
 relabel_configs:
 - source_labels: [__meta_kubernetes_service_name,
__meta_kubernetes_endpoint_port_name]
 action: keep
 regex: kubernetes;https
 - action: replace
 source_labels:
 - __meta_kubernetes_namespace
 target_label: Namespace
 - action: replace
 source_labels:
 - __meta_kubernetes_service_name
 target_label: Service
```

3. 当 YAML 文件仍在文本编辑器中处于打开状态时，找到 `cwagentconfig.json` 部分。添加以下子部分并保存更改。此部分将 API 服务器指标放到 CloudWatch 代理允许列表中。将三种类型的 API 服务器指标添加到允许列表中：

- etcd 对象计数
- API 服务器注册控制器指标
- API 服务器请求指标

```
{ "source_labels": ["job", "resource"],
 "label_matcher": "^kubernetes-apiservers;(services|daemonsets.apps|deployments.apps|
configmaps|endpoints|secrets|serviceaccounts|replicasets.apps)",
 "dimensions": [["ClusterName", "Service", "resource"]],
 "metric_selectors": [
 "^etcd_object_counts$"
]
},
{ "source_labels": ["job", "name"],
 "label_matcher": "^kubernetes-apiservers;APIServiceRegistrationController$",
 "dimensions": [["ClusterName", "Service", "name"]],
 "metric_selectors": [
 "^workqueue_depth$",
 "^workqueue_adds_total$",
 "^workqueue_retries_total$"
]
},
{ "source_labels": ["job", "code"],
 "label_matcher": "^kubernetes-apiservers;2[0-9]{2}$",
 "dimensions": [["ClusterName", "Service", "code"]],
 "metric_selectors": [
 "^apiserver_request_total$"
]
},
{ "source_labels": ["job"],
 "label_matcher": "^kubernetes-apiservers",
 "dimensions": [["ClusterName", "Service"]],
 "metric_selectors": [
 "^apiserver_request_total$"
]
}
```

```
},
```

4. 如果您已在集群中部署了具有 Prometheus 支持的 CloudWatch 代理，则必须通过输入以下命令将其删除：

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

5. 通过输入以下任一命令，使用更新的配置部署 CloudWatch 代理。对于具有 EC2 启动类型的 Amazon EKS 集群，请输入：

```
kubectl apply -f prometheus-eks.yaml
```

对于具有 Fargate 启动类型的 Amazon EKS 集群，请输入以下命令。将 **MyCluster** 和 **region##** 替换为值以匹配您的部署。

```
cat prometheus-eks-fargate.yaml \
| sed "s/{{cluster_name}}/MyCluster/s/{{region_name}}/region/" \
| kubectl apply -f -
```

对于 Kubernetes 集群，请输入以下命令。将 **MyCluster** 和 **region####** 替换为值以匹配您的部署。

```
cat prometheus-k8s.yaml \
| sed "s/{{cluster_name}}/MyCluster/s/{{region_name}}/region/" \
| kubectl apply -f -
```

完成此操作后，您应该会看到 `/aws/containerinsights/cluster_name/prometheus` 日志组中名为 `kubernetes-apiservers` 的新日志流。此日志流应使用嵌入式指标格式定义来嵌入日志事件，如下所示：

```
{
 "CloudWatchMetrics": [
 {
 "Metrics": [
 {
 "Name": "apiserver_request_total"
 }
],
 "Dimensions": [
 {
 "ClusterName",
 "Service"
 }
],
 "Namespace": "ContainerInsights/Prometheus"
 }
],
 "ClusterName": "my-cluster-name",
 "Namespace": "default",
 "Service": "kubernetes",
 "Timestamp": "1592267020339",
 "Version": "0",
 "apiserver_request_count": 0,
 "apiserver_request_total": 0,
 "code": "0",
 "component": "apiserver",
 "contentType": "application/json",
 "instance": "192.0.2.0:443",
 "job": "kubernetes-apiservers",
 "prom_metric_type": "counter",
 "resource": "pods",
}
```

```
"scope": "namespace",
"verb": "WATCH",
"version": "v1"
}
```

您可以在 ContainerInsights/Prometheus 命名空间中的 CloudWatch 控制台中查看您的指标。(可选)还可以为 Prometheus Kubernetes API 服务器指标创建 CloudWatch 控制面板。

#### (可选) 为 Kubernetes API 服务器指标创建控制面板

要在控制面板中查看 Kubernetes API 服务器指标，您必须先完成前面几个部分中的步骤，然后才能开始在 CloudWatch 中收集这些指标。

##### 为 Kubernetes API 服务器指标创建控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 确保选择了正确的 Amazon 区域。
3. 在导航窗格中，选择 Dashboards (控制面板)。
4. 选择创建控制面板。输入新控制面板的名称，然后选择创建控制面板。
5. 在添加到该控制面板中，选择取消。
6. 依次选择操作、编辑控制面板。
7. 下载以下 JSON 文件：[Kubernetes API 控制面板源](#)。
8. 使用文本编辑器打开下载的 JSON 文件，然后执行以下更改：
  - 将所有 `{{YOUR_CLUSTER_NAME}}` 字符串替换为您的集群的确切名称。请勿在文本之前或之后添加空格。
  - 将所有 `{{YOUR_AWS_REGION}}` 字符串替换为在其中收集指标的区域名称。例如 `us-west-2`。请勿在文本之前或之后添加空格。
9. 复制整个 JSON blob 并将其粘贴到 CloudWatch 控制台的文本框中，替换框中已有的内容。
10. 选择更新、保存控制面板。

#### (可选) 为 Prometheus 指标测试设置示例容器化 Amazon EKS 工作负载

要测试 CloudWatch Container Insights 中的 Prometheus 指标支持，您可设置以下一个或多个容器化工作负载。具有 Prometheus 支持的 CloudWatch 代理会自动从这些工作负载中的每一个收集指标。要查看默认情况下收集的指标，请参阅 [CloudWatch 代理收集的 Prometheus 指标 \(p. 480\)](#)。

在安装任意这些工作负载之前，您必须输入以下命令来安装 Helm 3.x：

```
brew install helm
```

有关更多信息，请参阅 [Helm](#)。

##### 主题

- 为 Amazon EKS 和 Kubernetes 设置 Amazon App Mesh 示例工作负载 (p. 467)
- 使用 Amazon EKS 和 Kubernetes 上的示例流量设置 NGINX (p. 469)
- 使用 Amazon EKS 和 Kubernetes 上的指标导出器设置 memcached (p. 470)
- 在 Amazon EKS 和 Kubernetes 上设置 Java/JMX 示例工作负载 (p. 471)
- 使用 Amazon EKS 和 Kubernetes 上的指标导出器设置 HAProxy (p. 475)
- 添加新 Prometheus 抓取目标的教程：Amazon EKS 和 Kubernetes 集群上的 Redis (p. 476)

## 为 Amazon EKS 和 Kubernetes 设置 Amazon App Mesh 示例工作负载

CloudWatch Container Insights 支持 Amazon App Mesh 中的 Prometheus 支持。以下部分介绍了如何设置 App Mesh。

CloudWatch Container Insights 还可以收集 App Mesh Envoy 访问日志。有关更多信息，请参阅 [\(可选\) 启用 App Mesh Envoy 访问日志 \(p. 386\)](#)。

### 主题

- [在具有 EC2 启动类型或 Kubernetes 集群的 Amazon EKS 集群上设置 Amazon App Mesh 示例工作负载 \(p. 467\)](#)
- [使用 Fargate 启动类型在 Amazon EKS 集群上设置 Amazon App Mesh 示例工作负载 \(p. 468\)](#)

## 在具有 EC2 启动类型或 Kubernetes 集群的 Amazon EKS 集群上设置 Amazon App Mesh 示例工作负载

如果您要在运行 Amazon EKS 且具有 EC2 启动类型的集群或 Kubernetes 集群上设置 App Mesh，请使用这些说明。

### 配置 IAM 权限

必须将 `AWSAppMeshFullAccess` 策略添加到您的 Amazon EKS 或 Kubernetes 节点组的 IAM 角色中。在 Amazon EKS 上，此节点组名称类似于 `eksctl-integ-test-eks-prometheus-NodeInstanceRole-ABCDEFHIJKL`。在 Kubernetes 上，它可能类似于 `nodes.integ-test-kops-prometheus.k8s.local`。

### 安装 App Mesh

要安装 App Mesh Kubernetes 控制器，请按照 [App Mesh 控制器](#) 中的说明进行操作。

### 安装示例应用程序

[aws-app-mesh-examples](#) 包含多个 Kubernetes App Mesh 演练。在本教程中，您将安装一个示例颜色应用程序，该应用程序会展示 http 路由如何使用标头来匹配传入请求。

使用示例 App Mesh 应用程序来测试 Container Insights

1. 按照以下说明安装应用程序：<https://github.com/aws/aws-app-mesh-examples/tree/master/walkthroughs/howto-k8s-http-headers>。
2. 启动 curler pod 以生成流量：

```
kubectl -n default run -it curler --image=tutum/curl /bin/bash
```

3. 通过更改 HTTP 标头来对不同的端点执行 curl。多次运行 curl 命令，如下所示：

```
curl -H "color_header: blue" front.howto-k8s-http-headers.svc.cluster.local:8080/; echo;

curl -H "color_header: red" front.howto-k8s-http-headers.svc.cluster.local:8080/; echo;

curl -H "color_header: yellow" front.howto-k8s-http-headers.svc.cluster.local:8080/; echo;
```

4. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
5. 在集群运行的 Amazon 区域的导航窗格中，选择 Metrics (指标)。指标位于 ContainerInsights/Prometheus 命名空间中。
6. 要查看 CloudWatch Logs 事件，请在导航窗格中选择 Log groups (日志组)。事件位于日志流 `kubernetes-pod-appmesh-envoy` 的日志组 `/aws/containerinsights/your_cluster_name/prometheus` 中。

## 删除 App Mesh 测试环境

App Mesh 和示例应用程序使用完成后，请使用以下命令删除不必要的资源。输入以下命令删除示例应用程序：

```
cd aws-app-mesh-examples/walkthroughs/howto-k8s-http-headers/
kubectl delete -f _output/manifest.yaml
```

输入以下命令删除 App Mesh 控制器：

```
helm delete appmesh-controller -n appmesh-system
```

## 使用 Fargate 启动类型在 Amazon EKS 集群上设置 Amazon App Mesh 示例工作负载

如果您要在运行 Amazon EKS 且具有 Fargate 启动类型的集群上设置 App Mesh，请遵循这些说明。

### 配置 IAM 权限

要设置 IAM 权限，请输入以下命令。将 **MyCluster** 替换为您的集群的名称。

```
eksctl create iamserviceaccount --cluster MyCluster \
--namespace howto-k8s-fargate \
--name appmesh-pod \
--attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshEnvoyAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapDiscoverInstanceAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess \
--attach-policy-arn arn:aws:iam::aws:policy/CloudWatchLogsFullAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshFullAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapFullAccess \
--override-existing-serviceaccounts \
--approve
```

## 安装 App Mesh

要安装 App Mesh Kubernetes 控制器，请按照 [App Mesh 控制器](#) 中的说明进行操作。请务必按照有关 Fargate 启动类型的 Amazon EKS 的说明进行操作。

### 安装示例应用程序

[aws-app-mesh-examples](#) 包含多个 Kubernetes App Mesh 演练。在本教程中，您将安装适用于 Fargate 启动类型的 Amazon EKS 集群的示例颜色应用程序。

使用示例 App Mesh 应用程序来测试 Container Insights

1. 按照以下说明安装应用程序：<https://github.com/aws/aws-app-mesh-examples/tree/master/walkthroughs/howto-k8s-fargate>。

这些说明假定您正在使用正确的 Fargate 配置文件创建新集群。如果您想使用已设置的 Amazon EKS 集群，您可以使用以下命令为本演示设置该集群。将 **MyCluster** 替换为您的集群的名称。

```
eksctl create iamserviceaccount --cluster MyCluster \
--namespace howto-k8s-fargate \
--name appmesh-pod \
--attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshEnvoyAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapDiscoverInstanceAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess \
--attach-policy-arn arn:aws:iam::aws:policy/CloudWatchLogsFullAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshFullAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapFullAccess \
--override-existing-serviceaccounts \
--approve
```

```
--approve
```

```
eksctl create fargateprofile --cluster MyCluster \
--namespace howto-k8s-fargate --name howto-k8s-fargate
```

2. 端口转发前端应用程序部署：

```
kubect1 -n howto-k8s-fargate port-forward deployment/front 8080:8080
```

3. Curl 前端应用程序：

```
while true; do curl -s http://localhost:8080/color; sleep 0.1; echo ; done
```

4. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
5. 在集群运行的 Amazon 区域的导航窗格中，选择 Metrics ( 指标 )。指标位于 ContainerInsights/Prometheus 命名空间中。
6. 要查看 CloudWatch Logs 事件，请在导航窗格中选择 Log groups ( 日志组 )。事件位于日志流 kubernetes-pod-appmesh-envoy 的日志组 /aws/containerinsights/*your\_cluster\_name*/prometheus 中。

## 删除 App Mesh 测试环境

App Mesh 和示例应用程序使用完成后，请使用以下命令删除不必要的资源。输入以下命令删除示例应用程序：

```
cd aws-app-mesh-examples/walkthroughs/howto-k8s-fargate/
kubectl delete -f _output/manifest.yaml
```

输入以下命令删除 App Mesh 控制器：

```
helm delete appmesh-controller -n appmesh-system
```

## 使用 Amazon EKS 和 Kubernetes 上的示例流量设置 NGINX

NGINX 是一个 Web 服务器，也可以用作负载均衡器和反向代理。有关 Kubernetes 如何使用 NGINX for ingress 的更多信息，请参阅 [kubernetes/ingress-nginx](#)。

安装带有示例流量服务的 Ingress-NGINX 以测试 Container Insights Prometheus 支持

1. 输入以下命令以添加 Helm ingress-nginx 存储库：

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

2. 输入以下命令：

```
kubect1 create namespace nginx-ingress-sample

helm install my-nginx ingress-nginx/ingress-nginx \
--namespace nginx-ingress-sample \
--set controller.metrics.enabled=true \
--set-string controller.metrics.service.annotations."prometheus\.io/port"="10254" \
--set-string controller.metrics.service.annotations."prometheus\.io/scrape"="true"
```

3. 通过输入以下命令检查服务是否正确启动：

```
kubect1 get service -n nginx-ingress-sample
```

此命令的输出应显示多列，包括一个 EXTERNAL-IP 列。

4. 将 EXTERNAL-IP 变量设置为 NGINX 摄取控制器的行中 EXTERNAL-IP 列的值。

```
EXTERNAL_IP=your-nginx-controller-external-ip
```

5. 输入以下命令启动一些示例 NGINX 流量。

```
SAMPLE_TRAFFIC_NAMESPACE=nginx-sample-traffic
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/master/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_traffic/nginx-traffic/nginx-traffic-sample.yaml |
sed "s/{{external_ip}}/$EXTERNAL_IP/g" |
sed "s/{{namespace}}/$SAMPLE_TRAFFIC_NAMESPACE/g" |
kubectl apply -f -
```

6. 输入以下命令以确认所有三个 pod 都处于 Running 状态。

```
kubectl get pod -n $SAMPLE_TRAFFIC_NAMESPACE
```

如果它们正在运行，您应该很快在 ContainerInsights/Prometheus 命名空间中看到指标。

## 卸载 NGINX 和示例流量应用程序

1. 输入以下命令删除示例流量服务：

```
kubectl delete namespace $SAMPLE_TRAFFIC_NAMESPACE
```

2. 按 Helm 版本名称删除 NGINX 出口。

```
helm uninstall my-nginx --namespace nginx-ingress-sample
kubectl delete namespace nginx-ingress-sample
```

## 使用 Amazon EKS 和 Kubernetes 上的指标导出器设置 memcached

memcached 是一个开源内存对象缓存系统。有关更多信息，请参阅[什么是 Memcached？](#)

如果您在具有 Fargate 启动类型的集群上运行 memcached，则需要在执行此过程中的步骤之前设置 Fargate 配置文件。要设置配置文件，请输入以下命令。将 *MyCluster* 替换为您的集群的名称。

```
eksctl create fargateprofile --cluster MyCluster \
--namespace memcached-sample --name memcached-sample
```

## 安装带有 Metric Exporter 的 memcached 以测试 Container Insights Prometheus 支持

1. 输入以下命令以添加存储库：

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. 输入以下命令以创建新的命名空间：

```
kubectl create namespace memcached-sample
```

3. 输入以下命令以安装 Memcached



```
helm install my-memcached bitnami/memcached --namespace memcached-sample \
--set metrics.enabled=true \
--set-string serviceAnnotations.prometheus\\.io/port="9150" \
--set-string serviceAnnotations.prometheus\\.io/scrape="true"
```

4. 输入以下命令以确认正在运行的服务的注释：

```
kubectl describe service my-memcached-metrics -n memcached-sample
```

您应该看到以下两个注释：

```
Annotations: prometheus.io/port: 9150
 prometheus.io/scrape: true
```

#### 卸载 memcached

- 输入以下命令：

```
helm uninstall my-memcached --namespace memcached-sample
kubectl delete namespace memcached-sample
```

### 在 Amazon EKS 和 Kubernetes 上设置 Java/JMX 示例工作负载

JMX Exporter 是 Prometheus 的官方导出程序，可以将 JMX MBeans 作为 Prometheus 指标进行抓取和公开。有关详细信息，请参阅 [prometheus/jmx\\_exporter](#)。

Container Insights 可以使用 JMX Exporter 从 Java 虚拟机 (JVM)、Java 和 Tomcat (Catalina) 收集预定义的 Prometheus 指标。

#### 默认 Prometheus 抓取配置

默认情况下，支持 Prometheus 的 CloudWatch 代理从 Amazon EKS 或 Kubernetes 集群中的每个 pod 上的 `http://CLUSTER_IP:9404/metrics` 抓取 Java/JMX Prometheus 指标。这是通过 Prometheus `kubernetes_sd_config` 的 `role: pod` 发现来完成的。9404 是 Prometheus 为 JMX Exporter 分配的默认端口。有关 `role: pod` 发现的更多信息，请参阅 [pod](#)。您可以将 JMX Exporter 配置为在不同端口或 `metrics_path` 上公开指标。如果您更改了端口或路径，请更新 CloudWatch 代理 config 映射中的默认 `jmx_scrape_config`。运行以下命令以获取当前 CloudWatch 代理 Prometheus 配置：

```
kubectl describe cm prometheus-config -n amazon-cloudwatch
```

要更改的字段是 `/metrics` 和 `regex: '.*:9404$'` 字段，如以下示例中突出显示所示。

```
job_name: 'kubernetes-jmx-pod'
sample_limit: 10000
metrics_path: /metrics
kubernetes_sd_configs:
- role: pod
relabel_configs:
- source_labels: [__address__]
 action: keep
regex: '.*:9404$'
- action: replace
 regex: (.+)
 source_labels:
```

## 其他 Prometheus 抓取配置

如果您通过 Kubernetes Service 使用 Java/JMX Prometheus 导出器公开在一组 pod 上运行的应用程序，您还可以切换到使用 Prometheus `kubernetes_sd_config` 的 `role: service` 发现或 `role: endpoint` 发现。有关这些发现方法的更多信息，请参阅 [服务](#)、[端点](#) 和 [<kubernetes\\_sd\\_config>](#)。

这两种服务发现模式提供了更多元标签，这对构建 CloudWatch 指标维度大有裨益。例如，您可以将 `__meta_kubernetes_service_name` 重新标记为 `Service` 并将其包含在指标维度中。有关自定义 CloudWatch 指标及其维度的更多信息，请参阅 [Prometheus 的 CloudWatch 代理配置](#) (p. 424)。

## 带有 JMX Exporter 的 Docker 镜像

接下来，构建 Docker 镜像。以下各节提供了两个示例 Dockerfiles。

构建镜像后，请将其加载到 Amazon EKS 或 Kubernetes 中，然后运行以下命令以验证 JMX\_EXPORTER 是否在端口 9404 上公开 Prometheus 指标。使用正在运行的 pod 名称替换 `$JAR_SAMPLE_TRAFFIC_POD`，并使用您的应用程序命名空间替换 `$JAR_SAMPLE_TRAFFIC_NAMESPACE`。

如果您在具有 Fargate 启动类型的集群上运行 JMX Exporter，则在执行此过程中的步骤之前，您还需要设置 Fargate 配置文件。要设置配置文件，请输入以下命令。将 `MyCluster` 替换为您的集群的名称。

```
eksctl create fargateprofile --cluster MyCluster \
--namespace $JAR_SAMPLE_TRAFFIC_NAMESPACE\
--name $JAR_SAMPLE_TRAFFIC_NAMESPACE
```

```
kubectl exec $JAR_SAMPLE_TRAFFIC_POD -n $JARCAT_SAMPLE_TRAFFIC_NAMESPACE -- curl http://
localhost:9404
```

## 示例：具有 Prometheus 指标的 Apache Tomcat Docker 镜像

默认情况下，Apache Tomcat 服务器公开 JMX MBeans。您可以将 JMX Exporter 与 Tomcat 集成，以便将 JMX MBeans 作为 Prometheus 指标公开。以下示例 Dockerfile 显示了构建测试镜像的步骤：

```
From Tomcat 9.0 JDK8 OpenJDK
FROM tomcat:9.0-jdk8-openjdk

RUN mkdir -p /opt/jmx_exporter

COPY ./jmx_prometheus_javaagent-0.12.0.jar /opt/jmx_exporter
COPY ./config.yaml /opt/jmx_exporter
COPY ./setenv.sh /usr/local/tomcat/bin
COPY your web application.war /usr/local/tomcat/webapps/

RUN chmod o+x /usr/local/tomcat/bin/setenv.sh

ENTRYPOINT ["catalina.sh", "run"]
```

下面的列表解释了此 Dockerfile 中的四个 COPY 行。

- 从 [https://github.com/prometheus/jmx\\_exporter](https://github.com/prometheus/jmx_exporter) 下载最新的 JMX Exporter jar 文件。
- `config.yaml` 是 JMX Exporter 配置文件。有关更多信息，请参阅 [https://github.com/prometheus/jmx\\_exporter#Configuration](https://github.com/prometheus/jmx_exporter#Configuration)。

以下是 Java 和 Tomcat 的示例配置文件：

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true
```

```
rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
 name: java_lang_OperatingSystem_$1
 type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
 name: java_lang_threading_$1
 type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
 name: catalina_globalrequestprocessor_$3_total
 labels:
 port: "$2"
 protocol: "$1"
 help: Catalina global $3
 type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=-_!:.],)*[-a-zA-Z0-9+&@#/%?=-_!:.], name=([-a-zA-Z0-9+/%~_-!:.]*), J2EEApplication=none,
J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
 name: catalina_servlet_$3_total
 labels:
 module: "$1"
 servlet: "$2"
 help: Catalina servlet $3 total
 type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name=\"(\w+-\w+)-(\d+)\"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
 name: catalina_threadpool_$3
 labels:
 port: "$2"
 protocol: "$1"
 help: Catalina threadpool $3
 type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=-_!:.],)*[-a-zA-Z0-9+&@#/%?=-_!:.], context=([-a-zA-Z0-9+/%~_-!:.]*)><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
 name: catalina_session_$3_total
 labels:
 context: "$2"
 host: "$1"
 help: Catalina session $3 total
 type: COUNTER

- pattern: ".*"
```

- `setenv.sh` 是一个 Tomcat 启动脚本，用于启动 JMX Exporter 和 Tomcat，并在本地主机的端口 9404 上公开 Prometheus 指标。它还为 JMX Exporter 提供 `config.yaml` 文件路径。

```
$ cat setenv.sh
export JAVA_OPTS="-javaagent:/opt/jmx_exporter/jmx_prometheus_javaagent-0.12.0.jar=9404:/opt/jmx_exporter/config.yaml $JAVA_OPTS"
```

- 您的 Web 应用程序 `.war` 是由 Tomcat 加载的 Web 应用程序 `war` 文件。

使用此配置构建 Docker 镜像并将其上载到镜像存储库。

**示例：具有 Prometheus 指标的 Java Jar 应用程序 Docker 镜像**

以下示例 Dockerfile 显示了构建测试镜像的步骤：

```
Alpine Linux with OpenJDK JRE
FROM openjdk:8-jre-alpine

RUN mkdir -p /opt/jmx_exporter

COPY ./jmx_prometheus_javaagent-0.12.0.jar /opt/jmx_exporter
COPY ./SampleJavaApplication-1.0-SNAPSHOT.jar /opt/jmx_exporter
COPY ./start_exporter_example.sh /opt/jmx_exporter
COPY ./config.yaml /opt/jmx_exporter

RUN chmod -R o+x /opt/jmx_exporter
RUN apk add curl

ENTRYPOINT exec /opt/jmx_exporter/start_exporter_example.sh
```

下面的列表解释了此 Dockerfile 中的四个 COPY 行。

- 从 [https://github.com/prometheus/jmx\\_exporter](https://github.com/prometheus/jmx_exporter) 下载最新的 JMX Exporter jar 文件。
- config.yaml 是 JMX Exporter 配置文件。有关更多信息，请参阅 [https://github.com/prometheus/jmx\\_exporter#Configuration](https://github.com/prometheus/jmx_exporter#Configuration)。

以下是 Java 和 Tomcat 的示例配置文件：

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true

rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
 name: java_lang_OperatingSystem_$1
 type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
 name: java_lang_threading_$1
 type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
 name: catalina_globalrequestprocessor_$3_total
 labels:
 port: "$2"
 protocol: "$1"
 help: Catalina global $3
 type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//[(-a-zA-Z0-9+&@#/%?=-_!|:.,;]*[-a-zA-Z0-9+&@#/%?=-_!|:.,;]*), name=(-a-zA-Z0-9+/%?=-_!|:.,;)*, J2EEApplication=none, J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
 name: catalina_servlet_$3_total
 labels:
 module: "$1"
 servlet: "$2"
 help: Catalina servlet $3 total
 type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name=\"(\w+-\w+)-(\d+)\"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
 name: catalina_threadpool_$3
 labels:
 port: "$2"
 protocol: "$1"
 help: Catalina threadpool $3
 type: GAUGE
```

```
- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=-_~|!:.];)*[-a-zA-Z0-9+&@#/%?=-_~|!:.]*>(processingTime|sessionCounter|rejectedSessions|expiredSessions)'
 name: catalina_session_$3_total
 labels:
 context: "$2"
 host: "$1"
 help: Catalina session $3 total
 type: COUNTER

- pattern: ".*"
```

- `start_exporter_example.sh` 是用于启动导出了 Prometheus 指标的 JAR 应用程序的脚本。它还为 JMX Exporter 提供 `config.yaml` 文件路径。

```
$ cat start_exporter_example.sh
java -javaagent:/opt/jmx_exporter/jmx_prometheus_javaagent-0.12.0.jar=9404:/opt/jmx_exporter/config.yaml -cp /opt/jmx_exporter/SampleJavaApplication-1.0-SNAPSHOT.jar com.gubupt.sample.app.App
```

- `SampleJavaApplication-1.0-SNAPSHOT.jar` 是示例 Java 应用程序 jar 文件。将其替换为要监控的 Java 应用程序。

使用此配置构建 Docker 镜像并将其上载到镜像存储库。

### 使用 Amazon EKS 和 Kubernetes 上的指标导出器设置 HAProxy

HAProxy 是一个开源代理应用程序。有关更多信息，请参阅 [HAProxy](#)。

如果您在具有 Fargate 启动类型的集群上运行 HAProxy，则在执行此过程中的步骤之前，您需要设置 Fargate 配置文件。要设置配置文件，请输入以下命令。将 `MyCluster` 替换为您的集群的名称。

```
eksctl create fargateprofile --cluster MyCluster \
--namespace haproxy-ingress-sample --name haproxy-ingress-sample
```

安装带有 Metric Exporter 的 HAProxy 以测试 Container Insights Prometheus 支持

1. 输入以下命令以添加 Helm incubator 存储库：

```
helm repo add haproxy-ingress https://haproxy-ingress.github.io/charts
```

2. 输入以下命令以创建新的命名空间：

```
kubectl create namespace haproxy-ingress-sample
```

3. 输入以下命令来安装 HAProxy：

```
helm install haproxy haproxy-ingress/haproxy-ingress \
--namespace haproxy-ingress-sample \
--set defaultBackend.enabled=true \
--set controller.stats.enabled=true \
--set controller.metrics.enabled=true \
--set-string controller.metrics.service.annotations."prometheus\.io/port"="9101" \
--set-string controller.metrics.service.annotations."prometheus\.io/scrape"="true"
```

4. 输入以下命令以确认服务的注释：

```
kubectl describe service haproxy-haproxy-ingress-metrics -n haproxy-ingress-sample
```

您应该看到以下注释。

```
Annotations: prometheus.io/port: 9101
 prometheus.io/scrape: true
```

## 卸载 HAProxy

- 输入以下命令：

```
helm uninstall haproxy --namespace haproxy-ingress-sample
kubectl delete namespace haproxy-ingress-sample
```

## 添加新 Prometheus 抓取目标的教程：Amazon EKS 和 Kubernetes 集群上的 Redis

本教程提供了在 Amazon EKS 和 Kubernetes 上抓取示例 Redis 应用程序的 Prometheus 指标的实践介绍。Redis (<https://redis.io/>) 是一个开源（获得 BSD 许可）的内存数据结构存储，用作数据库、缓存和消息代理。有关更多信息，请参阅 [redis](#)。

redis\_exporter（MIT 许可证）用于在指定端口（默认值：0.0.0.0:9121）上公开 Redis prometheus 指标。有关更多信息，请参阅 [redis\\_exporter](#)。

本教程使用了以下两个 Docker Hub 存储库中的 Docker 镜像：

- [redis](#)
- [redis\\_exporter](#)

## 安装公开 Prometheus 指标的示例 Redis 工作负载

1. 为示例 Redis 工作负载设置命名空间。

```
REDIS_NAMESPACE=redis-sample
```

2. 如果您在具有 Fargate 启动类型的集群上运行 Redis，则需要设置 Fargate 配置文件。要设置配置文件，请输入以下命令。将 **MyCluster** 替换为您的集群的名称。

```
eksctl create fargateprofile --cluster MyCluster \
--namespace $REDIS_NAMESPACE --name $REDIS_NAMESPACE
```

3. 输入以下命令以安装示例 Redis 工作负载。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/eks-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_traffic/redis/redis-traffic-sample.yaml \
| sed "s/{{namespace}}/$REDIS_NAMESPACE/g" \
| kubectl apply -f -
```

4. 安装包括一个名为 my-redis-metrics 的服务，该服务在端口 9121 上公开 Redis Prometheus 指标。输入以下命令以获取该服务的详细信息：

```
kubectl describe service/my-redis-metrics -n $REDIS_NAMESPACE
```

在结果的 Annotations 部分，您将看到与 CloudWatch 代理的 Prometheus 抓取配置相匹配的两个注释，以便其可以自动发现工作负载：

```
prometheus.io/port: 9121
prometheus.io/scrape: true
```

相关的 Prometheus 抓取配置可以在 `kubernetes-eks.yaml` 或 `kubernetes-k8s.yaml` 部分的 `-job_name: kubernetes-service-endpoints` 中找到。

## 开始在 CloudWatch 中收集 Redis Prometheus 指标

1. 通过输入以下任一命令，下载 `kubernetes-eks.yaml` 或 `kubernetes-k8s.yaml` 文件的最新版本。对于具有 EC2 启动类型的 Amazon EKS 集群，请输入此命令。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
```

对于具有 Fargate 启动类型的 Amazon EKS 集群，请输入此命令。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml
```

对于在 Amazon EC2 实例上运行的 Kubernetes 集群，请输入此命令。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml
```

2. 使用文本编辑器打开文件，然后找到 `cwagentconfig.json` 部分。添加以下子部分并保存更改。确保遵循现有模式缩进。

```
{
 "source_labels": ["pod_name"],
 "label_matcher": "^redis-instance$",
 "dimensions": [["Namespace", "ClusterName"]],
 "metric_selectors": [
 "^redis_net_(in|out)put_bytes_total$",
 "^redis_(expired|evicted)_keys_total$",
 "^redis_keyspace_(hits|misses)_total$",
 "^redis_memory_used_bytes$",
 "^redis_connected_clients$"
]
},
{
 "source_labels": ["pod_name"],
 "label_matcher": "^redis-instance$",
 "dimensions": [["Namespace", "ClusterName", "cmd"]],
 "metric_selectors": [
 "^redis_commands_total$"
]
},
{
 "source_labels": ["pod_name"],
 "label_matcher": "^redis-instance$",
 "dimensions": [["Namespace", "ClusterName", "db"]],
 "metric_selectors": [
 "^redis_db_keys$"
]
}
```

```
},
```

您添加的部分将 Redis 指标放入 CloudWatch 代理允许列表中。有关这些指标的列表，请参阅以下部分。

3. 如果您已在此集群中部署了具有 Prometheus 支持的 CloudWatch 代理，则必须通过输入以下命令将其删除。

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

4. 通过输入以下任一命令，使用更新的配置部署 CloudWatch 代理。替换 **MyCluster** 和 **region####** 以匹配您的设置。

对于具有 EC2 启动类型的 Amazon EKS 集群，请输入此命令。

```
kubectl apply -f prometheus-eks.yaml
```

对于具有 Fargate 启动类型的 Amazon EKS 集群，请输入此命令。

```
cat prometheus-eks-fargate.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

对于 Kubernetes 集群，请输入此命令。

```
cat prometheus-k8s.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

## 查看 Redis Prometheus 指标

本教程会将以下指标发送到 CloudWatch 中的 ContainerInsights/Prometheus 命名空间。您可以使用 CloudWatch 控制台查看该命名空间中的指标。

| 指标名称                     | 维度                          |  |
|--------------------------|-----------------------------|--|
| redis_net_input_bytes    | ClusterName, Namespace      |  |
| redis_net_output_bytes   | ClusterName, Namespace      |  |
| redis_expired_keys_total | ClusterName, Namespace      |  |
| redis_evicted_keys_total | ClusterName, Namespace      |  |
| redis_keyspace_hits      | ClusterName, Namespace      |  |
| redis_keyspace_misses    | ClusterName, Namespace      |  |
| redis_memory_used_bytes  | ClusterName, Namespace      |  |
| redis_connected_clients  | ClusterName, Namespace      |  |
| redis_commands_total     | ClusterName, Namespace, cmd |  |
| redis_db_keys            | ClusterName, Namespace, db  |  |



#### Note

cmd 维度的值可以是

append、client、command、config、dbsize、flushall、get、incr、info、latency  
或 slowlog。

db 维度的值可以从 db0 到 db15。

您还可以为 Redis Prometheus 指标创建 CloudWatch 控制面板。

为 Redis Prometheus 指标创建控制面板

1. 创建环境变量，替换以下值以匹配部署。

```
DASHBOARD_NAME=your_cw_dashboard_name
REGION_NAME=your_metric_region_such_as_us-east-1
CLUSTER_NAME=your_k8s_cluster_name_here
NAMESPACE=your_redis_service_namespace_here
```

2. 输入以下命令以创建控制面板。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/master/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_cloudwatch_dashboards/redis/cw_dashboard_redis.json \
| sed "s/{{YOUR_AWS_REGION}}/{{REGION_NAME}}/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/{{CLUSTER_NAME}}/g" \
| sed "s/{{YOUR_NAMESPACE}}/{{NAMESPACE}}/g" \
```

## CloudWatch 代理的 Prometheus 指标类型转换

Prometheus 客户端库提供四种核心指标类型：

- 计数器
- 计量表
- 摘要
- 直方图

CloudWatch 代理支持计数器、计量表和汇总指标类型。对直方图指标的支持计划在未来的版本中推出。

CloudWatch 代理会删除具有不受支持的直方图指标类型的 Prometheus 指标。有关更多信息，请参阅[录入丢弃的 Prometheus 指标 \(p. 491\)](#)。

### 计量表指标

Prometheus 计量表指标是表示可以任意上下的单个数值的度量。CloudWatch 代理会抓取计量表指标并直接发送这些值。

### 计数器指标

Prometheus 计数器指标是一个累积指标，表示单个单调增加的计数器，其值只能增加或重置为零。CloudWatch 代理根据上次抓取计算增量，并将增量值作为日志事件中的指标值发送。因此，CloudWatch 代理将从第二次抓取开始生成一个日志事件，并继续进行后续抓取（如果有）。

### 汇总指标

Prometheus 汇总指标是一种复杂的指标类型，由多个数据点表示。它提供观察的总数和所有观察值的总和。它在滑动时间窗口内计算可配置的分位数。

汇总指标的总和和计数是累积的，但分位数不是。以下示例显示了分位数的方差。

```
TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 7.123e-06
go_gc_duration_seconds{quantile="0.25"} 9.204e-06
go_gc_duration_seconds{quantile="0.5"} 1.1065e-05
go_gc_duration_seconds{quantile="0.75"} 2.8731e-05
go_gc_duration_seconds{quantile="1"} 0.003841496
go_gc_duration_seconds_sum 0.37630427
go_gc_duration_seconds_count 9774
```

CloudWatch 代理以与处理计数器指标相同的方式处理汇总指标的总和和计数，如上一节所述。CloudWatch 代理会保留最初报告的分位数。

## CloudWatch 代理收集的 Prometheus 指标

具有 Prometheus 支持的 CloudWatch 代理可自动从多个服务和 workload 收集指标。以下部分列出了默认情况下收集的指标。您还可以将代理配置为从这些服务收集更多指标，并从其他应用程序和服务收集 Prometheus 指标。有关可收集的其他指标的更多信息，请参阅 [Prometheus 的 CloudWatch 代理配置 \(p. 424\)](#)。

从 Amazon EKS 和 Kubernetes 集群收集的 Prometheus 指标位于 ContainerInsights/Prometheus 命名空间中。从 Amazon ECS 集群收集的 Prometheus 指标位于 ECS/ContainerInsights/Prometheus 命名空间中。

### 主题

- [App Mesh 的 Prometheus 指标 \(p. 480\)](#)
- [NGINX 的 Prometheus 指标 \(p. 482\)](#)
- [Memcached 的 Prometheus 指标 \(p. 483\)](#)
- [Java/JMX 的 Prometheus 指标 \(p. 483\)](#)
- [HAProxy 的 Prometheus 指标 \(p. 485\)](#)

## App Mesh 的 Prometheus 指标

从 App Mesh 自动收集以下指标。

CloudWatch Container Insights 还可以收集 App Mesh Envoy 访问日志。有关更多信息，请参阅 [\(可选\) 启用 App Mesh Envoy 访问日志 \(p. 386\)](#)。

Amazon EKS 和 Kubernetes 集群上 App Mesh 的 Prometheus 指标

| 指标名称                                      | 维度                                                                                |  |
|-------------------------------------------|-----------------------------------------------------------------------------------|--|
| envoy_http_downstream_active_connections  | ClusterName, Namespace                                                            |  |
| envoy_http_downstream_active_connections  | ClusterName, Namespace                                                            |  |
|                                           | ClusterName, Namespace, envoy_http_conn_manager_prefix, envoy_response_code_class |  |
| envoy_cluster_upstream_active_connections | ClusterName, Namespace                                                            |  |
| envoy_cluster_upstream_active_connections | ClusterName, Namespace                                                            |  |
| envoy_cluster_member_active_connections   | ClusterName, Namespace                                                            |  |
| envoy_cluster_member_active_connections   | ClusterName, Namespace                                                            |  |
| envoy_server_memory_allocated_bytes       | ClusterName, Namespace                                                            |  |
| envoy_server_memory_allocated_bytes       | ClusterName, Namespace                                                            |  |

| 指标名称                                              | 维度                     |  |
|---------------------------------------------------|------------------------|--|
| envoy_cluster_upstream_connections                | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections_eject          | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections_low            | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections                | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections                | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections                | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections_with_active_rq | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections_active_rq      | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections                | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections_reading_total  | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections_reading_total  | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections_up_total       | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections_total          | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections                | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections                | ClusterName, Namespace |  |
| envoy_cluster_upstream_connections                | ClusterName, Namespace |  |
| envoy_server_live                                 | ClusterName, Namespace |  |
| envoy_server_uptime                               | ClusterName, Namespace |  |

#### Amazon ECS 集群上 App Mesh 的 Prometheus 指标

| 指标名称                                  | 维度                                |  |
|---------------------------------------|-----------------------------------|--|
| envoy_http_downstream_connections     | ClusterName, TaskDefinitionFamily |  |
| envoy_http_downstream_connections     | ClusterName, TaskDefinitionFamily |  |
| envoy_cluster_upstream_connections    | ClusterName, TaskDefinitionFamily |  |
| envoy_cluster_upstream_connections    | ClusterName, TaskDefinitionFamily |  |
| envoy_cluster_member_connections      | ClusterName, TaskDefinitionFamily |  |
| envoy_cluster_member_connections      | ClusterName, TaskDefinitionFamily |  |
| envoy_server_memory_working_set_bytes | ClusterName, TaskDefinitionFamily |  |
| envoy_server_memory_working_set_bytes | ClusterName, TaskDefinitionFamily |  |
| envoy_cluster_upstream_connections    | ClusterName, TaskDefinitionFamily |  |
| envoy_cluster_upstream_connections    | ClusterName, TaskDefinitionFamily |  |

| 指标名称                                             | 维度                                                                                        |  |
|--------------------------------------------------|-------------------------------------------------------------------------------------------|--|
| envoy_cluster_upstream_requests                  | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_50               | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_99               | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_999              | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_active_rq        | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_active_rq_50     | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_active_rq_99     | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_active_rq_999    | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_failed           | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_failed_total     | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_failed_total_50  | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_failed_total_99  | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_failed_total_999 | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_failed_total_50  | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_failed_total_99  | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_cluster_upstream_requests_failed_total_999 | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_server_live                                | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_server_uptime                              | ClusterName、TaskDefinitionFamily                                                          |  |
| envoy_http_downstream_requests                   | ClusterName、TaskDefinitionFamily、envoy_http_conn_manager_prefix、envoy_response_code_class |  |

#### Note

TaskDefinitionFamily 是网格的 Kubernetes 命名空间。  
envoy\_http\_conn\_manager\_prefix 的值可以是 ingress、egress 或 admin。  
envoy\_response\_code\_class 的值可以是 1 (代表 1xx)、2 (代表 2xx)、3 (代表 3xx)、4 (代表 4xx) 或 5 (代表 5xx)。

### NGINX 的 Prometheus 指标

从 Amazon EKS 和 Kubernetes 集群上的 NGINX 自动收集以下指标。

| 指标名称                                           | 维度                            |  |
|------------------------------------------------|-------------------------------|--|
| nginx_ingress_controller_cpu_seconds_total     | ClusterName、Namespace、Service |  |
| nginx_ingress_controller_cpu_seconds_total_50  | ClusterName、Namespace、Service |  |
| nginx_ingress_controller_cpu_seconds_total_99  | ClusterName、Namespace、Service |  |
| nginx_ingress_controller_cpu_seconds_total_999 | ClusterName、Namespace、Service |  |
| nginx_ingress_controller_cpu_seconds_total_50  | ClusterName、Namespace、Service |  |
| nginx_ingress_controller_cpu_seconds_total_99  | ClusterName、Namespace、Service |  |
| nginx_ingress_controller_cpu_seconds_total_999 | ClusterName、Namespace、Service |  |

| 指标名称                                  | 维度                                      |  |
|---------------------------------------|-----------------------------------------|--|
| nginx_ingress_controller_memory_bytes | ClusterName, Namespace, Service         |  |
| nginx_ingress_controller_successful   | ClusterName, Namespace, Service         |  |
| nginx_ingress_controller_status       | ClusterName, Namespace, Service, status |  |

## Memcached 的 Prometheus 指标

从 Amazon EKS 和 Kubernetes 集群上的 Memcached 中自动收集以下指标。

| 指标名称                          | 维度                                                                                                                              |  |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------|--|
| memcached_current_items       | ClusterName, Namespace, Service                                                                                                 |  |
| memcached_current_connections | ClusterName, Namespace, Service                                                                                                 |  |
| memcached_limit_bytes         | ClusterName, Namespace, Service                                                                                                 |  |
| memcached_current_bytes       | ClusterName, Namespace, Service                                                                                                 |  |
| memcached_writes_bytes        | ClusterName, Namespace, Service                                                                                                 |  |
| memcached_reads_bytes         | ClusterName, Namespace, Service                                                                                                 |  |
| memcached_items_evicted       | ClusterName, Namespace, Service                                                                                                 |  |
| memcached_items_rejected      | ClusterName, Namespace, Service                                                                                                 |  |
| memcached_commands_total      | ClusterName, Namespace, Service<br>ClusterName, Namespace, Service, command<br>ClusterName, Namespace, Service, status, command |  |

## Java/JMX 的 Prometheus 指标

在 Amazon EKS 和 Kubernetes 集群上收集的指标

在 Amazon EKS 和 Kubernetes 集群上，Container Insights 可以使用 JMX Exporter 从 Java 虚拟机 (JVM)、Java 和 Tomcat (Catalina) 收集以下预定义的 Prometheus 指标。有关更多信息，请参阅 Github 上的 [prometheus/jmx\\_exporter](#)。

Amazon EKS 和 Kubernetes 集群上的 Java/JMX

| 指标名称                                       | 维度                     |  |
|--------------------------------------------|------------------------|--|
| jvm_classes_loaded                         | ClusterName, Namespace |  |
| jvm_threads_current                        | ClusterName, Namespace |  |
| jvm_threads_daemon                         | ClusterName, Namespace |  |
| java_lang_operatingsystem_heap_memory_size | ClusterName, Namespace |  |
| java_lang_operatingsystem_heap_memory_free | ClusterName, Namespace |  |

| 指标名称                                         | 维度                           |  |
|----------------------------------------------|------------------------------|--|
| java_lang_operatingsystem                    | ClusterName, Namespace       |  |
| java_lang_operatingsystemswapspace           | ClusterName, Namespace       |  |
| java_lang_operatingsystemtotalmemorysize     | ClusterName, Namespace       |  |
| java_lang_operatingsystemphymemorysize       | ClusterName, Namespace       |  |
| java_lang_operatingsystemfiledescriptorcount | ClusterName, Namespace       |  |
| java_lang_operatingsysteml1processors        | ClusterName, Namespace       |  |
| jvm_memory_bytes_used                        | ClusterName, Namespace, area |  |
| jvm_memory_pool_bytes_used                   | ClusterName, Namespace, pool |  |

#### Note

area 维度的值可以是 heap 或 nonheap。

pool 维度的值可以是 Tenured Gen、Compress Class Space、Survivor Space、Eden Space、Code Cache 或 Metaspace。

#### Amazon EKS 和 Kubernetes 集群上的 Tomcat/JMX

除了上表中的 Java/JMX 指标外，还收集 Tomcat 工作负载的以下指标。

| 指标名称                                 | 维度                     |  |
|--------------------------------------|------------------------|--|
| catalina_manager_activeconnections   | ClusterName, Namespace |  |
| catalina_manager_rejectedconnections | ClusterName, Namespace |  |
| catalina_globalrequeststotalreceived | ClusterName, Namespace |  |
| catalina_globalrequeststotal         | ClusterName, Namespace |  |
| catalina_globalrequeststotalcount    | ClusterName, Namespace |  |
| catalina_globalrequeststotalpercent  | ClusterName, Namespace |  |
| catalina_globalrequeststotalavgtime  | ClusterName, Namespace |  |

#### Amazon ECS 集群上的 Java/JMX

| 指标名称                               | 维度                               |  |
|------------------------------------|----------------------------------|--|
| jvm_classes_loaded                 | ClusterName、TaskDefinitionFamily |  |
| jvm_threads_current                | ClusterName、TaskDefinitionFamily |  |
| jvm_threads_daemon                 | ClusterName、TaskDefinitionFamily |  |
| java_lang_operatingsystem          | ClusterName、TaskDefinitionFamily |  |
| java_lang_operatingsystemswapspace | ClusterName、TaskDefinitionFamily |  |

| 指标名称                       | 维度                                  |  |
|----------------------------|-------------------------------------|--|
| java_lang_operatingsystem  | ClusterName、TaskDefinitionFamily    |  |
| java_lang_operatingsystem  | ClusterName、TaskDefinitionFamily    |  |
| java_lang_operatingsystem  | ClusterName、TaskDefinitionFamily    |  |
| java_lang_operatingsystem  | ClusterName、TaskDefinitionFamily    |  |
| java_lang_operatingsystem  | ClusterName、TaskDefinitionFamily    |  |
| java_lang_operatingsystem  | ClusterName、TaskDefinitionFamily    |  |
| jvm_memory_bytes_used      | ClusterName、TaskDefinitionFamily、区域 |  |
| jvm_memory_pool_bytes_used | ClusterName、TaskDefinitionFamily、池  |  |

#### Note

area 维度的值可以是 heap 或 nonheap。

pool 维度的值可以是 Tenured Gen、Compress Class Space、Survivor Space、Eden Space、Code Cache 或 Metaspace。

#### Amazon ECS 集群上的 Tomcat/JMX

除上表中的 Java/JMX 指标外，还收集了 Amazon ECS 集群上 Tomcat 工作负载的以下指标。

| 指标名称                               | 维度                               |  |
|------------------------------------|----------------------------------|--|
| catalina_manager_active_sessions   | ClusterName、TaskDefinitionFamily |  |
| catalina_manager_rejected_sessions | ClusterName、TaskDefinitionFamily |  |
| catalina_globalrequest_count       | ClusterName、TaskDefinitionFamily |  |
| catalina_globalrequest_count       | ClusterName、TaskDefinitionFamily |  |
| catalina_globalrequest_count       | ClusterName、TaskDefinitionFamily |  |
| catalina_globalrequest_count       | ClusterName、TaskDefinitionFamily |  |
| catalina_globalrequest_count       | ClusterName、TaskDefinitionFamily |  |

#### HAProxy 的 Prometheus 指标

从 Amazon EKS 和 Kubernetes 集群上的 HAProxy 自动收集以下指标。

收集的指标取决于您使用的 HAProxy Ingress 版本。有关 HAProxy Ingress 及其版本的更多信息，请参阅 [haproxy-ingress](#)。

| 指标名称                  | 维度                            | 可用性                   |
|-----------------------|-------------------------------|-----------------------|
| haproxy_backend_bytes | ClusterName、Namespace、Service | 所有版本的 HAProxy Ingress |
| haproxy_backend_bytes | ClusterName、Namespace、Service | 所有版本的 HAProxy Ingress |

| 指标名称                                 | 维度                                              | 可用性                            |
|--------------------------------------|-------------------------------------------------|--------------------------------|
| haproxy_backend_connections          | ClusterName, Namespace, Service                 | 所有版本的 HAProxy Ingress          |
| haproxy_backend_connections_total    | ClusterName, Namespace, Service                 | 所有版本的 HAProxy Ingress          |
| haproxy_backend_current_connections  | ClusterName, Namespace, Service                 | 所有版本的 HAProxy Ingress          |
| haproxy_backend_http_errors          | ClusterName, Namespace, Service, code, backend  | 所有版本的 HAProxy Ingress          |
| haproxy_backend_status               | ClusterName, Namespace, Service                 | 仅在 HAProxy Ingress 0.10 或更高版本中 |
| haproxy_backend_up                   | ClusterName, Namespace, Service                 | 仅在 HAProxy Ingress 0.10 的更早版本中 |
| haproxy_frontend_bytes               | ClusterName, Namespace, Service                 | 所有版本的 HAProxy Ingress          |
| haproxy_frontend_bytes_total         | ClusterName, Namespace, Service                 | 所有版本的 HAProxy Ingress          |
| haproxy_frontend_connections         | ClusterName, Namespace, Service                 | 所有版本的 HAProxy Ingress          |
| haproxy_frontend_current_connections | ClusterName, Namespace, Service                 | 所有版本的 HAProxy Ingress          |
| haproxy_frontend_http_errors         | ClusterName, Namespace, Service                 | 所有版本的 HAProxy Ingress          |
| haproxy_frontend_http_errors_total   | ClusterName, Namespace, Service, code, frontend | 所有版本的 HAProxy Ingress          |
| haproxy_frontend_requests            | ClusterName, Namespace, Service                 | 所有版本的 HAProxy Ingress          |
| haproxy_frontend_requests_total      | ClusterName, Namespace, Service                 | 所有版本的 HAProxy Ingress          |

#### Note

code 维度的值可以是 1xx、2xx、3xx、4xx、5xx 或 other。  
backend 维度的值可以是：

- 适用于 HAProxy Ingress 0.0.27 或更早版本的 http-default-backend、http-shared-backend 或 httpsback-shared-backend。
- 适用于高于 HAProxy Ingress 0.0.27 版本的 \_default\_backend。

frontend 维度的值可以是：

- 适用于 HAProxy Ingress 0.0.27 或更早版本的 httpfront-default-backend、httpfront-shared-frontend 或 httpfronts。
- 适用于高于 HAProxy Ingress 0.0.27 版本的 \_front\_http 或 \_front\_https。



## 查看 Prometheus 指标

您可以监控所有 Prometheus 指标并发出告警，包括来自 App Mesh、NGINX、Java/JMX、Memcached 和 HAProxy 的精选预聚合指标，以及任何其他您可能已添加的手动配置的 Prometheus 导出程序。有关从其他 Prometheus 导出程序收集指标的更多信息，请参阅[添加新 Prometheus 抓取目标的教程：Prometheus API 服务器指标 \(p. 463\)](#)。

在 CloudWatch 控制台中，Container Insights 提供以下预构建报告：

- 对于 Amazon EKS 和 Kubernetes 集群，有针对 App Mesh、NGINX、HAPROXY、Memcached 和 Java/JMX 的预构建报告。
- 对于 Amazon ECS 集群，有针对 App Mesh 和 Java/JMX 的预构建报告。

Container Insights 还为 Container Insights 从中收集策管指标的每个工作负载提供自定义控制面板。您可以从 GitHub 下载这些控制面板

### 查看所有 Prometheus 指标

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 在命名空间列表中，选择 ContainerInsights/Prometheus 或 ECS/ContainerInsights/Prometheus。
4. 在以下列表中选择一组维度。然后选中您要查看的指标旁边的复选框。

### 查看有关 Prometheus 指标的预构建报告

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择性能监控。
3. 在页面顶部附近的下拉框中，选择任一 Prometheus 选项。

在另一个下拉框中，选择要查看的集群

我们还为 NGINX、App Mesh、Memcached、HAProxy 和 Java/JMX 提供了自定义控制面板。

### 使用 Amazon 提供的自定义控制面板

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Dashboards ( 控制面板 )。
3. 选择创建控制面板。输入新控制面板的名称，然后选择创建控制面板。
4. 在添加到该控制面板中，选择取消。
5. 依次选择操作、编辑控制面板。
6. 下载以下 JSON 文件之一：
  - [Github 上的 NGINX 自定义控制面板源代码。](#)
  - [Github 上的 App Mesh 自定义控制面板源代码。](#)
  - [Github 上的 Memcached 自定义控制面板源代码](#)
  - [Github 上的 HAProxy-Ingress 自定义控制面板源代码](#)
  - [Github 上的 Java/JMX 自定义控制面板源代码。](#)
7. 使用文本编辑器打开下载的 JSON 文件，然后执行以下更改：
  - 将所有 `{{YOUR_CLUSTER_NAME}}` 字符串替换为您的集群的确切名称。请勿在文本之前或之后添加空格。

- 将所有 `{{YOUR_REGION}}` 字符串替换为正在运行集群的 Amazon 区域。例如，**us-west-1** 请勿在文本之前或之后添加空格。
  - 将所有 `{{YOUR_NAMESPACE}}` 字符串替换为您的工作负载的确切命名空间。
  - 将所有 `{{YOUR_SERVICE_NAME}}` 字符串替换为您的工作负载的确切服务名称。例如，**haproxy-haproxy-ingress-controller-metrics**
8. 复制整个 JSON blob 并将其粘贴到 CloudWatch 控制台的文本框中，替换框中已有的内容。
  9. 选择更新、保存控制面板。

## Prometheus 指标故障排除

本节提供有关排除 Prometheus 指标设置故障排除的帮助。

### 主题

- [对 Amazon ECS 上的 Prometheus 指标进行故障排除 \(p. 488\)](#)
- [Amazon EKS 和 Kubernetes 集群上的 Prometheus 指标故障排除 \(p. 490\)](#)

### 对 Amazon ECS 上的 Prometheus 指标进行故障排除

本节提供有关对 Amazon ECS 集群上的 Prometheus 指标设置进行故障排除的帮助。

#### 我没有看到发送到 CloudWatch Logs 的 Prometheus 指标

Prometheus 指标在日志组 `/aws/ecs/containerinsights/cluster-name/Prometheus` 中摄取作为日志事件。如果未创建日志组或未将 Prometheus 指标发送到日志组，您需要首先检查 CloudWatch 代理是否已成功发现 Prometheus 目标。然后检查 CloudWatch 代理的安全组和权限设置。以下步骤会指导您进行调试。

#### 步骤 1：启用 CloudWatch 代理调试模式

首先，通过将以下粗体行添加到 Amazon CloudFormation 模版文件、`cwagent-ecs-prometheus-metric-for-bridge-host.yaml` 或 `cwagent-ecs-prometheus-metric-for-awsvpc.yaml` 中并保存文件，将 CloudWatch 代理更改为调试模式。然后保存文件。

```
cwagentconfig.json: |
{
 "agent": {
 "debug": true
 },
 "logs": {
 "metrics_collected": {
```

针对现有堆栈创建一个新的 Amazon CloudFormation 变更集。将变更集中的其他参数设置为与现有 Amazon CloudFormation 堆栈中相同的值。以下示例适用于使用 EC2 启动类型和桥式网络模式安装在 Amazon ECS 集群中的 CloudWatch 代理。

```
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name
NEW_CHANGESET_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
--template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
--parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
 ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
```

```
ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
ParameterKey=ExecutionRoleName,ParameterValue=$ECS_EXECUTION_ROLE_NAME \
--capabilities CAPABILITY_NAMED_IAM \
--region $AWS_REGION \
--change-set-name $NEW_CHANGESET_NAME
```

转至 Amazon CloudFormation 控制台查看新的变更集，请参阅 `$NEW_CHANGESET_NAME`。应该对 `CWAgentConfigSSMParameter` 资源应用一项更改。通过输入以下命令执行变更集并重新启动 CloudWatch 代理任务。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service your_service_name_here \
--region $AWS_REGION
```

等待大约 10 秒，然后输入以下命令。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service your_service_name_here \
--region $AWS_REGION
```

## 步骤 2：检查 ECS 服务发现日志

CloudWatch 代理的 ECS 任务定义默认启用以下部分中的日志。日志将发送到 `/ecs/ecs-cwagent-prometheus` 日志组中的 CloudWatch Logs。

```
LogConfiguration:
 LogDriver: awslogs
 Options:
 awslogs-create-group: 'True'
 awslogs-group: "/ecs/ecs-cwagent-prometheus"
 awslogs-region: !Ref AWS::Region
 awslogs-stream-prefix: !Sub 'ecs-${ECSLaunchType}-awsvpc'
```

根据字符串 `ECS_SD_Stats` 筛选日志以获取与 ECS 服务发现相关的指标，如下例所示。

```
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeContainerInstances: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeInstancesRequest: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeTaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeTasks: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_ListTasks: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: Exporter_DiscoveredTargetCount: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Get_EC2MetaData: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Get_TaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Size_ContainerInstance: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Size_TaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: Latency: 43.399783ms
```

特定 ECS 服务发现周期的每个指标的含义如下：

- `AWSCLI_DescribeContainerInstances` – `ECS::DescribeContainerInstances` 进行的 API 调用次数。
- `AWSCLI_DescribeInstancesRequest` – `ECS::DescribeInstancesRequest` 进行的 API 调用次数。
- `AWSCLI_DescribeTaskDefinition` – `ECS::DescribeTaskDefinition` 进行的 API 调用次数。
- `AWSCLI_DescribeTasks` – `ECS::DescribeTasks` 进行的 API 调用次数。
- `AWSCLI_ListTasks` – `ECS::ListTasks` 进行的 API 调用次数。

- `ExporterDiscoveredTargetCount` – 已发现并成功导出到容器内的目标结果文件中的 Prometheus 目标数量。
- `LRUCache_Get_EC2MetaData` – 从缓存中检索容器实例元数据的次数。
- `LRUCache_Get_TaskDefinition` – 从缓存中检索 ECS 任务定义元数据的次数。
- `LRUCache_Size_ContainerInstance` – 缓存在内存中的唯一容器实例元数据的数量。
- `LRUCache_Size_TaskDefinition` – 缓存在内存中的唯一 ECS 任务定义的数量。
- 延迟 – 服务发现周期所需的时间。

检查 `ExporterDiscoveredTargetCount` 的值以查看发现的 Prometheus 目标是否符合您的预期。如果不符合，可能的原因如下：

- ECS 服务发现的配置可能与应用程序的设置不匹配。对于基于 docker 标签的服务发现，您的目标容器可能并未在 CloudWatch 代理中配置必要的 docker 标签以将其自动发现。对于 ECS 任务定义 ARN 基于正则表达式的服务发现，CloudWatch 代理中的正则表达式设置可能与应用程序的任务定义不匹配。
- CloudWatch 代理的 ECS 任务角色可能没有权限检索 ECS 任务的元数据。检查 CloudWatch 代理是否已被授予以下只读权限：
  - `ec2:DescribeInstances`
  - `ecs:ListTasks`
  - `ecs:DescribeContainerInstances`
  - `ecs:DescribeTasks`
  - `ecs:DescribeTaskDefinition`

### 步骤 3：检查网络连接以及 ECS 任务角色策略

如果即使 `Exporter_DiscoveredTargetCount` 的值表明已发现 Prometheus 目标，仍然没有日志事件发送到目标 CloudWatch Logs 日志组，这可能是由以下任一原因引起的：

- CloudWatch 代理可能无法连接到 Prometheus 目标端口。检查 CloudWatch 代理背后的安全组设置。私有 IP 应允许 CloudWatch 代理连接到 Prometheus 导出器端口。
- CloudWatch 代理的 ECS 任务角色可能没有 `CloudWatchAgentServerPolicy` 托管策略。CloudWatch 代理的 ECS 任务角色需要具有此策略才能将 Prometheus 指标作为日志事件发送。如果您使用示例 Amazon CloudFormation 模板自动创建 IAM 角色，则 ECS 任务角色和 ECS 执行角色都会被授予执行 Prometheus 监控的最低权限。

## Amazon EKS 和 Kubernetes 集群上的 Prometheus 指标故障排除

本节提供有关对 Amazon EKS 和 Kubernetes 集群上的 Prometheus 指标设置进行故障排除的帮助。

### Amazon EKS 上的一般故障排除步骤

要确认 CloudWatch 代理正在运行，请输入以下命令。

```
kubectl get pod -n amazon-cloudwatch
```

输出应包含其 `NAME` 列中包含 `cwagent-prometheus-id` 并且 `STATUS` column. 中包含 `Running` 的行。

要显示有关正在运行的 pod 的详细信息，请输入以下命令。将 `pod-name` 替换为名称以 `cw-agent-prometheus` 开头的 pod 的完整名称。

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

如果您安装了 CloudWatch Container Insights，则可以使用 CloudWatch Logs Insights 从收集 Prometheus 指标的 CloudWatch 代理查询日志。

## 查询应用程序日志

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 CloudWatch Logs Insights。
3. 选择应用程序日志的日志组 `/aws/containerinsights/cluster-name/application`
4. 使用以下查询替换搜索查询表达式，然后选择运行查询

```
fields ispresent(kubernetes.pod_name) as haskubernetes_pod_name, stream,
kubernetes.pod_name, log |
filter haskubernetes_pod_name and kubernetes.pod_name like /cwagent-prometheus
```

您还可以确认 Prometheus 指标和元数据是否摄取作为 CloudWatch Logs 事件。

## 确认正在摄取 Prometheus 数据

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 CloudWatch Logs Insights。
3. 选择 `/aws/containerinsights/cluster-name/prometheus`
4. 使用以下查询替换搜索查询表达式，然后选择运行查询

```
fields @timestamp, @message | sort @timestamp desc | limit 20
```

## 录入丢弃的 Prometheus 指标

此版本不会收集直方图的 Prometheus 指标。您可以使用 CloudWatch 代理来检查是否因为任何 Prometheus 指标是直方图指标而丢弃了这些指标。您还可以记录由于是直方图指标，因此丢弃而未发送到 CloudWatch 的前 500 个 Prometheus 指标列表。

要查看是否丢弃了任何指标，请输入以下命令：

```
kubectll logs -l "app=cwagent-prometheus" -n amazon-cloudwatch --tail=-1
```

如果丢弃了任何指标，您将在 `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` 文件中看到以下行。

```
I! Drop Prometheus metrics with unsupported types. Only Gauge, Counter and Summary are
supported.
I! Please enable CWAgent debug mode to view the first 500 dropped metrics
```

如果您看到这些行并想知道丢弃了哪些指标，请使用以下步骤。

## 记录已丢弃的 Prometheus 指标列表

1. 通过将以下粗体行添加到 `prometheus-eks.yaml` 或 `prometheus-k8s.yaml` 文件中并保存文件，将 CloudWatch 代理更改为调试模式。

```
{
 "agent": {
 "debug": true
 },
}
```

该文件的这一部分随后应该如下所示：

```
cwagentconfig.json: |
{
 "agent": {
 "debug": true
 },
 "logs": {
 "metrics_collected": {
```

2. 通过输入以下命令重新安装 CloudWatch 代理以启用调试模式：

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
kubectl apply -f prometheus.yaml
```

丢弃的指标将在 CloudWatch 代理 pod 中录入。

3. 要从 CloudWatch 代理 pod 检索日志，请输入以下命令：

```
kubectl logs -l "app=cwagent-prometheus" -n amazon-cloudwatch --tail=-1
```

或者，如果您安装了 Container Insights FluentD 日志记录，日志也会保存在 CloudWatch Logs 日志组 / aws/containerinsights/**cluster\_name**/application 中。

要查询这些日志，您可以按照[Amazon EKS 上的一般故障排除步骤 \(p. 490\)](#)中的步骤查询应用程序日志。

### 在哪里摄取 Prometheus 指标作为 CloudWatch Logs 日志事件？

CloudWatch 代理为每个 Prometheus 抓取任务配置创建一个日志流。例如，在 prometheus-eks.yaml 和 prometheus-k8s.yaml 文件中，job\_name: 'kubernetes-pod-appmesh-envoy' 行将抓取 App Mesh 指标。Prometheus 目标被定义为 kubernetes-pod-appmesh-envoy。因此，所有 App Mesh Prometheus 指标都作为 CloudWatch Logs 事件摄取到名为 /aws/containerinsights/cluster-name/ Prometheus 的日志组下的 kubernetes-pod-appmesh-envoy 日志流中。

### 我在 CloudWatch 指标中没有看到 Amazon EKS 或 Kubernetes Prometheus 指标

首先，确保 Prometheus 指标在日志组 /aws/containerinsights/cluster-name/Prometheus 中摄取作为日志事件。使用[在哪里摄取 Prometheus 指标作为 CloudWatch Logs 日志事件？ \(p. 492\)](#)中的信息可帮助您检查目标日志流。如果未创建日志流或日志流中没有新的日志事件，请检查以下内容：

- 检查 Prometheus 指标导出程序端点是否正确设置
- 检查 CloudWatch 代理 YAML 文件 config map: cwagent-prometheus 部分中的 Prometheus 抓取配置是否正确。该配置应与 Prometheus 配置文件中的配置相同。有关更多信息，请参阅 Prometheus 文档中的 [<scrape\\_config>](#)。

如果正确摄取了 Prometheus 指标作为日志事件，请检查嵌入式指标格式设置是否已添加到日志事件以生成 CloudWatch 指标。

```
"CloudWatchMetrics": [
 {
 "Metrics": [
 {
 "Name": "envoy_http_downstream_cx_destroy_remote_active_rq"
 }
],
 "Dimensions": [
 [
 "ClusterName",
```



```
 "Namespace"
],
],
 "Namespace": "ContainerInsights/Prometheus"
 }
],
```

有关嵌入式指标格式的更多信息，请参阅[规范：嵌入式指标格式](#) (p. 755)。

如果日志事件中没有嵌入式指标格式，请检查在 CloudWatch 代理安装 YAML 文件的 config map: prometheus-cwagentconfig 部分是否正确配置了 metric\_declaration 部分。有关更多信息，请参阅[添加新 Prometheus 抓取目标的教程：Prometheus API 服务器指标](#) (p. 463)。

## 与 Application Insights 集成

Amazon CloudWatch Application Insights 可帮助您监控应用程序，并在应用程序资源和技术堆栈中指定和设置关键指标、日志和告警。有关更多信息，请参阅[Amazon CloudWatch Application Insights](#) (p. 508)。

您可以启用 Application Insights，从容器化应用程序和微服务中收集其他数据。如果您尚未执行此操作，可以通过在 Container Insights 控制面板中的性能视图下方选择 Auto-configure Application Insights (自动配置 Application Insights) 将其启用。

如果您已设置 CloudWatch Application Insights 来监控容器化应用程序，Application Insights 控制面板将显示在 Container Insights 控制面板下方。

有关 Application Insights 和容器化应用程序的更多信息，请参阅[为 Amazon ECS 和 Amazon EKS 资源监控启用 Application Insights](#) (p. 523)。

## 在 Container Insights 中查看 Amazon ECS 生命周期事件

您可以在 Container Insights 控制台中查看 Amazon ECS 生命周期事件。这有助于您在单个视图中关联容器指标、日志和事件，从而为您提供更完整的操作可见性。

事件包括容器实例状态更改事件、任务状态更改事件和服务操作事件。它们由 Amazon ECS 自动发送到 Amazon EventBridge，并且还以事件日志格式在 CloudWatch 中收集。有关这些事件的更多信息，请参阅[Amazon ECS 事件](#)。

生命周期事件不会产生额外费用。适用标准的 Container Insights 定价。有关更多信息，请参阅[Amazon CloudWatch 定价](#)。

要配置生命周期事件表并为集群创建规则，您必须拥有 events:PutRule、events:PutTargets 和 logs:CreateLogGroup 权限。

要查看生命周期事件表，您必须拥有 events:DescribeRule、events:ListTargetsByRule 和 logs:DescribeLogGroups 权限。

要在 CloudWatch Container Insights 控制台中查看 Amazon ECS 生命周期事件

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 选择 Insights、Container Insights。
3. 在页面顶部附近的下拉框中，选择 Performance monitoring (性能监控)。
4. 在下一个下拉列表中，选择 ECS Clusters (ECS 集群)、ECS Services (ECS 服务) 或 ECS Tasks (ECS 任务)。
5. 如果您在上一步中选择了 ECS Services (ECS 服务) 或 ECS Tasks (ECS 任务)，则选择 Lifecycle events (生命周期事件) 选项卡。

- 在页面底部，如果您看到 Configure lifecycle events（配置生命周期事件），选择它来为您的集群创建 EventBridge 规则。

事件显示在 Container Insights 窗格的下方和 Application Insights 部分的上方。要对这些事件进行额外分析并创建其他可视化效果，请在生命周期事件表中选择 View in Logs Insights（在 Logs Insights 中查看）。

## Container Insights 问题排查

如果在使用 Container Insights 时遇到问题，以下部分可为您提供帮助。

### 在 Amazon EKS 或 Kubernetes 上部署失败

如果未在 Kubernetes 集群上正确部署该代理，请尝试执行以下操作：

- 运行以下命令以获取 pod 列表。

```
kubectl get pods -n amazon-cloudwatch
```

- 运行以下命令并在输出底部检查事件。

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

- 运行以下命令以检查日志。

```
kubectl logs pod-name -n amazon-cloudwatch
```

### 未经授权的 panic：无法从 kubelet 检索 cadvisor 数据

如果您的部署失败，并显示错误 Unauthorized panic: Cannot retrieve cadvisor data from kubelet，则您的 kubelet 可能未启用 Webhook 授权模式。Container Insights 需要此模式。有关更多信息，请参阅[验证先决条件](#) (p. 368)。

### 在 Amazon ECS 上已删除和重新创建的集群上部署 Container Insights

如果删除未启用 Container Insights 的现有 Amazon ECS 集群，并使用相同名称重新创建它，则无法在重新创建此集群时在此新集群上启用 Container Insights。您可以通过重新创建来启用它，然后输入以下命令：

```
aws ecs update-cluster-settings --cluster myECScluster --settings
name=containerInsights,value=enabled
```

### 端点无效错误

如果您看到类似于以下内容的错误消息，请检查以确保已将正在使用的命令中的所有占位符（例如 *cluster-name* 和 *region-name*）替换为正确的部署信息。

```
"log": "2020-04-02T08:36:16Z E! cloudwatchlogs: code: InvalidEndpointURL, message:
invalid endpoint uri, original error: &url.Error{Op:\"parse\", URL:\"https://logs.
{region_name}.amazonaws.com/\", Err:\"\", &awserr.baseError{code:\"InvalidEndpointURL
\", message:\"invalid endpoint uri\", errs:[]error{(*url.Error)(0xc0008723c0)}}\\n",
```



## 指标未显示在控制台中

如果您在 Amazon Web Services Management Console 中未看到任何 Container Insights 指标，请确保已完成 Container Insights 的设置。在完全设置 Container Insights 之前，不会显示指标。有关更多信息，请参阅 [设置 Container Insights \(p. 355\)](#)。

## 升级集群后，Amazon EKS 或 Kubernetes 上缺少 Pod 指标

如果您将 CloudWatch 代理作为守护进程部署在新的或升级的集群上后丢失了全部或部分 pod 指标，或者您看到带有 W! No pod metric collected 消息的错误日志，则此节可能很有帮助。

这些错误可能是由容器运行时的更改引起的，例如 containerd 或 docker systemd cgroup 驱动程序。您通常可以通过更新部署清单来解决此问题，以便将主机的 containerd 套接字挂载到容器中。请参见以下示例：

```
For full example see https://github.com/aws-samples/amazon-cloudwatch-container-insights/
blob/master/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-
monitoring/cwagent/cwagent-daemonset.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: cloudwatch-agent
 namespace: amazon-cloudwatch
spec:
 template:
 spec:
 containers:
 - name: cloudwatch-agent
...
 # Don't change the mountPath
 volumeMounts:
...
 - name: dockersock
 mountPath: /var/run/docker.sock
 readOnly: true
 - name: varlibdocker
 mountPath: /var/lib/docker
 readOnly: true
 - name: containerdsock # NEW mount
 mountPath: /run/containerd/containerd.sock
 readOnly: true
...
 volumes:
...
 - name: dockersock
 hostPath:
 path: /var/run/docker.sock
 - name: varlibdocker
 hostPath:
 path: /var/lib/docker
 - name: containerdsock # NEW volume
 hostPath:
 path: /run/containerd/containerd.sock
```

## 使用 Bottlerocket for Amazon EKS 时没有 pod 指标

Bottlerocket 是一个基于 Linux 的开源操作系统，Amazon 专为运行容器而构建。

Bottlerocket 在主机上使用不同的 containerd 路径，因此您需要将卷更改为其位置。如果未这样设置，包含 W! No pod metric collected 的日志会显示错误。请参阅以下示例。

```
volumes:
```

```
...
- name: containerdsock
 hostPath:
 # path: /run/containerd/containerd.sock
 # bottlerocket does not mount containerd sock at normal place
 # https://github.com/bottlerocket-os/bottlerocket/
 commit/91810c85b83ff4c3660b496e243ef8b55df0973b
 path: /run/dockershim.sock
```

## 将 containerd 运行时用于 Amazon EKS 或 Kubernetes 时，没有容器文件系统指标

这是一个已知问题，正在由社群贡献者解决。有关更多信息，请参阅 GitHub 上的 [containerd 的磁盘使用指标](#) 和 [containerd 的 cadvisor 不支持容器文件系统指标](#)。

## 收集 Prometheus 指标时，CloudWatch 代理的日志卷意外增加

这是 CloudWatch 代理的 1.247347.6b250880 版本中推出的回归。此回归已在代理的更新版本中得到修复。它的影响仅限于客户收集 CloudWatch 代理本身的日志并且仍在使用 Prometheus 的情况。有关更多信息，请参阅 GitHub 上的 [\[prometheus\] 代理正在日志中打印所有收集的指标](#)。

## 未从 Dockerhub 找到发布说明中提到的最新 docker 镜像

在内部开始实际发布之前，我们会更新 Github 上的发布说明和标签。在 Github 上增加版本号后，通常需要 1–2 周才能在注册表上看到最新的 Docker 镜像。CloudWatch 代理容器镜像不会在夜间发布。您可以在以下位置直接从源代码构建镜像：<https://github.com/aws/amazon-cloudwatch-agent/tree/master/amazon-cloudwatch-container-insights/cloudwatch-agent-dockerfile>

## CloudWatch 代理上的 CrashLoopBackoff 错误

如果您看到 CloudWatch 代理出现 CrashLoopBackOff 错误，请确保您的 IAM 权限设置正确。有关更多信息，请参阅[验证先决条件](#) (p. 368)。

## CloudWatch 代理或 FluentD Pod 卡在待处理状态

如果您有一个 CloudWatch 代理或 FluentD pod 卡在 Pending 状态或出现 FailedScheduling 错误，请根据代理所需的内核数量和 RAM 量确定您的节点是否有足够的计算资源。使用以下命令描述此 pod：

```
kubectl describe pod cloudwatch-agent-85ppg -n amazon-cloudwatch
```

## 构建您自己的 CloudWatch 代理 Docker 镜像

您可以通过引用位于以下网址的 Dockerfile 来构建自己的 CloudWatch 代理 Docker 镜像：<https://github.com/aws-samples/amazon-cloudwatch-container-insights/blob/master/cloudwatch-agent-dockerfile/Dockerfile>。

Dockerfile 支持直接使用 `docker buildx` 构建多架构镜像。

## 在容器中部署其他 CloudWatch 代理功能

您可以使用 CloudWatch 代理在容器中部署其他监控功能。这些功能如下所示：

- 嵌入式指标格式 – 有关更多信息，请参阅 [采用 CloudWatch 嵌入式指标格式摄取高基数日志以生成指标](#) (p. 754)。
- StatsD – 有关更多信息，请参阅 [使用 StatsD 检索自定义指标](#) (p. 717)。

说明和必要的文件位于 GitHub 上的以下位置：

- 对于 Amazon ECS 容器，请参阅[基于部署模式的 Amazon ECS 任务定义示例](#)。
- 对于 Amazon EKS 和 Kubernetes 容器，请参阅[基于部署模式的 Kubernetes YAML 文件示例](#)。

## 使用 Contributor Insights 分析高基数数据

您可以使用 Contributor Insights 分析日志数据，并创建显示贡献者数据的时间序列。您可以查看有关前 N 个贡献者、独特贡献者总数及其使用情况的指标。这有助于您找到排名靠前的说话者，并了解影响系统性能的人员或内容。例如，您可以查找无效主机，确定最多的网络用户或查找产生最多错误的 URL。

您可以从 Scratch 构建规则，并且在使用 Amazon Web Services Management Console 时，还可以使用 Amazon 已创建的示例规则。规则定义要用于定义贡献者的日志字段，例如 `IpAddress`。还可以筛选日志数据来查找和分析各个贡献者的行为。

CloudWatch 还提供了内置规则，可用于分析来自其他 Amazon 服务的指标。

所有规则都将实时分析传入数据。

### Note

如果您使用 Contributor Insights，需要为每个与规则匹配的日志事件付费。有关更多信息，请参阅[Amazon CloudWatch 定价](#)。

### 主题

- [创建 Contributor Insights 规则 \(p. 497\)](#)
- [Contributor Insights 规则语法 \(p. 500\)](#)
- [Contributor Insights 规则示例 \(p. 502\)](#)
- [查看 Contributor Insights 报告 \(p. 505\)](#)
- [绘制规则生成的指标的图表 \(p. 505\)](#)
- [使用 Contributor Insights 内置规则 \(p. 507\)](#)

## 创建 Contributor Insights 规则

您可创建规则以分析日志数据。可以评估任何采用 JSON 或常用日志格式 (CLF) 的日志。这包括遵循这些格式之一的自定义日志以及来自 Amazon 服务的日志，例如 Amazon VPC 流日志、Amazon Route 53 DNS 查询日志、Amazon ECS 容器日志以及来自 Amazon CloudTrail、Amazon SageMaker、Amazon RDS、Amazon AppSync 和 API Gateway 的日志。

在规则中指定字段名称或值时，所有匹配都区分大小写。

在创建规则时，您可以使用内置示例规则，也可以从 Scratch 创建自己的规则。Contributor Insights 包括以下日志类型的示例规则：

- Amazon API Gateway 日志
- Amazon Route 53 公有 DNS 查询日志
- Amazon Route 53 resolver 查询日志
- CloudWatch Container Insights 日志
- VPC 流日志

### Important

当您向用户授予 `cloudwatch:PutInsightRule` 权限时，默认情况下，该用户可以创建一个规则来评估 CloudWatch Logs 中的任何日志组。您可以添加 IAM policy 条件，以限制用户的这些权限，

使其包含和排除特定的日志组。有关更多信息，请参阅[使用条件键限制 Contributor Insights 用户对日志组的访问](#) (p. 862)。

要使用内置示例规则创建规则，请执行以下操作：

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Contributor Insights。
3. 请选择 Create rule (创建规则)。
4. 选择 Sample rule (示例规则)，然后从 Select sample rule (选择示例规则) 中选择规则。
5. 为 Rule name (规则名称) 输入一个名称。有效字符包括 A-Z、a-z、0-9、( 连字符 )、( 下划线 ) 和 ( 半角句点 )。
6. 对于 Log group(s) (日志组)，选择您希望您的规则要监控的日志组。您可以选择多达 20 个日志组。
  - ( 可选 ) 要选择名称以特定字符串开头的所有日志组，请选择 Select by prefix match ( 按前缀匹配选择 ) 下拉菜单，然后输入前缀。

#### Note

您需要为与您的规则匹配的每个日志事件支付费用。如果您选择 Select by prefix match ( 按前缀匹配选择 ) 下拉菜单，请注意前缀可以匹配多少个日志组。如果您搜索了超出预期数量的日志组，可能会产生意外费用。有关更多信息，请参阅 [Amazon CloudWatch 定价](#)。

7. 示例规则已填写 Fields ( 字段 )、Contribution ( 贡献 )、Filters ( 筛选条件 ) 和 Aggregate on ( 聚合 ) 字段。您可以根据需要对这些值进行调整。
8. 选择是创建处于已禁用状态还是已启用状态的规则。如果您选择启用规则，它将立即开始对您的数据进行分析。运行启用的规则时，您需要支付费用。有关更多信息，请参阅 [Amazon CloudWatch 定价](#)。

Contributor Insights 仅在创建规则后分析新的日志事件。规则无法处理之前已由 CloudWatch Logs 处理的日志事件。

9. 选择 Create (创建)。

要从 Scratch 创建规则，请执行以下操作：

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Contributor Insights。
3. 请选择 Create rule (创建规则)。
4. 在 Create rule (创建规则) 向导中，选择 Custom rule (自定义规则)。
5. 为 Rule name (规则名称) 输入一个名称。有效字符为 A-Z、a-z、0-9、“-”、“\_”和“.”。
6. 您可以通过以下方式完成规则的创建过程：使用向导，或选择 Syntax (语法) 选项卡并手动指定规则语法。

要继续使用此向导，请执行以下操作：

- a. 对于 Log group(s) (日志组)，选择您希望规则监控的日志组。您可以选择多达 20 个日志组。

要选择名称以特定字符串开头的所有日志组，请选择 Select by prefix match (按前缀匹配选择) 并输入前缀。

如果使用 Select by prefix match (按前缀匹配选择)，您需要对有多少日志组将与您的前缀匹配并由规则进行分析心中有数。您需要为与规则匹配的每个日志事件支付费用。如果不小心搜索了超出预期数量的日志组，可能会产生意外费用。有关更多信息，请参阅 [Amazon CloudWatch 定价](#)。

- b. 对于 Log format (日志格式)，请选择 JSON 或 CLF。

- c. 对于 Contribution (贡献)、Key (键), 请输入要报告的贡献者类型。此报告将显示该贡献者类型的前 N 个值。

有效条目为任何具有值的日志字段。示例包括 **requestId**、**sourceIPAddress** 和 **containerID**。

有关查找特定日志组中日志的日志字段名称的信息, 请参阅 [Finding Log Fields \(p. 499\)](#)。

大于 1 KB 的键将被截断至 1KB。

- d. (可选) 为 Contribution (贡献)、Key (键) 添加多个值。可以在规则中包含最多四个键。如果输入多个键, 则报告中的贡献者将由键的唯一值组合定义。例如, 如果您指定三个键, 则三个键的每个唯一值组合将被计为一个独特贡献者。
- e. (可选) 如果要按数值日志字段的值 (而不是日志事件中的出现次数) 对贡献者进行排名, 请使用 Contribution (贡献)、Value (值) 字段。输入要进行求和以确定贡献者排名的数值日志字段的名称。例如, 如果要查找通过网络发送最多字节的源 IP 地址, 则可以添加 **bytes** 作为值, 并假定 **bytes** 是日志事件中该字段的正确关键字。
- f. (可选) 如果要添加筛选条件以缩小结果范围, 请选择 Add filter (添加筛选条件)。对于 Match (匹配), 输入要作为筛选条件的日志字段的名称。对于 Condition (条件), 选择比较运算符, 并输入要作为筛选条件的值。

您可以在规则中最多添加 4 个筛选条件。多个筛选条件通过 AND 逻辑联接, 因此, 仅返回与所有筛选条件都匹配的日志事件。

#### Note

遵循比较运算符的数组 (例如 In、NotIn 或者 StartsWith) 可以包含最多 10 个字符串值。有关 Contributor Insights 规则语法的更多信息, 请参阅 [Contributor Insights 规则语法 \(p. 500\)](#)。

- g. 对于 Aggregate on (聚合), 选择 Count (计数) 或 Sum (总计)。选择 Count (计数) 会使贡献者排名基于出现次数。选择 Sum (总计) 会使排名基于您为 Contribution (贡献)、Value (值) 指定的字段值的总和。
- 7. 要输入规则作为 JSON 对象而不是使用此向导, 请执行以下操作:
    - a. 选择 Syntax (语法) 选项卡。
    - b. 在 Rule body (规则正文) 中, 输入规则的 JSON 对象。有关规则语法的信息, 请参阅 [Contributor Insights 规则语法 \(p. 500\)](#)。
  - 8. 选择是创建处于已禁用状态还是已启用状态的规则。如果您选择启用规则, 它将立即开始对您的数据进行分析。运行启用的规则时, 您需要支付费用。有关更多信息, 请参阅 [Amazon CloudWatch 定价](#)。

Contributor Insights 仅在创建规则后分析新的日志事件。规则无法处理之前已由 CloudWatch Logs 处理的日志事件。

- 9. 选择 Create (创建)。

可以禁用、启用或删除已创建的规则。

在 Contributor Insights 中启用、禁用或删除规则

- 1. 通过以下网址打开 CloudWatch 控制台: <https://console.aws.amazon.com/cloudwatch/>。
- 2. 在导航窗格中, 选择 Contributor Insights。
- 3. 在规则列表中, 选中单个规则旁边的复选框。

内置规则由 Amazon 服务创建, 无法编辑、禁用或删除这些规则。

- 4. 选择 Actions (操作), 然后选择所需选项。

### 查找日志字段

创建规则时，您需要知道日志组的日志条目中的字段名称。

### 查找日志组中的日志字段

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中的 Logs (日志) 下，选择 Insights (见解)。
3. 在查询编辑器上方，选择要查询的一个或多个日志组。

当您选择日志组时，CloudWatch Logs Insights 会自动检测日志组数据中的字段，并将其显示在右侧窗格中的 Discovered fields (发现的字段) 中。

## Contributor Insights 规则语法

此部分说明了 Contributor Insights 规则的语法。仅当您通过输入 JSON 块来创建规则时使用此语法。如果您使用此向导创建规则，则无需知道语法。有关使用此向导创建规则的更多信息，请参阅 [创建 Contributor Insights 规则 \(p. 497\)](#)。

日志事件字段名称和值的所有规则匹配都区分大小写。

以下示例说明了 JSON 日志的语法。

```
{
 "Schema": {
 "Name": "CloudWatchLogRule",
 "Version": 1
 },
 "LogGroupNames": [
 "API-Gateway-Access-Logs*",
 "Log-group-name2"
],
 "LogFormat": "JSON",
 "Contribution": {
 "Keys": [
 "$.ip"
],
 "ValueOf": "$.requestBytes",
 "Filters": [
 {
 "Match": "$.httpMethod",
 "In": [
 "PUT"
]
 }
]
 },
 "AggregateOn": "Sum"
}
```

### Contributor Insights 规则中的字段

#### 架构

用于分析 CloudWatch Logs 数据 Schema 的规则的值必须始终为 {"Name": "CloudWatchLogRule", "Version": 1}

#### LogGroupNames

字符串数组。对于数组中的每个元素，您可以选择在字符串末尾使用 \* 以包含名称以该前缀开头的所有日志组。



将通配符用于日志组名称时要小心。您需要为与规则匹配的每个日志事件支付费用。如果不小心搜索了超出预期数量的日志组，可能会产生意外费用。有关更多信息，请参阅 [Amazon CloudWatch 定价](#)。

#### LogFormat

有效值为 JSON 和 CLF。

#### 贡献

此对象包含一个具有最多四个成员的 Keys 数组、（可选）一个 ValueOf、（可选）一个具有最多四个 Filters 的数组。

#### 密钥

一个最多具有四个日志字段的数组，这些字段用作维度以对贡献者进行分类。如果您输入多个键，则这些键的每个唯一值组合将被计为一个独特贡献者。必须使用 JSON 属性格式表示法指定字段。

#### ValueOf

（可选）仅当您指定 Sum 作为 AggregateOn 的值时指定此选项。ValueOf 指定具有数值的日志字段。在此类规则中，贡献者按其在此字段中的值的和来进行排名，而不是按其在此日志条目中的出现次数进行排名。例如，如果您希望按贡献者在一段时间内的 BytesSent 总数对其进行排名，则需要将 ValueOf 设置为 BytesSent，并为 AggregateOn 指定 Sum。

#### Filters

（可选）指定具有最多四个筛选条件的数组可缩小报告中包含的日志事件的范围。如果您指定多个筛选条件，则 Contributor Insights 会使用逻辑 AND 运算符对它们进行评估。您可以使用此选项来筛掉搜索中不相关的日志事件，也可以使用它选择单个贡献者来分析其行为。

数组中的每个成员必须包含一个 Match 字段和一个指示要使用的匹配运算符类型的字段。

Match 字段指定要在筛选条件中评估的日志字段。使用 JSON 属性格式表示法指定日志字段。

匹配的运算符字段必须是下列项目之一：

— In、NotIn、StartsWith、GreaterThan、LessThan、EqualTo、NotEqualTo 或 IsPresent。如果运算符字段为 In、NotIn 或 StartsWith，则它后跟一个要检查的字符串值数组。Contributor Insights 使用 OR 运算符评估字符串值的数组。该数组可以包含最多 10 个字符串值。

如果运算符字段为 GreaterThan、LessThan、EqualTo 或 NotEqualTo，则它后跟要比较的单个数值。

如果运算符字段为 IsPresent，则它后跟 true 或 false。此运算符将根据日志事件中是否存在指定的日志字段来匹配日志事件。isPresent 仅使用 JSON 属性的叶节点中的值。例如，查找 c-count 的匹配项的筛选条件不会返回具有值 details.c-count.c1 的日志事件。

有关四个筛选条件示例，请参阅以下内容：

```
{ "Match": "$.httpMethod", "In": ["PUT",] }
{ "Match": "$.StatusCode", "EqualTo": 200 }
{ "Match": "$.BytesReceived", "GreaterThan": 10000 }
{ "Match": "$.eventSource", "StartsWith": ["ec2", "ecs"] }
```

#### AggregateOn

有效值为 Count 和 Sum。指定是基于出现次数，还是基于 ValueOf 字段中指定的字段值的总和来聚合报告。

#### JSON 属性格式表示法

Keys、ValueOf 和 Match 字段遵循带点符号的 JSON 属性格式，其中 \$ 表示 JSON 对象的根。这之后是句点，然后是带子属性名称的字母数字字符串。支持多个属性级别。

以下列表显示了 JSON 属性格式的有效示例：

```
$.userAgent
$.endpoints[0]
$.users[1].name
$.requestParameters.instanceId
```

#### CLF 日志规则中的其他字段

常用日志格式 (CLF) 日志事件不像 JSON 那样具有字段的名称。要提供用于 Contributor Insights 规则的字段，CLF 日志事件可被视为具有从 1 开始的索引的数组。您可以将第一个字段指定为 "1"，将第二个字段指定为 "2"，以此类推。

要使 CLF 日志的规则更易于读取，可以使用 Fields。这可让您为 CLF 字段位置提供命名别名。例如，您可以指定位置"4"为 IP 地址。指定后，IpAddress 可用作规则中的 Keys、ValueOf 和 Filters 中的属性。

以下是使用 Fields 字段的 CLF 日志规则的示例。

```
{
 "Schema": {
 "Name": "CloudWatchLogRule",
 "Version": 1
 },
 "LogGroupNames": [
 "API-Gateway-Access-Logs*"
],
 "LogFormat": "CLF",
 "Fields": {
 "4": "IpAddress",
 "7": "StatusCode"
 },
 "Contribution": {
 "Keys": [
 "IpAddress"
],
 "Filters": [
 {
 "Match": "StatusCode",
 "EqualTo": 200
 }
]
 },
 "AggregateOn": "Count"
}
```

## Contributor Insights 规则示例

此部分包含的示例说明了 Contributor Insights 规则的使用案例。

VPC 流日志：按源和目标 IP 地址进行的字节传输

```
{
 "Schema": {
 "Name": "CloudWatchLogRule",
 "Version": 1
 },
 "LogGroupNames": [
 "/aws/containerinsights/sample-cluster-name/flowlogs"
],
 "LogFormat": "CLF",
 "Fields": {
 "4": "srcaddr",
```



```

 "5": "dstaddr",
 "10": "bytes"
 },
 "Contribution": {
 "Keys": [
 "srcaddr",
 "dstaddr"
],
 "ValueOf": "bytes",
 "Filters": []
 },
 "AggregateOn": "Sum"
}

```

#### VPC 流日志：最大 HTTPS 请求数

```

{
 "Schema": {
 "Name": "CloudWatchLogRule",
 "Version": 1
 },
 "LogGroupNames": [
 "/aws/containerinsights/sample-cluster-name/flowlogs"
],
 "LogFormat": "CLF",
 "Fields": {
 "5": "destination address",
 "7": "destination port",
 "9": "packet count"
 },
 "Contribution": {
 "Keys": [
 "destination address"
],
 "ValueOf": "packet count",
 "Filters": [
 {
 "Match": "destination port",
 "EqualTo": 443
 }
]
 },
 "AggregateOn": "Sum"
}

```

#### VPC 流日志：被拒绝的 TCP 连接数

```

{
 "Schema": {
 "Name": "CloudWatchLogRule",
 "Version": 1
 },
 "LogGroupNames": [
 "/aws/containerinsights/sample-cluster-name/flowlogs"
],
 "LogFormat": "CLF",
 "Fields": {
 "3": "interfaceID",
 "4": "sourceAddress",
 "8": "protocol",
 "13": "action"
 },
 "Contribution": {
 "Keys": [

```

```
 "interfaceID",
 "sourceAddress"
],
 "Filters": [
 {
 "Match": "protocol",
 "EqualTo": 6
 },
 {
 "Match": "action",
 "In": [
 "REJECT"
]
 }
]
},
"AggregateOn": "Sum"
}
```

#### Route 53 NXDomain 按源地址响应

```
{
 "Schema": {
 "Name": "CloudWatchLogRule",
 "Version": 1
 },
 "AggregateOn": "Count",
 "Contribution": {
 "Filters": [
 {
 "Match": "$.rcode",
 "StartsWith": [
 "NXDOMAIN"
]
 }
],
 "Keys": [
 "$.srcaddr"
]
 },
 "LogFormat": "JSON",
 "LogGroupNames": [
 "<loggroupname>"
]
}
```

#### Route 53 Resolver 按域名查询

```
{
 "Schema": {
 "Name": "CloudWatchLogRule",
 "Version": 1
 },
 "AggregateOn": "Count",
 "Contribution": {
 "Filters": [],
 "Keys": [
 "$.query_name"
]
 },
 "LogFormat": "JSON",
 "LogGroupNames": [
 "<loggroupname>"
]
}
```

```
}
```

Route 53 Resolver 按查询类型和源地址查询

```
{
 "Schema": {
 "Name": "CloudWatchLogRule",
 "Version": 1
 },
 "AggregateOn": "Count",
 "Contribution": {
 "Filters": [],
 "Keys": [
 "$.query_type",
 "$.srcaddr"
]
 },
 "LogFormat": "JSON",
 "LogGroupNames": [
 "<loggroupname>"
]
}
```

## 查看 Contributor Insights 报告

要查看报告数据的图表和您的规则找到的贡献者排名列表，请执行以下步骤。

### 查看规则报告

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Contributor Insights。
3. 在该规则列表中，选择规则的名称。

图表显示了过去三小时内的规则结果。图表下方的表显示了前 10 名贡献者。

4. 要更改表中显示的贡献者数，请选择图表顶部的 Top 10 contributors (前 10 名贡献者)。
5. 要筛选图表以仅显示单个贡献者的结果，请在表图例中选择该贡献者。要再次显示所有贡献者，请在图例中再次选择同一个贡献者。
6. 要更改报告中显示的时间范围，请选择图表顶部的 15m (15 分钟)、30m (30 分钟)、1h (1 小时)、2h (2 小时)、3h (3 小时) 或 custom (自定义)。

报告的最大时间范围为 24 小时，但您可以选择最多 15 天前发生的 24 小时时段。要选择过去的时段，请选择 custom (自定义)、absolute (绝对)，然后指定您的时段。

7. 要更改用于聚合和贡献者排名的时段的长度，请选择图表顶部的 period (周期)。查看较长的时段通常会显示更平滑的报告，其中包含的峰值很少。选择较短的时段更有可能显示峰值。
8. 要将此图表添加到 CloudWatch 控制面板，请选择 Add to dashboard (添加到控制面板)。
9. 要打开 CloudWatch Logs Insights 查询窗口 (此报告中的日志组已加载到查询框中)，请选择 View logs (查看日志)。
10. 要将报告数据导出到剪贴板或 CSV 文件，请选择 Export (导出)。

## 绘制规则生成的指标的图表

Contributor Insights 提供了一个指标数学函数，即 `INSIGHT_RULE_METRIC`。您可以使用此函数将 Contributor Insights 报告中的数据添加到 CloudWatch 控制台的 Metrics (指标) 选项卡中的图表。您也可以根据此数学函数来设置警报。有关指标数学函数的更多信息，请参阅 [使用指标数学 \(p. 116\)](#)

要使用此指标数学函数，您必须登录到同时具有 `cloudwatch:GetMetricData` 和 `cloudwatch:GetInsightRuleReport` 权限的账户。

语法为 `INSIGHT_RULE_METRIC(ruleName, metricName)`。*ruleName* 是 Contributor Insights 规则的名称，*metricName* 是以下列表中的值之一。*metricName* 的值决定了数学函数返回的数据类型。

- `UniqueContributors` – 每个数据点的独特贡献者数。
- `MaxContributorValue` – 每个数据点的顶级贡献者的值。对于图表中的每个数据点，确定的贡献者可能会变化。

如果此规则按 `Count` 进行聚合，则每个数据点的顶级贡献者是该时段内出现次数最多的贡献者。如果此规则按 `Sum` 进行聚合，则排在最前面的贡献者是此时段内具有该规则的 `value` 指定的日志字段中最大总和的贡献者。

- `SampleCount` – 规则匹配的数据点数。
- `Sum` – 该数据点表示的时段内来自所有贡献者的值的总和。
- `Minimum` – 该数据点所表示的时段内单个观察结果中的最小值。
- `Maximum` – 该数据点所表示的时段内单个观察结果中的最大值。
- `Average` – 该数据点表示的时段内来自所有贡献者的平均值。

## 为 Contributor Insights 指标数据设置告警

您可以使用函数 `INSIGHT_RULE_METRIC` 对 Contributor Insights 生成的指标设置告警。例如，您可以根据已被拒绝的传输控制协议 (TCP) 连接的百分比来创建告警。要开始使用这种类型的告警，您可以创建类似以下两个示例中所显示的规则：

示例规则: "RejectedConnectionsRule"

```
{
 "Schema": {
 "Name": "CloudWatchLogRule",
 "Version": 1
 },
 "LogGroupNames": [
 "/aws/containerinsights/sample-cluster-name/flowlogs"
],
 "LogFormat": "CLF",
 "Fields": {
 "3": "interfaceID",
 "4": "sourceAddress",
 "8": "protocol",
 "13": "action"
 },
 "Contribution": {
 "Keys": [
 "interfaceID",
 "sourceAddress"
],
 "Filters": [
 {
 "Match": "protocol",
 "EqualTo": 6
 },
 {
 "Match": "action",
 "In": [
 "REJECT"
]
 }
]
 }
}
```

```
],
 },
 "AggregateOn": "Sum"
 }
}
```

示例规则: "TotalConnectionsRule"

```
{
 "Schema": {
 "Name": "CloudWatchLogRule",
 "Version": 1
 },
 "LogGroupNames": [
 "/aws/containerinsights/sample-cluster-name/flowlogs"
],
 "LogFormat": "CLF",
 "Fields": {
 "3": "interfaceID",
 "4": "sourceAddress",
 "8": "protocol",
 "13": "action"
 },
 "Contribution": {
 "Keys": [
 "interfaceID",
 "sourceAddress"
],
 "Filters": [
 {
 "Match": "protocol",
 "EqualTo": 6
 }
],
 "AggregateOn": "Sum"
 }
}
```

创建规则后，您可以选择 CloudWatch 控制台中的 Metrics ( 指标 ) 选项卡，在其中使用以下示例指标数学表达式来绘制 Contributor Insights 报告的数据的图表：

示例: 指标数学表达式

```
e1 INSIGHT_RULE_METRIC("RejectedConnectionsRule", "Sum")
e2 INSIGHT_RULE_METRIC("TotalConnectionsRule", "Sum")
e3 (e1/e2)*100
```

在该示例中，指标数学表达式 e3 返回所有已被拒绝的 TCP 连接。如果您想要在已被拒绝的 TCP 连接达到 20% 时收到通知，则可以修改表达式，将阈值从 100 更改为 20。

#### Note

在 Metrics ( 指标 ) 部分中，您可以对您监控的指标设置告警。在 Graphed metrics ( 绘制的指标 ) 选项卡中，您可以选择 Actions ( 操作 ) 列下的 Create alarm ( 创建告警 )。Create alarm ( 创建告警 ) 图标看起来像个铃铛。

有关绘制指标的图表和使用指标数学函数的更多信息，请参阅以下部分: [向 CloudWatch 图表中添加数学表达式 \(p. 116\)](#)。

## 使用 Contributor Insights 内置规则

您可以使用 Contributor Insights 内置规则来分析其他 Amazon 服务的指标。以下服务支持内置规则：

- Amazon DynamoDB 开发人员指南中的 [Amazon DynamoDB 的 Contributor Insights](#)。

- Amazon PrivateLink 指南中的 [使用内置的 Contributor Insights 规则](#)。

## Amazon CloudWatch Application Insights

Amazon CloudWatch Application Insights 简化了观察您的应用程序和基础 Amazon 资源的过程。它可用帮助您为应用程序资源设置最佳的监控，以持续分析数据，以便找出应用程序出现问题的迹象。Application Insights 由 [Sagemaker](#) 和其他 Amazon 技术提供支持，它提供了自动化控制面板以显示监控的应用程序的潜在问题，从而帮助您快速找出应用程序和基础设施当前存在的问题。Application Insights 提高了应用程序运行状况的可见性，可以帮助您缩短平均修复时间 (MTTR)，以便对应用程序进行故障排除。

在将应用程序添加到 Amazon CloudWatch Application Insights 时，它扫描应用程序中的资源，并在 [CloudWatch](#) 上为应用程序组件建议和配置指标和日志。示例应用程序组件可能包括 SQL Server 后端数据库和 Microsoft IIS/Web 层。Application Insights 使用历史数据分析指标模式以检测异常情况，并从应用程序、操作系统和基础设施日志中持续检测错误和异常情况。它使用分类算法和内置规则组合以关联这些观察结果。然后，它自动创建控制面板以显示相关的观察结果和问题严重性信息，以帮助确定操作的优先级。对于 .NET 和 SQL 应用程序堆栈中的常见问题（例如，应用程序延迟、SQL Server 备份失败、内存泄漏、过大的 HTTP 请求以及输入/输出操作取消），它提供了额外的洞察，以指出可能的根本原因和解决步骤。与 [Amazon SSM OpsCenter](#) 进行内置集成后，您可以通过运行相关的 Systems Manager Automation 文档来解决问题。

### 小节目录

- [Amazon CloudWatch Application Insights 是什么？ \(p. 508\)](#)
- [Amazon CloudWatch Application Insights 的工作方式 \(p. 512\)](#)
- [开始使用 Amazon CloudWatch Application Insights \(p. 518\)](#)
- [使用组件配置 \(p. 536\)](#)
- [使用 CloudFormation 模板创建和配置 CloudWatch Application Insights 监控 \(p. 572\)](#)
- [教程：为 SAP HANA 设置监控 \(p. 581\)](#)
- [教程：使用 SQL Server 为 .NET 应用程序设置监控 \(p. 589\)](#)
- [查看和排查 Amazon CloudWatch Application Insights 检测到的问题 \(p. 594\)](#)
- [Amazon CloudWatch Application Insights 支持的日志和指标 \(p. 595\)](#)

## Amazon CloudWatch Application Insights 是什么？

CloudWatch Application Insights 可帮助您监控使用 Amazon EC2 实例以及其他 [应用程序资源 \(p. 511\)](#) 的应用程序。它可在应用程序资源和技术堆栈（例如，Microsoft SQL Server 数据库、Web (IIS) 和应用程序服务器、操作系统、负载均衡器和队列）中识别和设置关键指标、日志和警报。它会持续监控指标和日志，以检测异常情况和错误并将它们关联起来。在检测到错误和异常情况时，Application Insights 生成 [CloudWatch Events](#)，您可以使用这些事件来设置通知或执行操作。为了帮助进行故障排除，它会为检测到的问题创建自动化控制面板，其中包含关联的指标异常情况和日志错误，以及可指出潜在根本原因的其他洞察。自动化控制面板可帮助您快速采取修复操作，以保持应用程序正常运行，并防止对应用程序的终端用户造成影响。此外，它还会创建 OpsItems，以便您可以使用 [Amazon SSM OpsCenter](#) 解决问题。

您可以在 CloudWatch 上配置重要的计数器，例如镜像写入事务数/秒、恢复队列长度、事务延迟以及 Windows 事件日志。当 SQL HA 工作负载发生故障转移事件或问题（例如，查询目标数据库的访问权限受限）时，CloudWatch Application Insights 会提供自动化洞察。

CloudWatch Application Insights 与 [Amazon Launch Wizard](#) 集成，从而为在 Amazon 上部署 SQL Server HA 工作负载提供一键式监控设置体验。当您在 [Launch Wizard 控制台](#) 上选择使用 Application Insights 设置监控和洞察的选项时，CloudWatch Application Insights 会自动在 CloudWatch 上设置相关指标、日志和告警，并开始监控新部署的工作负载。您可以在 CloudWatch 控制台上查看自动化的洞察和检测到的问题，以及 SQL Server HA 工作负载的运行状况。

## 目录

- [功能 \(p. 509\)](#)
- [概念 \(p. 509\)](#)
- [定价 \(p. 509\)](#)
- [相关服务 \(p. 510\)](#)
- [支持的应用程序组件 \(p. 511\)](#)
- [支持的技术堆栈 \(p. 512\)](#)

## 功能

Application Insights 提供以下功能。

### 自动为应用程序资源设置监视器

CloudWatch Application Insights 可减少为应用程序设置监控所需的时间。它扫描应用程序资源，提供可自定义的建议指标和日志列表，并在 CloudWatch 上设置，从而为所需的应用程序资源（例如 Amazon EC2 和 Elastic Load Balancer (ELB)）提供可见性，从而减少为应用程序设置监控所需的时间。它还会为监控的指标设置动态警报。将根据前两周检测到的异常情况自动更新警报。

### 问题检测和通知

CloudWatch Application Insights 检测应用程序出现潜在问题的迹象，例如指标异常情况和日志错误。它关联这些观察结果以找出应用程序的潜在问题。然后，它生成 CloudWatch Events，[可以配置这些事件以接收通知或执行操作 \(p. 536\)](#)。这样，您无需为指标或日志错误创建单独的警报。

### 问题排查

CloudWatch Application Insights 为检测到的问题创建 CloudWatch 自动化控制面板。这些控制面板显示有关问题的详细信息（包括关联的指标异常情况和日志错误）以帮助进行故障排除。它们还提供额外的信息，以指出异常情况和错误的潜在根本原因。

## 概念

要了解 Application Insights 如何监控应用程序，以下概念是非常重要的。

### 组件

对组成应用程序的类似资源进行自动分组、单独分组或自定义分组。我们建议将类似资源分组到自定义组件中，以更好地进行监控。

### 观察

为应用程序或应用程序资源检测到的单个事件（指标异常情况、日志错误或异常）。

### 问题

关联、分类和分组相关的观察结果以检测问题。

有关 CloudWatch Application Insights 的其他重要概念的定义，请参阅 [Amazon CloudWatch 概念](#)。

## 定价

CloudWatch Application Insights 使用 CloudWatch Metrics 和 Logs 为选定的应用程序资源设置建议的指标和日志，并使用 CloudWatch Events 对检测到的问题进行通知。将根据 [CloudWatch 定价](#) 对您的 Amazon 账



户收取这些功能的费用。对于检测到的问题，它创建 [CloudWatch Events](#) 和 [自动控制面板](#)。不会向您收取设置帮助、监控数据分析或问题检测费用。

## 相关服务

以下服务与 CloudWatch Application Insights 一起使用：

### 相关 Amazon 服务

- Amazon CloudWatch 在系统范围内提供资源使用率、应用程序性能和运行状况信息。它收集并跟踪指标，发送警报通知，根据您定义的规则自动更新监控的资源，并允许您监控自己的自定义指标。CloudWatch Application Insights 是通过 CloudWatch 启动的，具体来说，是在 CloudWatch 默认运行控制面板中启动的。有关更多信息，请参阅 [Amazon CloudWatch 用户指南](#)。
- CloudWatch Container Insights 从容器化应用程序和微服务中收集、聚合及汇总指标与日志。您可以使用 Container Insights 来监控 Amazon ECS、Amazon Elastic Kubernetes Service 以及 Amazon EC2 上的 Kubernetes 平台。在 Container Insights 或 Application Insights 控制台中启用 Application Insights 后，Application Insights 会将检测到的问题显示在 Container Insights 控制面板上。有关更多信息，请参阅 [使用 Container Insights \(p. 353\)](#)。
- Amazon DynamoDB 是完全托管式 NoSQL 数据库服务器，您可以用它免除操作和扩展分布式数据库的管理工作负担，因而无需担心硬件预置、设置和配置、复制、软件修补或集群扩展等问题。此外，DynamoDB 提供了加密静态，这可以消除在保护敏感数据时涉及的操作负担和复杂性。
- Amazon EC2 在 Amazon 云中提供可扩展的计算容量。您可以使用 Amazon EC2 启动所需数量的虚拟服务器，配置安全性和联网，以及管理存储。您可以扩展或缩减以处理需求变化或使用高峰，从而减少预测流量的需求。有关更多信息，请参阅 [适用于 Linux 实例的 Amazon EC2 用户指南](#) 或 [适用于 Windows 实例的 Amazon EC2 指南](#)。
- Amazon Elastic Block Store (Amazon EBS) 提供了数据块级的存储卷以用于 Amazon EC2 实例。Amazon EBS 卷的行为类似于原始、未格式化的块储存设备。您可以将这些卷作为设备挂载在实例上。附加到实例的 Amazon EBS 卷公开为独立于实例生命周期而持续存在的存储卷。您可以在这些卷上创建文件系统，或者以使用块储存设备（如硬盘）的任何方式使用这些卷。您可以动态更改附加到实例的卷的配置。有关更多信息，请参阅 [Amazon EBS 用户指南](#)。
- Amazon EC2 Auto Scaling 帮助确保您具有正确数量的 EC2 实例以处理应用程序负载。有关更多信息，请参阅 [Amazon EC2 Auto Scaling 用户指南](#)。
- Elastic Load Balancing 在多个可用区中的多个目标（如 EC2 实例、容器和 IP 地址）之间分配传入的应用程序或网络流量。有关更多信息，请参阅 [Elastic Load Balancing 用户指南](#)。
- IAM 是一项 Web 服务，可帮助您安全地控制用户对 Amazon 资源的访问权限。可以通过 IAM 控制哪些用户可以使用您的 Amazon 资源（身份验证），以及控制他们可以使用的资源和使用资源的方式（授权）。有关更多信息，请参阅 [Amazon CloudWatch 的身份验证和访问控制](#)。
- 您可以通过 Amazon Lambda 构建由事件触发的函数组成的无服务器应用程序，并使用 CodePipeline 和 Amazon CodeBuild 自动部署这些应用程序。有关更多信息，请参阅 [Amazon Lambda 应用程序](#)。
- Amazon Launch Wizard for SQL Server 缩短了将 SQL Server 高可用性解决方案部署到云所需的时间。您可以在服务控制台上输入应用程序要求，包括性能、节点数量和连接性，并且，Amazon Launch Wizard 可标识正确的 Amazon 资源来部署和运行 SQL Server Always On 应用程序。
- Amazon Resource Groups 帮助您划分组成应用程序的资源。通过使用 Resource Groups，您可以同时管理和自动完成针对大量资源的任务。只能为单个应用程序注册一个资源组。有关更多信息，请参阅 [Amazon Resource Groups 用户指南](#)。
- Amazon SQS 提供了一个安全、持久且可用的托管队列，以允许您集成和分离分布式软件系统和组件。有关更多信息，请参阅 [Amazon SQS 用户指南](#)。
- Amazon Step Functions 是一个无服务器函数构建工具，它允许您对各种 Amazon 服务和资源（包括 Amazon Lambda 函数）排序为结构化的可视化工作流。有关更多信息，请参阅 [Amazon Step Functions 用户指南](#)。
- Amazon SSM OpsCenter 跨服务聚合并标准化 OpsItem，同时提供关于每个 OpsItem、相关 OpsItem 和相关资源的上下文调查数据。OpsCenter 还提供 Systems Manager Automation 文档 (runbook)，让您可



以快速解决问题。您可以为每个 OpsItem 指定可搜索的自定义数据。您还可以按状态和源查看自动生成的 OpsItem 相关摘要报告。有关更多信息，请参阅 [Amazon Systems Manager 用户指南](#)。

- Amazon API Gateway 是一项 Amazon 服务，用于创建、发布、维护、监控和保护任意规模的 REST、HTTP 和 WebSocket API。API 开发人员可以创建能够访问 Amazon 或其他 Web 服务以及存储在 Amazon 云中的数据的数据的 API。有关更多信息，请参阅 [Amazon API Gateway 用户指南](#)。

#### Note

Application Insights 仅支持 REST API 协议（v1 的 API Gateway 服务）。

- Amazon Elastic Container Service (Amazon ECS) 是一项完全托管式的容器编排服务。您可以使用 Amazon ECS 运行最敏感和任务关键型的应用程序。有关更多信息，请参阅 [Amazon Elastic Container Service 开发人员指南](#)。
- Amazon Elastic Kubernetes Service (Amazon EKS) 是一项托管式服务，可让您在 Amazon 上轻松运行 Kubernetes，而无需安装、操作或维护您自己的 Kubernetes 控制面板或节点。Kubernetes 是一个用于实现容器化应用程序的部署、扩缩和管理自动化的开源系统。有关更多信息，请参阅 [Amazon EKS 用户指南](#)。
- Amazon EC2 上的 Kubernetes。Kubernetes 是一个开源软件，可帮助您大规模部署和管理容器化应用程序。Kubernetes 可管理 Amazon EC2 计算实例的集群，并通过部署、维护和扩展流程在这些实例上运行容器。使用 Kubernetes，您可以在本地和云中使用相同的工具集运行任何类型的容器化应用程序。有关更多信息，请参阅 [在 Amazon EC2 上运行 Kubernetes](#)。
- Amazon FSx 可帮助您启动和运行流行的 Amazon 完全托管式文件系统。使用 Amazon FSx，您可以利用通用开源文件系统和获商业许可的文件系统的功能集和性能来避免耗时的管理任务。有关更多信息，请参阅 [Amazon FSx 文档](#)。
- Amazon Simple Notification Service (SNS) 是一项完全托管式消息收发服务，适用于应用程序间和应用程序与人之间的通信。您可以将 Amazon SNS 配置为由 Application Insights 对其进行监控。当将 Amazon SNS 配置为监控资源时，Application Insights 会跟踪 SNS 指标，帮助确定 SNS 消息可能会遇到问题或失败的原因。

#### 相关第三方服务

- 对于在 Application Insights 中监控的某些工作负载和应用程序，Prometheus JMX Exporter 使用 Amazon Systems Manager Distributor 安装，以便 CloudWatch Application Insights 可以检索 Java 特定的指标。当您选择监控 Java 应用程序时，Application Insights 会自动为您安装 Prometheus JMX Exporter。

## 支持的应用程序组件

CloudWatch Application Insights 可扫描您的资源组以识别应用程序组件。组件可能采用单独分组、自动分组（例如 Auto Scaling 组中的实例或负载均衡器后面的实例）或自定义分组（将各个 EC2 实例分组在一起）。

CloudWatch Application Insights 支持以下组件：

#### Amazon 组件

- Amazon EC2
- Amazon EBS
- Amazon RDS
- Elastic Load Balancing：Application Load Balancer 和经典负载均衡器（指定和配置这些负载均衡器的所有目标实例）。
- Amazon EC2 Auto Scaling 组：Amazon Auto Scaling（为所有目标实例动态配置 Auto Scaling 组；如果应用程序纵向扩展，CloudWatch Application Insights 将自动配置新实例）。基于 CloudFormation 的 Resource Groups 不支持 Auto Scaling 组。
- Amazon Lambda

- Amazon Simple Queue Service (Amazon SQS)
- Amazon DynamoDB 表
- Amazon S3 存储桶指标
- Amazon Step Functions
- Amazon API Gateway REST API 阶段
- Amazon Elastic Container Service (Amazon ECS)：集群、服务和任务
- Amazon Elastic Kubernetes Service (Amazon EKS)：集群
- Amazon EC2 上的 Kubernetes：在 EC2 上运行的 Kubernetes 集群
- Amazon SNS 主题

CloudWatch Application Insights 当前不会追踪任何其他组件类型资源。如果在 Application Insights 应用程序中未显示某种支持的组件类型，则您拥有并由 Application Insights 监控的其他应用程序可能已注册和管理该组件。

## 支持的技术堆栈

您可以在“Application tier (应用程序层)”下拉菜单选项中选择以下技术之一，从而使用 CloudWatch Application Insights 监控在 Windows Server 和 Linux 操作系统上运行的应用程序：

- 前端：Microsoft Internet 信息服务 (IIS) Web 服务器
- 工件层：
  - .NET Framework
  - .NET 内核
- 应用程序：Java
- Active Directory
- SharePoint
- 数据库：
  - 在 Amazon RDS 或 Amazon EC2 上运行的 Microsoft SQL Server (包括 SQL Server 高可用性配置，请参阅 [组件配置示例 \(p. 543\)](#))。
  - 在 Amazon RDS、Amazon Aurora 或 Amazon EC2 上运行的 MySQL
  - 在 Amazon RDS 或 Amazon EC2 上运行的 PostgreSQL
  - Amazon DynamoDB 表
  - 在 Amazon RDS 或 Amazon EC2 上运行的 Oracle
  - 单个 Amazon EC2 实例和多个 EC2 实例上的 SAP HANA 数据库
  - 跨可用区 SAP HANA 数据库高可用性设置。

如果上面列出的任何技术堆栈都不适用于您的应用程序资源，则您可以通过在 Manage monitoring (管理监控) 页面上的“Application tier (应用程序层)”下拉菜单中选择 Custom (自定义) 来监控您的应用程序堆栈。

## Amazon CloudWatch Application Insights 的工作方式

此部分包含有关 CloudWatch Application Insights 的工作方式的信息，其中包括：

- [Application Insights 监控应用程序的方式 \(p. 513\)](#)
- [数据保留 \(p. 513\)](#)
- [配额 \(p. 513\)](#)
- [CloudWatch Application Insights 使用的 Amazon Systems Manager \(SSM\) 软件包 \(p. 513\)](#)

## Application Insights 监控应用程序的方式

Application Insights 监控应用程序的方式如下。

### 应用程序发现和配置

首次将应用程序添加到 CloudWatch Application Insights 时，它扫描应用程序组件以建议关键指标、日志和其他数据源，以便监控您的应用程序。然后，您可以根据这些建议配置应用程序。

### 数据预处理

CloudWatch Application Insights 持续分析在应用程序资源中监控的数据源，以查找指标异常情况和日志错误（观察结果）。

### 智能问题检测

CloudWatch Application Insights 引擎使用分类算法和内置规则关联观察结果，以检测应用程序中的问题。为了帮助进行故障排除，它创建一些自动化 CloudWatch 控制面板，其中包含有关问题的上下文信息。

### 警报和操作

在 CloudWatch Application Insights 检测到应用程序出现问题时，它生成 CloudWatch Events 以通知您该问题。有关如何设置这些事件的更多信息，请参阅[检测到的问题的 Application CloudWatch Events 和通知 \(p. 536\)](#)。

### 示例方案

您具有一个由 SQL Server 数据库支持的 ASP .NET 应用程序。突然，您的数据库由于内存压力过高而开始发生故障。这会导致应用程序性能下降，并且可能会导致在 Web 服务器和负载均衡器中出现 HTTP 500 错误。

借助于 CloudWatch Application Insights 及其智能分析，您可以检查动态创建的控制面板（显示相关指标和日志文件片段）以找出导致问题的应用程序层。在这种情况下，问题可能出在 SQL 数据库层。

## 数据保留

CloudWatch Application Insights 将问题保留 55 天，并将观察结果保留 60 天。

## 配额

有关 CloudWatch Application Insights 的默认配额，请参阅[Amazon CloudWatch Application Insights 端点和配额](#)。除非另有说明，否则，每个配额针对的是每个 Amazon 区域。如需请求增加服务配额，请联系[Amazon Support](#)。许多服务包含无法更改的配额。有关特定服务的配额的更多信息，请参阅针对该服务的文档。

## CloudWatch Application Insights 使用的 Amazon Systems Manager (SSM) 软件包

本节中列出的软件包由 Application Insights 使用，可独立管理并可通过 Amazon Systems Manager Distributor 部署。有关 SSM Distributor 的更多信息，请参阅 Amazon Systems Manager 用户指南中的[Amazon Systems Manager Distributor](#)。

软件包：

- [AWSObservabilityExporter-JMXExporterInstallAndConfigure \(p. 514\)](#)
- [AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure \(p. 516\)](#)
- [AWSObservabilityExporter-HAClusterExporterInstallAndConfigure \(p. 518\)](#)

## AWSObservabilityExporter-JMXExporterInstallAndConfigure

您可以从 [Prometheus JMX Exporter](#) 中检索工作负载特定的 Java 指标，以供 Application Insights 配置和监控告警。在 Application Insights 控制台的 Manage monitoring ( 管理监控 ) 页面上，在 Application tier ( 应用程序层 ) 下拉菜单选择 JAVA application ( JAVA 应用程序 )。然后在 JAVA Prometheus exporter configuration ( JAVA Prometheus Exporter 配置 ) 下，选择您的 Collection method ( 收集方法 ) 和 JMX port number ( JMX 端口号 )。

要使用 [Amazon Systems Manager Distributor](#) 来打包、安装和配置 Amazon 提供的 Prometheus JMX Exporter ( 独立于 Application Insights )，请完成以下步骤。

使用 Prometheus JMX Exporter SSM 软件包的先决条件

- 2.3.1550.0 版或更高版本的 SSM Agent 已安装
- JAVA\_HOME 环境变量已设置

安装和配置 **AWSObservabilityExporter-JMXExporterInstallAndConfigure** 软件包

AWSObservabilityExporter-JMXExporterInstallAndConfigure 软件包是一个 SSM Distributor 软件包，您可以使用它来安装和配置 [Prometheus JMX Exporter](#)。当 Prometheus JMX Exporter 发送 Java 指标时，可以配置 CloudWatch 代理来检索 CloudWatch 服务的指标。

1. 根据您的首选项，准备位于 Prometheus GitHub 存储库中的 [Prometheus JMX Exporter YAML 配置文件](#)。使用示例配置和选项说明为您提供指导。
2. 将编码为 Base64 的 Prometheus JMX Exporter YAML 配置文件复制到 [SSM Parameter Store](#) 中的新 SSM 参数。
3. 导航到 [SSM Distributor](#) 控制台，然后打开 Owned by Amazon ( Amazon 所拥有 ) 选项卡。选择 AWSObservabilityExporter-JMXExporterInstallAndConfigure，然后选择 Install one time ( 安装一次 )。
4. 通过将“Additional Arguments ( 其他实际参数 )”替换为以下内容来更新您在第一步中创建的 SSM 参数：

```
{
 "SSM_EXPORTER_CONFIGURATION": "{{ssm:<SSM_PARAMETER_STORE_NAME>}}",
 "SSM_EXPOSITION_PORT": "9404"
}
```

### Note

端口 9404 是用于发送 Prometheus JMX 指标的默认端口。您可以更新此端口。

示例：配置 CloudWatch 代理以检索 Java 指标

1. 如前面的步骤所述，安装 Prometheus JMX Exporter。然后检查端口状态，确认它是否已正确安装在您的实例上。

在 Windows 实例示例上的成功安装

```
PS C:\> curl http://localhost:9404 (http://localhost:9404/)
StatusCode : 200
StatusDescription : OK
Content : # HELP jvm_info JVM version info
```

在 Linux 实例示例上的成功安装

```
$ curl localhost:9404
```

```
HELP jmx_config_reload_failure_total Number of times configuration have failed to be
reloaded.
TYPE jmx_config_reload_failure_total counter
jmx_config_reload_failure_total 0.0
```

2. 创建 Prometheus 服务发现 YAML 文件。以下示例服务发现文件将执行以下操作：

- 将 Prometheus JMX Exporter 主机端口指定为 localhost: 9404。
- 将标签 (Application、ComponentName 和 InstanceId) 附加到指标，这些指标可以设置为 CloudWatch 指标维度。

```
$ cat prometheus_sd_jmx.yaml
- targets:
 - 127.0.0.1:9404
 labels:
 Application: myApp
 ComponentName: arn-cn:aws:elasticloadbalancing:cn-
north-1:123456789012:loadbalancer/app/sample-Appli-MMZW8E3GH4H2/aac36d7fea2a6e5b
 InstanceId: i-12345678901234567
```

3. 创建 Prometheus JMX Exporter 配置 YAML 文件。以下示例配置文件可指定以下内容：

- 指标检索任务间隔和超时周期。
- 指标检索任务 (jmx 和 sap) 也称为抓取，其中包括任务名称、一次返回的最大时间序列以及服务发现文件路径。

```
$ cat prometheus.yaml
global:
 scrape_interval: 1m
 scrape_timeout: 10s
scrape_configs:
 - job_name: jmx
 sample_limit: 10000
 file_sd_configs:
 - files: ["/tmp/prometheus_sd_jmx.yaml"]
 - job_name: sap
 sample_limit: 10000
 file_sd_configs:
 - files: ["/tmp/prometheus_sd_sap.yaml"]
```

4. 验证 CloudWatch 代理是否已安装在您的 Amazon EC2 实例上，以及版本是否为 1.247346.1b249759 或更高版本。要在 EC2 实例上安装 CloudWatch 代理，请参阅[安装 CloudWatch 代理](#)。要验证版本，请参阅[查找有关 CloudWatch 代理版本的信息](#)。
5. 配置 CloudWatch 代理。有关如何配置 CloudWatch 代理配置文件的更多信息，请参阅[手动创建或编辑 CloudWatch 代理配置文件](#)。以下示例 CloudWatch 代理配置文件可执行以下操作：

- 指定 Prometheus JMX Exporter 配置文件路径。
- 指定要向其发布 EMF 指标日志的目标日志组。
- 为每个指标名称指定两组维度。
- 发送 8 个 (每个指标名称 2 组维度，每组 4 个指标名称) CloudWatch 指标。

```
{
 "logs":{
 "logs_collected":{

 },
 "metrics_collected":{
```

```
"prometheus":{
 "cluster_name":"prometheus-test-cluster",
 "log_group_name":"prometheus-test",
 "prometheus_config_path":"/tmp/prometheus.yaml",
 "emf_processor":{
 "metric_declaration_dedup":true,
 "metric_namespace":"CWAgent",
 "metric_unit":{
 "jvm_threads_current":"Count",
 "jvm_gc_collection_seconds_sum":"Second",
 "jvm_memory_bytes_used":"Bytes"
 },
 "metric_declaration":[
 {
 "source_labels":[
 "job"
],
 "label_matcher":"^jmx$",
 "dimensions":[
 [
 "InstanceId",
 "ComponentName"
],
 [
 "ComponentName"
]
],
 "metric_selectors":[
 "^java_lang_threading_threadcount$",
 "^java_lang_memory_heapmemoryusage_used$",
 "^java_lang_memory_heapmemoryusage_committed$"
]
 }
]
 },
 "metrics":{

 }
}
```

## AWSObservabilityExporter-SAP- HANADBExporterInstallAndConfigure

您可以从 [Prometheus HANA Database Exporter](#) 中检索工作负载特定的 SAP HANA 指标，以供 Application Insights 配置和监控告警。有关更多信息，请参阅本指南中的 [设置 SAP HANA 数据库以进行监控 \(p. 584\)](#)。

要使用 [Amazon Systems Manager Distributor](#) 来打包、安装和配置 Amazon 提供的 Prometheus HANA Database Exporter 软件包（独立于 Application Insights），请完成以下步骤。

使用 Prometheus HANA Database Exporter SSM 软件包的先决条件

- 2.3.1550.0 版或更高版本的 SSM Agent 已安装
- SAP HANA 数据库
- Linux 操作系统（SUSE Linux、RedHat Linux）
- 带有 SAP HANA 数据库监控凭证的密钥（使用 Amazon Secrets Manager）。使用键/值对格式创建密钥，指定密钥用户名，然后输入值的数据库用户。再添加一个密钥密码，然后输入值的密码。有关如何创建密钥的更多信息，请参阅 Amazon Secrets Manager 用户指南中的 [创建密钥](#)。密钥必须采用以下格式：



```
{
 "username": "<database_user>",
 "password": "<database_password>"
}
```

#### 安装和配置 `AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure` 软件包

`AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure` 软件包是一个 SSM Distributor 软件包，您可以使用它来安装和配置 [Prometheus HANA Database Exporter](#)。当 Prometheus HANA Database Exporter 发送 HANA 数据库指标时，就可以配置 CloudWatch 代理来检索 CloudWatch 服务的指标。

1. 在 [SSM Parameter Store](#) 中创建 SSM 参数以存储该 Exporter 的配置。以下为示例参数值。

```
{\"exposition_port\":9668,\"multi_tenant\":true,\"timeout\":600,\"hana\":{\"host\": \"localhost\", \"port\":30013,\"aws_secret_name\":\"HANA_DB_CREDS\", \"scale_out_mode\":true}}
```

#### Note

在此示例中，导出任务仅在具有活跃 SYSTEM 数据库的 Amazon EC2 实例上运行，并在其他 EC2 实例上保持空闲状态，以免指标重复。该 Exporter 可以检索 SYSTEM 数据库中的所有数据库租户信息。

2. 在 [SSM Parameter Store](#) 中创建 SSM 参数以存储该 Exporter 的指标查询。软件包可以接受多个指标参数。每个参数必须具有有效的 JSON 对象格式。以下为示例参数值：

```
{\"SELECT MAX(TIMESTAMP) TIMESTAMP, HOST, MEASURED_ELEMENT_NAME CORE, SUM(MAP(CAPTION, 'User Time', TO_NUMBER(VALUE), 0)) USER_PCT, SUM(MAP(CAPTION, 'System Time', TO_NUMBER(VALUE), 0)) SYSTEM_PCT, SUM(MAP(CAPTION, 'Wait Time', TO_NUMBER(VALUE), 0)) WAITIO_PCT, SUM(MAP(CAPTION, 'Idle Time', 0, TO_NUMBER(VALUE))) BUSY_PCT, SUM(MAP(CAPTION, 'Idle Time', TO_NUMBER(VALUE), 0)) IDLE_PCT FROM sys.M_HOST_AGENT_METRICS WHERE MEASURED_ELEMENT_TYPE = 'Processor' GROUP BY HOST, MEASURED_ELEMENT_NAME;\":{\"enabled\":true,\"metrics\":[{\"name\":\"hanadb_cpu_user\",\"description\":\"Percentage of CPU time spent by HANA DB in user space, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":\"USER_PCT\", \"unit\":\"percent\", \"type\":\"gauge\"},{\"name\":\"hanadb_cpu_system\",\"description\":\"Percentage of CPU time spent by HANA DB in Kernel space, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":\"SYSTEM_PCT\", \"unit\":\"percent\", \"type\":\"gauge\"},{\"name\":\"hanadb_cpu_waitio\",\"description\":\"Percentage of CPU time spent by HANA DB in IO mode, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":\"WAITIO_PCT\", \"unit\":\"percent\", \"type\":\"gauge\"},{\"name\":\"hanadb_cpu_busy\",\"description\":\"Percentage of CPU time spent by HANA DB, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":\"BUSY_PCT\", \"unit\":\"percent\", \"type\":\"gauge\"},{\"name\":\"hanadb_cpu_idle\",\"description\":\"Percentage of CPU time not spent by HANA DB, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":\"IDLE_PCT\", \"unit\":\"percent\", \"type\":\"gauge\"}]}}
```

有关指标查询的更多信息，请参阅 GitHub 上的 [SUSE / hanadb\\_exporter](#) 存储库。

3. 导航到 [SSM Distributor](#) 控制台，然后打开 Owned by Amazon ( Amazon 所拥有 ) 选项卡。依次选择 `AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure*` 和 Install one time ( 一次性安装 )。
4. 通过将“Additional Arguments ( 其他实际参数 )”替换为以下内容来更新您在第一步中创建的 SSM 参数：

```
{
 "SSM_EXPORTER_CONFIG": "{\"ssm:<*SSM_CONFIGURATIONS_PARAMETER_STORE_NAME*>\"}",
```

```
"SSM_SID": "<SAP_DATABASE_SID>",
"SSM_EXPORTER_METRICS_1": "{{ssm:<SSM_FIRST_METRICS_PARAMETER_STORE_NAME>}}",
"SSM_EXPORTER_METRICS_2": "{{ssm:<SSM_SECOND_METRICS_PARAMETER_STORE_NAME>}}"
}
```

5. 选择带有 SAP HANA 数据库的 Amazon EC2 实例，然后选择 Run (运行)。

## AWSObservabilityExporter-HAClusterExporterInstallAndConfigure

您可以从 [Prometheus HANA Cluster Exporter](#) 中检索工作负载特定的高可用性 (HA) 集群指标，以供 Application Insights 配置和监控 SAP HANA 高可用性设置的告警。有关更多信息，请参阅本指南中的 [设置 SAP HANA 数据库以进行监控](#) (p. 584)。

要使用 [Amazon Systems Manager Distributor](#) 来打包、安装和配置 Amazon 提供的 Prometheus HA Cluster Exporter 软件包 (独立于 Application Insights)，请完成以下步骤。

使用 Prometheus HA Cluster Exporter SSM 软件包的先决条件

- 2.3.1550.0 版或更高版本的 SSM Agent 已安装
- 适用于 Pacemaker、Corosync、SBD 和 DRBD 的 HA 集群
- Linux 操作系统 (SUSE Linux、RedHat Linux)

安装和配置 **AWSObservabilityExporter-HAClusterExporterInstallAndConfigure** 软件包

AWSObservabilityExporter-HAClusterExporterInstallAndConfigure 软件包是一个 SSM Distributor 软件包，您可以使用它来安装和配置 Prometheus HA Cluster Exporter。当 Prometheus HANA Database Exporter 发送集群指标时，就可以配置 CloudWatch 代理来检索 CloudWatch 服务的指标。

1. 在 [SSM Parameter Store](#) 中创建 SSM 参数，以 JSON 格式存储该 Exporter 的配置。以下为示例参数值。

```
{\ "port\":"9664",\ "address\":"0.0.0.0",\ "log-level\":"info",\ "crm-mon-path\":"\usr/sbin/crm_mon",\ "cibadmin-path\":"\usr/sbin/cibadmin",\ "corosync-cfgtool-path-path\":"\usr/sbin/corosync-cfgtool",\ "corosync-quorumtool-path\":"\usr/sbin/corosync-quorumtool",\ "sbd-path\":"\usr/sbin/sbd",\ "sbd-config-path\":"\etc/sysconfig/sbd",\ "drbdsetup-path\":"\sbin/drbdsetup",\ "enable-timestamps\":"false}
```

有关该 Exporter 配置的更多信息，请参阅 GitHub 上的 [ClusterLabs / ha\\_cluster\\_exporter](#) 存储库。

2. 导航到 [SSM Distributor](#) 控制台，然后打开 Owned by Amazon (Amazon 所拥有) 选项卡。依次选择 AWSObservabilityExporter-HAClusterExporterInstallAndConfigure\* 和 Install one time (一次性安装)。
3. 通过将“Additional Arguments (其他实际参数)”替换为以下内容来更新您在第一步中创建的 SSM 参数：

```
{
 "SSM_EXPORTER_CONFIG": "{{ssm:<*SSM_CONFIGURATIONS_PARAMETER_STORE_NAME>}}"
}
```

4. 选择带有 SAP HANA 数据库的 Amazon EC2 实例，然后选择 Run (运行)。

## 开始使用 Amazon CloudWatch Application Insights

要开始使用 CloudWatch Application Insights，请确认您满足下面列出的先决条件并创建了 IAM policy。然后，您可以开始使用控制台链接以启用 CloudWatch Application Insights。要配置应用程序资源，请执行 [设置、配置和管理您的应用程序以进行监控](#) (p. 521) 中的步骤。



## 目录

- [访问 CloudWatch Application Insights \(p. 519\)](#)
- [先决条件 \(p. 519\)](#)
- [IAM policy \(p. 519\)](#)
- [基于账户的应用程序载入的 IAM 角色权限 \(p. 520\)](#)
- [设置、配置和管理您的应用程序以进行监控 \(p. 521\)](#)

## 访问 CloudWatch Application Insights

您可以通过以下界面之一访问及管理 CloudWatch Application Insights：

- CloudWatch 控制台。要为应用程序添加监控，请选择 [CloudWatch 控制台](#) 左侧导航窗格中 Insights 下的 Application Insights。在配置应用程序后，您可以使用 [CloudWatch 控制台](#) 查看和分析检测到的问题。
- Amazon Command Line Interface (Amazon CLI)。您可以使用 Amazon CLI 访问 Amazon API 操作。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的 [安装 Amazon Command Line Interface](#)。有关 Application Insights API 的信息，请参阅 [Amazon CloudWatch Application Insights API 参考](#)。

## 先决条件

您必须完成以下必需任务才能使用 CloudWatch Application Insights 配置应用程序：

- Amazon SSM 启用。您必须安装 Systems Manager Agent (SSM Agent)，并且实例必须启用了 SSM。有关如何安装 SSM Agent 的信息，请参阅 [设置 Amazon SSM](#)。
- EC2 实例角色。您必须附加 AmazonSSMManagedInstanceCore 角色以启用 Systems Manager (请参阅 [Amazon SSM 使用基于身份的策略 \(IAM 策略\)](#))，并附加 CloudWatchAgentServerPolicy 以允许通过 CloudWatch 发出实例指标和日志。有关更多信息，请参阅 [创建用于 CloudWatch 代理的 IAM 角色和用户](#)。
- Amazon Resource Groups。要将应用程序添加到 CloudWatch Application Insights，您必须创建一个包含应用程序堆栈所用的所有相关 Amazon 资源的资源组。这包括应用程序负载均衡器、运行 IIS 和 Web 前端的 EC2 实例、.NET 工作线程层和 SQL Server 数据库。CloudWatch Application Insights 自动包含使用与资源组相同的标签或 CloudFormation 堆栈的 Auto Scaling 组，因为资源组当前不支持 Auto Scaling 组。有关更多信息，请参阅 [Amazon Resource Groups 入门](#)。
- IAM 权限：对于非管理员用户，必须创建允许 Application Insights 创建服务相关角色的 Amazon Identity and Access Management (IAM) 策略，并将其附加到您的用户身份。有关附加策略的步骤，请参阅 [IAM policy \(p. 519\)](#)。
- 服务相关角色：CloudWatch Application Insights 使用 Amazon Identity and Access Management (IAM) 服务相关角色。在 Amazon 管理控制台中创建第一个 CloudWatch Application Insights 应用程序时，将为您创建服务相关角色。有关更多信息，请参阅 [在 CloudWatch Application Insights 中使用服务相关角色 \(p. 871\)](#)。
- EC2 Windows 实例对性能计数器指标的支持：要在 EC2 Windows 实例上监控性能计数器指标，必须在实例上安装性能计数器。有关性能计数器指标和相应的性能计数器集名称，请参阅 [性能计数器指标 \(p. 649\)](#)。有关性能计数器的更多信息，请参阅 [性能计数器](#)。

## IAM policy

要使用 CloudWatch Application Insights，您必须创建 [Identity and Access Management \(IAM\) 策略](#) 并将其附加到您的 IAM 用户身份。IAM policy 定义了用户权限。

使用控制台创建 IAM policy

要使用 IAM 控制台创建 IAM policy，请执行以下步骤。

1. 转到 [IAM 控制台](#)。在左侧导航窗格中，选择策略。
2. 在页面顶部，选择创建策略。
3. 选择 JSON 选项卡。
4. 在 JSON 选项卡下面复制并粘贴以下 JSON 文档。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "applicationinsights:*",
 "iam:CreateServiceLinkedRole",
 "iam:ListRoles",
 "resource-groups:ListGroup"
],
 "Effect": "Allow",
 "Resource": "*"
 }
]
}
```

5. 选择查看策略。
6. 输入策略的名称，例如“AppInsightsPolicy”。（可选）输入描述。
7. 选择 Create Policy (创建策略)。
8. 从左侧导航窗格中选择用户。
9. 选择要将策略附加到的用户的用户名。
10. 选择添加权限。
11. 选择 Attach existing policies directly (直接附加现有策略)。
12. 搜索刚创建的策略，然后选中策略名称左侧的复选框。
13. 选择下一步: 审核。
14. 确保列出了正确的策略，然后选择添加权限。
15. 在使用 CloudWatch Application Insights 时，请确保使用与刚创建的策略关联的用户登录。

#### 使用 Amazon CLI 创建 IAM policy

要使用 Amazon CLI 创建 IAM policy，请使用上面的 JSON 文档作为当前文件夹中的文件，从命令行运行 [create-policy](#) 操作。

#### 使用 Amazon Tools for Windows PowerShell 创建 IAM policy

要使用 Amazon Tools for Windows PowerShell 创建 IAM policy，请使用上面的 JSON 文档作为当前文件夹中的文件，运行 [New-IAMPolicy](#) cmdlet。

## 基于账户的应用程序载入的 IAM 角色权限

如果您想载入账户中的所有资源，并选择不使用 [Application Insights 托管式策略 \(p. 878\)](#) 来完全访问 Application Insights 功能，则必须向 IAM 角色附加以下权限，以便 Application Insights 可以发现您账户中的所有资源：

```
"ec2:DescribeInstances"
"ec2:DescribeVolumes"
"rds:DescribeDBInstances"
"rds:DescribeDBClusters"
"sqs:ListQueues"
"elasticloadbalancing:DescribeLoadBalancers"
"autoscaling:DescribeAutoScalingGroups"
```

```
"lambda:ListFunctions"
"dynamodb:ListTables"
"s3:ListAllMyBuckets"
"sns:ListTopics"
"states:ListStateMachines"
"apigateway:GET"
"ecs:ListClusters"
"ecs:DescribeTaskDefinition"
"ecs:ListServices"
"ecs:ListTasks"
"eks:ListClusters"
"eks:ListNodegroups"
"fsx:DescribeFileSystems"
```

## 设置、配置和管理您的应用程序以进行监控

本节提供使用控制台、Amazon CLI 和 Amazon Tools for Windows PowerShell 来设置、配置和管理 CloudWatch Application Insights 应用程序的步骤。

### 主题

- [设置、配置和管理您的应用程序以从 CloudWatch 控制台进行监控 \(p. 521\)](#)
- [使用命令行设置、配置和管理您的应用程序以进行监控 \(p. 524\)](#)
- [检测到的问题的 Application CloudWatch Events 和通知 \(p. 536\)](#)

## 设置、配置和管理您的应用程序以从 CloudWatch 控制台进行监控

本节提供设置、配置和管理应用程序以从 CloudWatch 控制台进行监控的步骤。

### 控制台程序

- [添加和配置应用程序 \(p. 521\)](#)
- [为 Amazon ECS 和 Amazon EKS 资源监控启用 Application Insights \(p. 523\)](#)
- [为应用程序组件禁用监控 \(p. 524\)](#)
- [删除应用程序 \(p. 524\)](#)

### 添加和配置应用程序

从 CloudWatch 控制台添加和配置应用程序

要通过 CloudWatch 控制台开始使用 CloudWatch Application Insights，请执行以下步骤。

1. 开始。打开 [CloudWatch 控制台登陆页面](#)。从左侧导航窗格中，选择 Insights 下的 Application Insights。所打开的页面将显示使用 CloudWatch Application Insights 监控的应用程序列表及它们的监控状态。
2. 添加应用程序。要为应用程序设置监控，请选择 Add an application (添加应用程序)。在选择 Add an application (添加应用程序) 时，系统会提示您 Choose Application Type (选择应用程序类型)。
  - 基于资源组的应用程序。选择此选项后，您可以选择此账户中要监控的资源组。
  - 基于账户的应用程序。选择此选项后，您可以监控此账户中的所有资源。如果您想监控账户中的所有资源，我们建议使用此选项而不是基于资源组的选项，因为应用程序载入过程更快。

### Note

不能使用 Application Insights 将基于资源组的监控与基于账户的监控结合起来。要更改应用程序类型，必须删除所有正在监控之下的应用程序，并 Choose Application Type (选择应用程序类型)。

当您添加第一个应用程序以进行监控时，CloudWatch Application Insights 会在账户中创建服务相关角色，该角色授予 Application Insights 代表您调用其他 Amazon 服务的权限。有关 Application Insights 在您的账户中创建的服务相关角色的更多信息，请参阅 [在 CloudWatch Application Insights 中使用服务相关角色 \(p. 871\)](#)。

### 3. Resource-based application monitoring

1. 选择资源组。在 Specify application details (指定应用程序详细信息) 页面上，从下拉列表中选择包含您应用程序资源的 Amazon 资源组。这些资源包括前端服务器、负载均衡器、Auto Scaling 组和数据库服务器。

如果还没有为应用程序创建资源组，则可以选择 Create new resource group (创建新资源组) 来创建一个资源组。有关创建资源组的详细信息，请参阅 [Amazon Resource Groups 用户指南](#)。

2. 监控 CloudWatch Events。选中复选框以将 Application Insights 监控与 CloudWatch Events 集成，获取 Amazon EBS、Amazon EC2、Amazon CodeDeploy、Amazon ECS、Amazon Health API 和通知、Amazon RDS、Amazon S3 和 Amazon Step Functions 中的洞察。
3. 与 Amazon Systems Manager OpsCenter 集成。要在检测到所选应用程序的问题时查看问题并获得通知，请选中 Generate Systems Manager OpsCenter OpsItems for remedial actions (生成 Systems Manager OpsCenter OpsItems 以采取修复措施) 复选框。要跟踪用于解析与 Amazon 资源相关的操作工作项 (OpsItem) 的操作，请提供 SNS 主题 ARN。
4. 标签 (可选)。CloudWatch Application Insights 支持基于标签和基于 CloudFormation 的资源组 (Auto Scaling 组除外)。有关更多信息，请参阅[使用标签编辑器](#)。
5. 选择 Next (下一步)。

将按以下格式为应用程序生成 ARN：

```
arn:partition:applicationinsights:region:account-id:application/resource-group/resource-group-name
```

示例

```
arn-cn:aws:applicationinsights:cn-north-1:123456789012:application/resource-group/my-resource-group
```

### Account-based application monitoring

1. 应用程序名称。为基于账户的应用程序输入名称。
2. 自动监控新资源。默认情况下，Application Insights 使用推荐的设置来配置对载入应用程序后添加到账户的资源组件的监控。通过清除复选框，可以排除对载入应用程序后添加的资源的监控。
3. 监控 CloudWatch Events。选中复选框以将 Application Insights 监控与 CloudWatch Events 集成，获取 Amazon EBS、Amazon EC2、Amazon CodeDeploy、Amazon ECS、Amazon Health API 和通知、Amazon RDS、Amazon S3 和 Amazon Step Functions 中的洞察。
4. 与 Amazon Systems Manager OpsCenter 集成。要在检测到所选应用程序的问题时查看问题并获得通知，请选中 Generate Systems Manager OpsCenter OpsItems for remedial actions (生成 Systems Manager OpsCenter OpsItems 以采取修复措施) 复选框。要跟踪用于解析与 Amazon 资源相关的操作工作项 (OpsItem) 的操作，请提供 SNS 主题 ARN。
5. 标签 (可选)。CloudWatch Application Insights 支持基于标签和基于 CloudFormation 的资源组 (Auto Scaling 组除外)。有关更多信息，请参阅[使用标签编辑器](#)。
6. 发现的资源。在您的账户中发现的所有资源都会添加到此列表中。如果 Application Insights 无法发现您账户中的所有资源，则会在页面顶部显示一条错误消息。此消息中包含一个指向[有关如何添加所需权限的文档 \(p. 520\)](#)的链接。
7. 选择 Next (下一步)。

将按以下格式为应用程序生成 [ARN](#)：

```
arn:partition:applicationinsights:region:account-id:application/TBD/application-name
```

示例

```
arn-cn:aws:applicationinsights:cn-north-1:123456789012:application/TBD/my-application
```

4. 提交应用程序监控配置后，您将转到应用程序的详细信息页面，您可以在该页面查看 Application summary（应用程序摘要）、Monitored components（已监控组件）的列表和 Unmonitored components（未监控组件）的列表，并通过选择 Components（组件）旁边的选项卡，查看 Configuration history（配置历史记录）、Log patterns（日志模式）和您已应用的任何 Tags（标签）。

要查看关于应用程序的洞察，请选择 View Insights（查看洞察）。

通过选择 Edit（编辑），您可以更新 CloudWatch Events 监控的选项以及与 Amazon Systems Manager OpsCenter 的集成。

在 Components（组件）下，您可以选择 Actions（操作）菜单来创建、修改或取消分组实例组。

您可以通过选择组件旁边的项目符号并选择 Manage monitoring（管理监控）来管理组件的监控，包括应用程序层、日志组、事件日志、指标和自定义告警。

## 为 Amazon ECS 和 Amazon EKS 资源监控启用 Application Insights

您可以启用 Application Insights，以从 Container Insights 控制台监控容器化应用程序和微服务。Application Insights 支持监控以下资源：

- Amazon ECS 集群
- Amazon ECS 服务
- Amazon ECS 任务
- Amazon EKS 集群

启用 Application Insights 后，它会提供建议的指标和日志，检测潜在问题，生成 CloudWatch Events，以及为容器化应用程序和微服务创建自动控制面板。

您可以从 Container Insights 或 Application Insights 控制台为容器化资源启用 Application Insights。

从 Container Insights 控制台启用 Application Insights

在 Container Insights 控制台中的 Container Insights Performance monitoring（性能监控）控制面板上，选择 Auto-configure Application Insights（自动配置 Application Insights）。启用 Application Insights 后，它会显示有关检测到的问题的详细信息。

从 Application Insights 控制台启用 Application Insights

当组件列表中出现 ECS 集群时，Application Insights 会自动使用 Container Insights 启用其他容器监控。

对于 EKS 集群，您可以通过 Container Insights 启用其他监控，提供诊断信息（如容器重启失败），以帮助您查明问题并解决问题。要为 EKS 设置 Container Insights，还需要执行其他步骤。有关详细信息，请参阅 [在 Amazon EKS 和 Kubernetes 上设置 Container Insights \(p. 367\)](#) 了解在 EKS 上设置 Container Insights 的步骤。

使用 EKS 的 Linux 实例支持使用 Container Insights 对 EKS 进行额外监控。

有关对 ECS 和 EKS 集群的 Container Insights 支持的更多信息，请参阅 [使用 Container Insights \(p. 353\)](#)。

### 为应用程序组件禁用监控

要为应用程序组件禁用监控，请从应用程序详细信息页面中选择要禁用监控的组件。选择 Actions ( 操作 )，然后 Remove from monitoring ( 取消监控 )。

### 删除应用程序

要删除应用程序，请在 CloudWatch 控制面板中的左侧导航窗格中选择 Insights 下方的 Application Insights。选择要删除的应用程序。在 Actions ( 操作 ) 下选择 Delete application ( 删除应用程序 )。这会删除监控，并删除为应用程序组件保存的所有监视器。不会删除应用程序资源。

## 使用命令行设置、配置和管理您的应用程序以进行监控

本节提供设置、配置和管理应用程序以使用 Amazon CLI 和 Amazon Tools for Windows PowerShell 进行监控的步骤。

### 命令行程序

- [添加和管理应用程序 \(p. 524\)](#)
- [管理和更新监控 \(p. 527\)](#)
- [针对 SQL Always On 可用性组配置监控 \(p. 530\)](#)
- [为 MySQL RDS 配置监控 \(p. 533\)](#)
- [为 MySQL EC2 配置监控 \(p. 533\)](#)
- [为 PostgreSQL RDS 配置监控 \(p. 533\)](#)
- [为 PostgreSQL EC2 配置监控 \(p. 534\)](#)
- [为 Oracle RDS 配置监控 \(p. 534\)](#)
- [为 Oracle EC2 配置监控 \(p. 535\)](#)

### 添加和管理应用程序

您可以使用命令行添加、管理和配置 Application Insights 应用程序以及获取有关它的信息。

### 主题

- [添加应用程序 \(p. 524\)](#)
- [描述应用程序 \(p. 525\)](#)
- [列出应用程序中的组件 \(p. 525\)](#)
- [描述组件 \(p. 525\)](#)
- [将类似资源分组到自定义组件中 \(p. 526\)](#)
- [对自定义组件取消分组 \(p. 526\)](#)
- [更新应用程序 \(p. 526\)](#)
- [更新自定义组件 \(p. 527\)](#)

### 添加应用程序

#### 使用 Amazon CLI 添加应用程序

要使用 Amazon CLI 为名为 my-resource-group 的资源组添加应用程序，并启用 OpsCenter 以将创建的 opsItem 传递到 SNS 主题 ARN arn-cn:aws:sns:cn-north-1:123456789012:MyTopic，请使用以下命令。



```
aws application-insights create-application --resource-group-name my-resource-group --ops-center-enabled --ops-item-sns-topic-arn arn-cn:aws:sns:cn-north-1:123456789012:MyTopic
```

使用 Amazon Tools for Windows PowerShell 添加应用程序

要使用 Amazon Tools for Windows PowerShell 为名为 `my-resource-group` 的资源组添加应用程序，并启用 OpsCenter 以将创建的 `opsItem` 传递到 SNS 主题 ARN `arn:aws:sns:cn-north-1:123456789012:MyTopic`，请使用以下命令。

```
New-CWAIAApplication -ResourceGroupName my-resource-group -OpsCenterEnabled true -OpsItemSNSTopicArn arn-cn:aws:sns:cn-north-1:123456789012:MyTopic
```

### 描述应用程序

使用 Amazon CLI 描述应用程序

要使用 Amazon CLI 描述在名为 `my-resource-group` 的资源组上创建的应用程序，请使用以下命令。

```
aws application-insights describe-application --resource-group-name my-resource-group
```

使用 Amazon Tools for Windows PowerShell 描述应用程序

要使用 Amazon Tools for Windows PowerShell 描述在名为 `my-resource-group` 的资源组上创建的应用程序，请使用以下命令。

```
Get-CWAIAApplication -ResourceGroupName my-resource-group
```

### 列出应用程序中的组件

使用 Amazon CLI 列出应用程序中的组件

要使用 Amazon CLI 列出在名为 `my-resource-group` 的资源组上创建的组件，请使用以下命令。

```
aws application-insights list-components --resource-group-name my-resource-group
```

使用 Amazon Tools for Windows PowerShell 列出应用程序中的组件

要使用 Amazon Tools for Windows PowerShell 列出在名为 `my-resource-group` 的资源组上创建的组件，请使用以下命令。

```
Get-CWAIDComponentList -ResourceGroupName my-resource-group
```

### 描述组件

使用 Amazon CLI 描述组件

您可以使用以下 Amazon CLI 命令描述属于在名为 `my-resource-group` 的资源组上创建的应用程序且名为 `my-component` 的组件。

```
aws application-insights describe-component --resource-group-name my-resource-group --component-name my-component
```

使用 Amazon Tools for Windows PowerShell 描述组件

您可以使用以下 Amazon Tools for Windows PowerShell 命令描述属于在名为 `my-resource-group` 的资源组上创建的应用程序且名为 `my-component` 的组件。

```
Get-CWAComponent -ComponentName my-component -ResourceGroupName my-resource-group
```

### 将类似资源分组到自定义组件中

我们建议将类似的资源（如 .NET Web 服务器实例）分组到自定义组件中，以轻松添加资源以及更好地进行监控和提供相应的信息。目前，CloudWatch Application Insights 支持 EC2 实例的自定义组。

使用 Amazon CLI 将资源分组到自定义组件

要为名为 `my-resource-group` 的资源组创建的应用程序使用 Amazon CLI 将三个实例（`arn-cn:aws:ec2:cn-north-1:123456789012:instance/i-11111`、`arn-cn:aws:ec2:cn-north-1:123456789012:instance/i-22222` 和 `arn-cn:aws:ec2:cn-north-1:123456789012:instance/i-33333`）一起分组到一个名为 `my-component` 的自定义组件，请使用以下命令。

```
aws application-insights create-component --resource-group-name my-resource-group --component-name my-component --resource-list arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-11111 arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-22222 arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-33333
```

使用 Amazon Tools for Windows PowerShell 将资源分组到自定义组件

要为名为 `my-resource-group` 的资源组创建的应用程序使用 Amazon Tools for Windows PowerShell 将三个实例（`arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-11111`、`arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-22222` 和 `arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-33333`）一起分组到一个名为 `my-component` 的自定义组件，请使用以下命令。

```
New-CWAComponent -ResourceGroupName my-resource-group -ComponentName my-component -ResourceList arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-11111,arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-22222,arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-33333
```

### 对自定义组件取消分组

使用 Amazon CLI 对自定义组件取消分组

要使用 Amazon CLI 对在资源组 `my-resource-group` 上创建的应用程序中名为 `my-component` 的自定义组件取消分组，请使用以下命令。

```
aws application-insights delete-component --resource-group-name my-resource-group --component-name my-new-component
```

使用 Amazon Tools for Windows PowerShell 对自定义组件取消分组

要使用 Amazon Tools for Windows PowerShell 对在资源组 `my-resource-group` 上创建的应用程序中名为 `my-component` 的自定义组件取消分组，请使用以下命令。

```
Remove-CWAComponent -ComponentName my-component -ResourceGroupName my-resource-group
```

### 更新应用程序

使用 Amazon CLI 更新应用程序



您可以使用 Amazon CLI 更新应用程序，以针对在应用程序中检测到的问题生成 Amazon Systems Manager OpsCenter OpsItem，并使用以下命令将创建的 OpsItem 与 SNS 主题 `arn:aws-cn:sns:cn-north-1:123456789012:MyTopic` 相关联。

```
aws application-insights update-application --resource-group-name my-resource-group --ops-center-enabled --ops-item-sns-topic-arn arn:aws-cn:sns:cn-north-1:123456789012:MyTopic
```

使用 Amazon Tools for Windows PowerShell 更新应用程序

您可以使用 Amazon Tools for Windows PowerShell 更新应用程序，以针对在应用程序中检测到的问题生成 Amazon SSM OpsCenter OpsItem，并使用以下命令将创建的 OpsItem 与 SNS 主题 相关联。

```
Update-CWAIAApplication -ResourceGroupName my-resource-group -OpsCenterEnabled true -OpsItemSNSTopicArn arn:aws-cn:sns:cn-north-1:123456789012:MyTopic
```

## 更新自定义组件

使用 Amazon CLI 更新自定义组件

您可以使用 Amazon CLI，通过使用以下命令，使用新的组件名称 `my-new-component` 和更新的实例组来更新名为 `my-component` 的自定义组件。

```
aws application-insights update-component --resource-group-name my-resource-group --component-name my-component --new-component-name my-new-component --resource-list arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-44444 arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-55555
```

使用 Amazon Tools for Windows PowerShell 更新自定义组件

您可以使用 Amazon Tools for Windows PowerShell，通过使用以下命令，使用新的组件名称 `my-new-component` 和更新的实例组来更新名为 `my-component` 的自定义组件。

```
Update-CWAIDComponent -ComponentName my-component -NewComponentName my-new-component -ResourceGroupName my-resource-group -ResourceList arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-44444,arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-55555
```

## 管理和更新监控

您可以使用命令行管理和更新针对 Application Insights 应用程序的监控。

### 主题

- [列出应用程序的问题 \(p. 527\)](#)
- [描述应用程序问题 \(p. 528\)](#)
- [描述与问题关联的异常或错误 \(p. 528\)](#)
- [描述应用程序的异常或错误 \(p. 528\)](#)
- [描述组件的监控配置 \(p. 529\)](#)
- [描述建议的组件监控配置 \(p. 529\)](#)
- [更新组件的监控配置 \(p. 529\)](#)
- [对指定的资源组取消 Application Insights 监控 \(p. 529\)](#)

## 列出应用程序的问题

使用 Amazon CLI 列出应用程序的问题

要使用 Amazon CLI 列出自 Unix Epoch 以来针对在名为 `my-resource-group` 的资源组上创建的应用程序在 1000 到 10000 毫秒之间检测到的应用程序问题，请使用以下命令。

```
aws application-insights list-problems --resource-group-name my-resource-group --start-time 1000 --end-time 10000
```

使用 Amazon Tools for Windows PowerShell 列出应用程序的问题

要使用 Amazon Tools for Windows PowerShell 列出自 Unix Epoch 以来针对在名为 `my-resource-group` 的资源组上创建的应用程序在 1000 到 10000 毫秒之间检测到的应用程序问题，请使用以下命令。

```
$startDate = "8/6/2019 3:33:00"
$endDate = "8/6/2019 3:34:00"
Get-CWAIProblemList -ResourceGroupName my-resource-group -StartTime $startDate -
EndTime $endDate
```

### 描述应用程序问题

使用 Amazon CLI 描述应用程序问题

要使用 Amazon CLI 描述问题 ID 为 `p-1234567890` 的问题，请使用以下命令。

```
aws application-insights describe-problem --problem-id p-1234567890
```

使用 Amazon Tools for Windows PowerShell 描述应用程序问题

要使用 Amazon Tools for Windows PowerShell 描述问题 ID 为 `p-1234567890` 的问题，请使用以下命令。

```
Get-CWAIProblem -ProblemId p-1234567890
```

### 描述与问题关联的异常或错误

使用 Amazon CLI 描述与问题关联的异常或错误

要使用 Amazon CLI 描述与问题 ID 为 `p-1234567890` 的问题关联的异常或错误，请使用以下命令。

```
aws application-insights describe-problem-observations --problem-id p-1234567890
```

使用 Amazon Tools for Windows PowerShell 描述与问题关联的异常或错误

要使用 Amazon Tools for Windows PowerShell 描述与问题 ID 为 `p-1234567890` 的问题关联的异常或错误，请使用以下命令。

```
Get-CWAIProblemObservation -ProblemId p-1234567890
```

### 描述应用程序的异常或错误

使用 Amazon CLI 描述应用程序的异常或错误

要使用 Amazon CLI 描述观察 ID 为 `o-1234567890` 的应用程序的异常或错误，请使用以下命令。

```
aws application-insights describe-observation --observation-id o-1234567890
```

使用 Amazon Tools for Windows PowerShell 描述应用程序的异常或错误

要使用 Amazon Tools for Windows PowerShell 描述观察 ID 为 o-1234567890 的应用程序的异常或错误，请使用以下命令。

```
Get-CWAIObservation -ObservationId o-1234567890
```

### 描述组件的监控配置

使用 Amazon CLI 组件描述组件的监控配置

要使用 Amazon CLI 描述在资源组 my-resource-group 上创建的应用程序中名为 my-component 的组件的监控配置，请使用以下命令。

```
aws application-insights describe-component-configuration --resource-group-name my-resource-group --component-name my-component
```

使用 Amazon Tools for Windows PowerShell 描述组件的监控配置

要使用 Amazon Tools for Windows PowerShell 描述在资源组 my-resource-group 上创建的应用程序中名为 my-component 的组件的监控配置，请使用以下命令。

```
Get-CWAIDComponentConfiguration -ComponentName my-component -ResourceGroupName my-resource-group
```

有关组件配置和示例 JSON 文件的更多信息，请参阅[使用组件配置 \(p. 536\)](#)。

### 描述建议的组件监控配置

使用 Amazon CLI 描述建议的组件监控配置

当组件是 .NET Worker 应用程序的一部分时，您可以使用 Amazon CLI，通过使用以下命令来描述在资源组 my-resource-group 上创建的应用程序中名为 my-component 的组件的建议监控配置。

```
aws application-insights describe-component-configuration-recommendation --resource-group-name my-resource-group --component-name my-component --tier DOT_NET_WORKER
```

使用 Amazon Tools for Windows PowerShell 描述建议的组件监控配置

当组件是 .NET Worker 应用程序的一部分时，您可以使用 Amazon Tools for Windows PowerShell，通过使用以下命令来描述在资源组 my-resource-group 上创建的应用程序中名为 my-component 的组件的建议监控配置。

```
Get-CWAIDComponentConfigurationRecommendation -ComponentName my-component -ResourceGroupName my-resource-group -Tier DOT_NET_WORKER
```

有关组件配置和示例 JSON 文件的更多信息，请参阅[使用组件配置 \(p. 536\)](#)。

### 更新组件的监控配置

使用 Amazon CLI 更新组件的监控配置

要使用 Amazon CLI 更新在名为 my-resource-group 的资源组上创建的应用程序中名为 my-component 的组件，请使用以下命令。该命令包括以下操作：

1. 启用组件监控。
2. 将组件层设置为 .NET Worker。
3. 更新组件的 JSON 配置以从本地文件 configuration.txt 中读取。

```
aws application-insights update-component-configuration --resource-group-name my-resource-group --component-name my-component --tier DOT_NET_WORKER --monitor --component-configuration "file://configuration.txt"
```

使用 Amazon Tools for Windows PowerShell 更新组件的监控配置

要使用 Amazon Tools for Windows PowerShell 更新在名为 `my-resource-group` 的资源组上创建的应用程序中名为 `my-component` 的组件，请使用以下命令。该命令包括以下操作：

1. 启用组件监控。
2. 将组件层设置为 `.NET Worker`。
3. 更新组件的 JSON 配置以从本地文件 `configuration.txt` 中读取。

```
[string]$config = Get-Content -Path configuration.txt
Update-CWAComponentConfiguration -ComponentName my-component -ResourceGroupName my-resource-group -Tier DOT_NET_WORKER -Monitor 1 -ComponentConfiguration $config
```

有关组件配置和示例 JSON 文件的更多信息，请参阅[使用组件配置 \(p. 536\)](#)。

### 对指定的资源组取消 Application Insights 监控

使用 Amazon CLI 对指定的资源组取消 Application Insights 监控

要使用 Amazon CLI 对在名为 `my-resource-group` 的资源组上创建的应用程序取消监控，请使用以下命令。

```
aws application-insights delete-application --resource-group-name my-resource-group
```

使用 Amazon Tools for Windows PowerShell 对指定的资源组取消 Application Insights 监控

要使用 Amazon Tools for Windows PowerShell 对在名为 `my-resource-group` 的资源组上创建的应用程序取消监控，请使用以下命令。

```
Remove-CWAApplication -ResourceGroupName my-resource-group
```

### 针对 SQL Always On 可用性组配置监控

1. 使用 SQL HA EC2 实例为资源组创建应用程序。

```
aws application-insights create-application --region <REGION> --resource-group-name <RESOURCE_GROUP_NAME>
```

2. 通过创建新的应用程序组件，定义表示 SQL HA 集群的 EC2 实例。

```
aws application-insights create-component --resource-group-name <RESOURCE_GROUP_NAME> --component-name SQL_HA_CLUSTER --resource-list "arn:aws-cn:ec2:<REGION>:<ACCOUNT_ID>:instance/<CLUSTER_INSTANCE_1_ID>" "arn:aws-cn:ec2:<REGION>:<ACCOUNT_ID>:instance/<CLUSTER_INSTANCE_2_ID>"
```

3. 配置 SQL HA 组件。

```
aws application-insights update-component-configuration --resource-group-name <RESOURCE_GROUP_NAME> --region <REGION> --component-name "SQL_HA_CLUSTER" --monitor --tier SQL_SERVER_ALWAYS_ON_AVAILABILITY_GROUP --monitor --component-configuration '{
 "subComponents" : [{
 "subComponentType" : "AWS::EC2::Instance",
 }
]
```

```

"alarmMetrics" : [{
 "alarmMetricName" : "CPUUtilization",
 "monitor" : true
}, {
 "alarmMetricName" : "StatusCheckFailed",
 "monitor" : true
}, {
 "alarmMetricName" : "Processor % Processor Time",
 "monitor" : true
}, {
 "alarmMetricName" : "Memory % Committed Bytes In Use",
 "monitor" : true
}, {
 "alarmMetricName" : "Memory Available Mbytes",
 "monitor" : true
}, {
 "alarmMetricName" : "Paging File % Usage",
 "monitor" : true
}, {
 "alarmMetricName" : "System Processor Queue Length",
 "monitor" : true
}, {
 "alarmMetricName" : "Network Interface Bytes Total/sec",
 "monitor" : true
}, {
 "alarmMetricName" : "PhysicalDisk % Disk Time",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:Buffer Manager Buffer cache hit ratio",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:Buffer Manager Page life expectancy",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:General Statistics Processes blocked",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:General Statistics User Connections",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:Locks Number of Deadlocks/sec",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:SQL Statistics Batch Requests/sec",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:Database Replica File Bytes Received/sec",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:Database Replica Log Bytes Received/sec",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:Database Replica Log remaining for undo",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:Database Replica Log Send Queue",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:Database Replica Mirrored Write Transaction/sec",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:Database Replica Recovery Queue",
 "monitor" : true
}, {
 "alarmMetricName" : "SQLServer:Database Replica Redo Bytes Remaining",
 "monitor" : true
}

```

```

 }, {
 "alarmMetricName" : "SQLServer:Database Replica Redone Bytes/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Total Log requiring undo",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Transaction Delay",
 "monitor" : true
 }],
 "windowsEvents" : [{
 "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
 "eventName" : "Application",
 "eventLevels" : ["WARNING", "ERROR", "CRITICAL", "INFORMATION"],
 "monitor" : true
 }, {
 "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
 "eventName" : "System",
 "eventLevels" : ["WARNING", "ERROR", "CRITICAL"],
 "monitor" : true
 }, {
 "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
 "eventName" : "Security",
 "eventLevels" : ["WARNING", "ERROR", "CRITICAL"],
 "monitor" : true
 }],
 "logs" : [{
 "logGroupName" : "SQL_SERVER_ALWAYS_ON_AVAILABILITY_GROUP-<RESOURCE_GROUP_NAME>",
 "logPath" : "C:\\Program Files\\Microsoft SQL Server\\MSSQL**\\MSSQLSERVER\\MSSQL\\Log\\ERRORLOG",
 "logType" : "SQL_SERVER",
 "monitor" : true,
 "encoding" : "utf-8"
 }]
}, {
 "subComponentType" : "AWS::EC2::Volume",
 "alarmMetrics" : [{
 "alarmMetricName" : "VolumeReadBytes",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeWriteBytes",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeReadOps",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeWriteOps",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeQueueLength",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeThroughputPercentage",
 "monitor" : true
 }, {
 "alarmMetricName" : "BurstBalance",
 "monitor" : true
 }]
}]
}'

```

## Note

Application Insights 必须获取应用程序事件日志（信息级别）才能检测集群活动（如故障转移）。

## 为 MySQL RDS 配置监控

1. 使用 RDS MySQL 数据库实例为资源组创建应用程序。

```
aws application-insights create-application --region <REGION> --resource-group-name
<RESOURCE_GROUP_NAME>
```

2. 默认情况下，启用错误日志。可以使用数据参数组启用慢速查询日志。有关更多信息，请参阅[访问 MySQL 慢速查询和常规日志](#)。

- set slow\_query\_log = 1
- set log\_output = FILE

3. 将要监控的日志导出到 CloudWatch Logs 中。有关更多信息，请参阅[将 MySQL 日志发布到 CloudWatch Logs](#)。

4. 配置 MySQL RDS 组件。

```
aws application-insights update-component-configuration --resource-group-name
" <RESOURCE_GROUP_NAME>" --region <REGION> --component-name "<DB_COMPONENT_NAME>"
--monitor --tier DEFAULT --monitor --component-configuration "{\"alarmMetrics\":
[{\"alarmMetricName\": \"CPUUtilization\", \"monitor\": true}], \"logs\": [{\"logType\":
\"MYSQL\", \"monitor\": true}, {\"logType\": \"MYSQL_SLOW_QUERY\", \"monitor\": false}]}"
```

## 为 MySQL EC2 配置监控

1. 使用 SQL HA EC2 实例为资源组创建应用程序。

```
aws application-insights create-application --region <REGION> --resource-group-name
<RESOURCE_GROUP_NAME>
```

2. 默认情况下，启用错误日志。可以使用数据参数组启用慢速查询日志。有关更多信息，请参阅[访问 MySQL 慢速查询和常规日志](#)。

- set slow\_query\_log = 1
- set log\_output = FILE

3. 配置 MySQL EC2 组件。

```
aws application-insights update-component-configuration --resource-group-name
" <RESOURCE_GROUP_NAME>" --region <REGION> --component-name "<DB_COMPONENT_NAME>"
--monitor --tier MYSQL --monitor --component-configuration "{\"alarmMetrics\":
[{\"alarmMetricName\": \"CPUUtilization\", \"monitor\": true}], \"logs\": [{\"logGroupName
\": \"<UNIQUE_LOG_GROUP_NAME>\", \"logPath\": \"C:\\\\ProgramData\\\\MySQL\\\\MySQL Server
*\\\\Data\\\\<FILE_NAME>.err\", \"logType\": \"MYSQL\", \"monitor\": true, \"encoding\":
\"utf-8\"}]}"
```

## 为 PostgreSQL RDS 配置监控

1. 使用 Postgre RDS 数据库实例为资源组创建应用程序。

```
aws application-insights create-application --region <REGION> --resource-group-name
<RESOURCE_GROUP_NAME>
```

2. 默认情况下，没有启用将 PostgreSQL 日志发布到 CloudWatch 的功能。要启用监控，请打开 RDS 控制台并选择要监控的数据库。选择右上角的 Modify ( 修改 )，然后选中标注为 PostgreSQL 日志的复选框。选择 Continue ( 继续 ) 以保存此设置。

3. 您的 PostgreSQL 日志将导出到 CloudWatch。

4. 配置 PostgreSQL RDS 组件。



```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --tier
DEFAULT --component-configuration
{
 \"alarmMetrics\": [
 {
 \"alarmMetricName\": \"CPUUtilization\",
 \"monitor\": true
 }
],
 \"logs\": [
 {
 \"logType\": \"POSTGRESQL\",
 \"monitor\": true
 }
]
}
```

## 为 PostgreSQL EC2 配置监控

1. 使用 PostgreSQL EC2 实例为资源组创建应用程序。

```
aws application-insights create-application --region <REGION> --resource-group-name
<RESOURCE_GROUP_NAME>
```

2. 配置 PostgreSQL EC2 组件。

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --tier
POSTGRESQL --component-configuration
{
 \"alarmMetrics\": [
 {
 \"alarmMetricName\": \"CPUUtilization\",
 \"monitor\": true
 }
],
 \"logs\": [
 {
 \"logGroupName\": \"<UNIQUE_LOG_GROUP_NAME>\",
 \"logPath\": \"/var/lib/pgsql/data/log/\",
 \"logType\": \"POSTGRESQL\",
 \"monitor\": true,
 \"encoding\": \"utf-8\"
 }
]
}
```

## 为 Oracle RDS 配置监控

1. 使用 Oracle RDS 数据库实例为资源组创建应用程序。

```
aws application-insights create-application --region <REGION> --resource-group-name
<RESOURCE_GROUP_NAME>
```

2. 默认情况下，没有启用将 Oracle 日志发布到 CloudWatch 的功能。要启用监控，请打开 RDS 控制台并选择要监控的数据库。选择右上角的 Modify（修改），然后选中标注为 Alert（提示）日志和 Listener（侦听器）日志的复选框。选择 Continue（继续）以保存此设置。

3. 您的 Oracle 日志将导出到 CloudWatch。
4. 配置 Oracle RDS 组件。

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --tier
DEFAULT --component-configuration
"{
 \"alarmMetrics\":[
 {
 \"alarmMetricName\": \"CPUUtilization\",
 \"monitor\": true
 }
],
 \"logs\":[
 {
 \"logType\": \"ORACLE_ALERT\",
 \"monitor\": true
 },
 {
 \"logType\": \"ORACLE_LISTENER\",
 \"monitor\": true
 }
]
}
```

## 为 Oracle EC2 配置监控

1. 使用 Oracle EC2 实例为资源组创建应用程序。

```
aws application-insights create-application --region <REGION> --resource-group-name
<RESOURCE_GROUP_NAME>
```

2. 配置 Oracle EC2 组件。

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --tier
ORACLE --component-configuration
"{
 \"alarmMetrics\":[
 {
 \"alarmMetricName\": \"CPUUtilization\",
 \"monitor\": true
 }
],
 \"logs\":[
 {
 \"logGroupName\": \"<UNIQUE_LOG_GROUP_NAME>\",
 \"logPath\": \"/opt/oracle/diag/rdbms/*/trace\",
 \"logType\": \"ORACLE_ALERT\",
 \"monitor\": true,
 },
 {
 \"logGroupName\": \"<UNIQUE_LOG_GROUP_NAME>\",
 \"logPath\": \"/opt/oracle/diag/tnslsnr/$HOSTNAME/listener/trace/\",
 \"logType\": \"ORACLE_ALERT\",
 \"monitor\": true,
 }
]
}
```

## 检测到的问题的 Application CloudWatch Events 和通知

对于添加到 CloudWatch Application Insights 的每个应用程序，将尽可能为以下事件发布 CloudWatch Events：

- 问题创建。在 CloudWatch Application Insights 检测到新问题时发出。
  - 详细信息类型：“检测到 Application Insights 问题”
  - 详细信息：
    - `problemId`：检测到的问题 ID。
    - `region`：创建问题的 Amazon 区域。
    - `resourceGroupName`：检测到问题的注册应用程序的资源组。
    - `status`：问题的状态。
    - `severity`：问题的严重性。
    - `problemUrl`：问题的控制台 URL。
- 问题更新。使用新观察结果更新问题或更新现有观察结果并随后更新问题时发出；更新包括问题解决方案或处理措施。
  - 详细信息类型：“已更新 Application Insights 问题”
  - 详细信息：
    - `problemId`：创建的问题 ID。
    - `region`：创建问题的 Amazon 区域。
    - `resourceGroupName`：检测到问题的注册应用程序的资源组。
    - `status`：问题的状态。
    - `severity`：问题的严重性。
    - `problemUrl`：问题的控制台 URL。

如何接收应用程序生成的问题事件的通知

从 CloudWatch 控制台中，选择 Events (事件) 下左侧导航窗格中的 Rule (规则)。从 Rules (规则) 页面中，选择 Create rule (创建规则)。选择 Service Name (服务名称) 下拉列表中的 Amazon CloudWatch Application Insights，然后选择 Event Type (事件类型)。然后，选择 Add target (添加目标)，并选择目标和参数，例如 SNS topic (SNS 主题) 或 Lambda function (Lambda 函数)。

通过 Amazon Systems Manager 执行的操作。CloudWatch Application Insights 提供与 Systems Manager OpsCenter 的内置集成。如果您选择将此集成用于您的应用程序，则会在 OpsCenter 控制台上为该应用程序检测到的每个问题创建一个 OpsItem。在 OpsCenter 控制台中，您可以查看 CloudWatch Application Insights 检测到的问题的摘要信息，并选择一个 Systems Manager Automation 运行手册来采取补救措施，或者进一步识别导致应用程序中出现资源问题的 Windows 进程。

## 使用组件配置

组件配置是 JSON 格式的文本文件，用于描述组件的配置设置。本节提供示例模板片段、组件配置部分的说明以及组件配置示例。

主题

- [组件配置模板片段 \(p. 537\)](#)
- [组件配置部分 \(p. 537\)](#)
- [组件配置示例 \(p. 543\)](#)

## 组件配置模板片段

以下示例显示 JSON 格式的模板片段。

```
{
 "alarmMetrics" : [
 list of alarm metrics
],
 "logs" : [
 list of logs
],
 "processes" : [
 list of processes
],
 "windowsEvents" : [
 list of windows events channels configurations
],
 "alarms" : [
 list of CloudWatch alarms
],
 "jmxPrometheusExporter": {
 JMX Prometheus Exporter configuration
 },
 "hanaPrometheusExporter": {
 SAP HANA Prometheus Exporter configuration
 },
 "haClusterPrometheusExporter": {
 HA Cluster Prometheus Exporter configuration
 },
 "subComponents" : [
 {
 "subComponentType" : "AWS::EC2::Instance" ...
 component nested instances configuration
 },
 {
 "subComponentType" : "AWS::EC2::Volume" ...
 component nested volumes configuration
 }
]
}
```

## 组件配置部分

组件配置包括几个主要部分。组件配置的部分可以任何顺序列出。

- alarmMetrics ( 可选 )

组件中要监控的[指标 \(p. 539\)](#)列表。所有组件类型都可以有一个 alarmMetrics 部分。

- 日志 ( 可选 )

组件中要监控的[日志 \(p. 539\)](#)列表。仅 EC2 实例可以具有日志部分。

- 进程 ( 可选 )

要为组件监控的[进程 \(p. 540\)](#)的列表。仅 EC2 实例可以具有进程部分。

- subComponents ( 可选 )

组件的嵌套实例和卷子组件配置。以下类型的组件可以具有嵌套实例和一个子组件部分：ELB、ASG、自定义分组的 EC2 实例和 EC2 实例。

- 警报 ( 可选 )

要为组件监控的[警报 \(p. 542\)](#)的列表。所有组件类型都可以有一个警报部分。

- windowsEvents ( 可选 )

要为组件监控的[Windows 事件 \(p. 542\)](#)的列表。只有 EC2 实例上的 Windows 有 windowsEvents 部分。

- JMXPrometheusExporter ( 可选 )

JMXPrometheus Exporter 配置。

- hanaPrometheusExporter ( 可选 )

SAP HANA Prometheus Exporter 配置。

- haClusterPrometheusExporter ( 可选 )

HA Cluster Prometheus Exporter 配置。

以下示例显示 JSON 格式的 subComponents 部分片段的语法。

```
[
 {
 "subComponentType" : "AWS::EC2::Instance",
 "alarmMetrics" : [
 list of alarm metrics
],
 "logs" : [
 list of logs
],
 "processes": [
 list of processes
],
 "windowsEvents" : [
 list of windows events channels configurations
]
 },
 {
 "subComponentType" : "AWS::EC2::Volume",
 "alarmMetrics" : [
 list of alarm metrics
]
 }
]
```

## 组件配置部分属性

这部分描述各部件配置部分的属性。

### 小节目录

- [指标 \(p. 539\)](#)
- [日志 \(p. 539\)](#)
- [过程 \(p. 540\)](#)
- [JMX Prometheus Exporter \(p. 540\)](#)
- [HANA Prometheus Exporter \(p. 541\)](#)
- [HA Cluster Prometheus Exporter \(p. 541\)](#)
- [Windows 事件 \(p. 542\)](#)
- [警报 \(p. 542\)](#)

## 指标

定义组件中要监控的指标。

JSON

```
{
 "alarmMetricName" : "monitoredMetricName",
 "monitor" : true/false
}
```

### 属性

- alarmMetricName ( 必需 )

组件中待监控指标的名称。有关 Application Insights 支持的指标的信息，请参阅 [Amazon CloudWatch Application Insights 支持的日志和指标 \(p. 595\)](#)。

- monitor ( 可选 )

布尔值，指示是否监控指标。默认值为 true。

## 日志

定义组件中要监控的日志。

JSON

```
{
 "logGroupName" : "logGroupName",
 "logPath" : "logPath",
 "logType" : "logType",
 "encoding" : "encodingType",
 "monitor" : true/false
}
```

### 属性

- logGroupName ( 必需 )

要与监控日志关联的 CloudWatch 日志组名称。对于日志组名称约束的信息，请参阅 [CreateLogGroup](#)。

- logPath ( EC2 实例组件需要；不使用 CloudWatch 代理的组件不需要，例如 Amazon Lambda )

要监控的日志路径。日志路径必须是 Windows 系统文件绝对路径。有关更多信息，请参阅 [CloudWatch 代理配置文件：日志部分](#)。

- logType ( 必需 )

日志类型决定 Application Insights 分析日志时所依据的日志模式。从以下选项中选择日志类型：

- SQL\_SERVER
- MYSQL
- MYSQL\_SLOW\_QUERY
- POSTGRESQL
- ORACLE\_ALERT
- ORACLE\_LISTENER
- IIS
- APPLICATION
- WINDOWS\_EVENTS

- WINDOWS\_EVENTS\_ACTIVE\_DIRECTORY
- WINDOWS\_EVENTS\_DNS
- WINDOWS\_EVENTS\_IIS
- WINDOWS\_EVENTS\_SHAREPOINT
- SQL\_SERVER\_ALWAYSON\_AVAILABILITY\_GROUP
- SQL\_SERVER\_FAILOVER\_CLUSTER\_INSTANCE
- DEFAULT
- CUSTOM
- STEP\_FUNCTION
- API\_GATEWAY\_ACCESS
- API\_GATEWAY\_EXECUTION
- SAP\_HANA\_LOGS
- SAP\_HANA\_TRACE
- SAP\_HANA\_HIGH\_AVAILABILITY
- 编码 ( 可选 )

要监控的日志的编码类型。指定的编码应包含在 [CloudWatch 代理支持的编码](#) 列表中。如果未提供，CloudWatch Application Insights 将对 utf-8 类型使用默认编码，以下除外：

- SQL\_SERVER : utf-16 编码
- IIS : ascii 编码
- 监控 ( 可选 )

布尔值，用于指示是否监控日志。默认值为 true。

## 过程

定义组件待监控的过程。

JSON

```
{
 "processName" : "monitoredProcessName",
 "alarmMetrics" : [
 list of alarm metrics
]
}
```

## 属性

- processName ( 必需 )

组件待监控的过程名称。进程名称不能包含进程主干，如 sqlservr 或 sqlservr.exe。

- alarmMetrics ( 必需 )

此进程要监控的 [指标 \(p. 539\)](#) 的列表。要查看 CloudWatch Application Insights 支持的进程指标，请参阅 [Amazon Elastic Compute Cloud \(EC2\) \(p. 597\)](#)。

## JMX Prometheus Exporter

定义 JMX Prometheus Exporter 设置。

JSON



```
"JMXPrometheusExporter": {
 "jmxURL" : "JMX URL",
 "hostPort" : "The host and port",
 "prometheusPort" : "Target port to emit Prometheus metrics"
}
```

#### 属性

- jmxURL ( 可选 )

要连接到的完整 JMX URL。

- hostPort ( 可选 )

要通过远程 JMX 连接的主机和端口。只能指定 jmxURL 和 hostPort 中的一个。

- prometheusPort ( 可选 )

要向其发送 Prometheus 指标的目标端口。如果未指定，则使用默认端口 9404。

### HANA Prometheus Exporter

定义 HANA Prometheus Exporter 设置。

#### JSON

```
"hanaPrometheusExporter": {
 "hanaSid": "SAP HANA SID",
 "hanaPort": "HANA database port",
 "hanaSecretName": "HANA secret name",
 "prometheusPort": "Target port to emit Prometheus metrics"
}
```

#### 属性

- hanaSid

SAP HANA 系统由三个字符组成的 SAP 系统 ID (SID)。

- hanaPort

导出程序将用于查询 HANA 指标的 HANA 数据库端口。

- HanaSecretName

存储 HANA 监控用户凭证的 Amazon Secrets Manager 密钥。HANA Prometheus Exporter 使用这些凭证连接到数据库并查询 HANA 指标。

- prometheusPort ( 可选 )

Prometheus 向其发送指标的目标端口。如果未指定，则使用默认端口 9668。

### HA Cluster Prometheus Exporter

定义 HA Cluster Prometheus Exporter 设置。

#### JSON

```
"haClusterPrometheusExporter": {
 "prometheusPort": "Target port to emit Prometheus metrics"
```

```
}
```

#### 属性

- prometheusPort ( 可选 )

Prometheus 向其发送指标的目标端口。如果未指定，则使用默认端口 9664。

### Windows 事件

定义要记录的 Windows 事件。

#### JSON

```
{
 "logGroupName" : "logGroupName",
 "eventName" : "eventName",
 "eventLevels" : ["ERROR", "WARNING", "CRITICAL", "INFORMATION", "VERBOSE"],
 "monitor" : true/false
}
```

#### 属性

- logGroupName ( 必需 )

要与监控日志关联的 CloudWatch 日志组名称。对于日志组名称约束的信息，请参阅 [CreateLogGroup](#)。

- eventName ( 必需 )

要记录的 Windows 事件的类型。这等同于 Windows 事件日志通道名称。例如，系统、安全、CustomEventName 等。要记录的每种类型的 Windows 事件需要使用该字段。

- eventLevels ( 必需 )

要记录的事件级别。您必须指定要录入的每个级别。可能的值包括 INFORMATION、WARNING、ERROR、CRITICAL 和 VERBOSE。要记录的每种类型的 Windows 事件需要使用该字段。

- 监控 ( 可选 )

布尔值，用于指示是否监控日志。默认值为 true。

### 警报

定义要为组件监控的 CloudWatch 告警。

#### JSON

```
{
 "alarmName" : "monitoredAlarmName",
 "severity" : HIGH/MEDIUM/LOW
}
```

#### 属性

- alarmName ( 必需 )

要为组件监控的 CloudWatch 告警的名称。

- 严重性 ( 可选 )

指示告警关闭时的中断程度。

## 组件配置示例

以下示例以 JSON 格式显示了相关服务的组件配置。

### 示例组件配置

- [Amazon Elastic Compute Cloud \(EC2\) 实例 \(p. 543\)](#)
- [Amazon Relational Database Service 实例 \(p. 545\)](#)
- [Amazon Relational Database Service \(RDS\) Aurora MySQL \(p. 545\)](#)
- [Elastic Load Balancing \(ELB\) \(p. 546\)](#)
- [Application Elastic Load Balancing \(p. 547\)](#)
- [Amazon EC2 Auto Scaling \(ASG\) \(p. 548\)](#)
- [Amazon Simple Queue Service \(SQS\) \(p. 549\)](#)
- [客户分组的 Amazon EC2 实例 \(p. 549\)](#)
- [Amazon Lambda 函数 \(p. 550\)](#)
- [Amazon DynamoDB 表 \(p. 550\)](#)
- [SQL Always On 可用性组 \(p. 551\)](#)
- [SQL 故障转移集群实例 \(p. 553\)](#)
- [RDS MariaDB 和 RDS MySQL \(p. 555\)](#)
- [RDS PostgreSQL \(p. 556\)](#)
- [Amazon S3 存储桶 \(p. 556\)](#)
- [Amazon Step Functions \(p. 556\)](#)
- [API Gateway REST API 阶段 \(p. 557\)](#)
- [Java \(p. 557\)](#)
- [RDS Oracle \(p. 558\)](#)
- [Amazon Elastic Container Service \(Amazon ECS\) \(p. 558\)](#)
- [Amazon ECS 服务 \(p. 561\)](#)
- [Amazon ECS 任务 \(p. 564\)](#)
- [Amazon EKS 集群 \(p. 564\)](#)
- [Amazon EC2 上的 Kubernetes \(p. 567\)](#)
- [Amazon FSx \(p. 569\)](#)
- [Amazon SNS 主题 \(p. 570\)](#)
- [Amazon EC2 上的 SAP HANA \(p. 570\)](#)
- [Amazon EC2 上的 SAP HANA 高可用性 \(p. 571\)](#)

## Amazon Elastic Compute Cloud (EC2) 实例

以下示例演示适用于 Amazon EC2 实例的 JSON 格式组件配置。

### Important

当 Amazon EC2 实例进入 `stopped` 状态时，表示已将它从监控中移除。当它恢复为 `running` 状态时，会将它添加到 CloudWatch Application Insights 控制台 Application details (应用程序详细信

息) 页面上的 Unmonitored components (未监控组件) 列表中。如果为应用程序启用了自动监控新资源, 则会将该实例添加到 Monitored components (已监控组件) 列表中。但会将日志和指标设置为工作负载的默认值。不会保存之前的日志和指标配置。

```
{
 "alarmMetrics" : [
 {
 "alarmMetricName" : "CPUUtilization",
 "monitor" : true
 }, {
 "alarmMetricName" : "StatusCheckFailed"
 }
],
 "logs" : [
 {
 "logGroupName" : "my_log_group",
 "logPath" : "C:\\LogFolder*",
 "logType" : "APPLICATION",
 "monitor" : true
 },
 {
 "logGroupName" : "my_log_group_2",
 "logPath" : "C:\\LogFolder2*",
 "logType" : "IIS",
 "encoding" : "utf-8"
 }
],
 "processes" : [
 {
 "processName" : "my_process",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "procstat cpu_usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "procstat memory_rss",
 "monitor" : true
 }
]
 }
],
 "windowsEvents" : [
 {
 "logGroupName" : "my_log_group_3",
 "eventName" : "Application",
 "eventLevels" : ["ERROR", "WARNING", "CRITICAL"],
 "monitor" : true
 }, {
 "logGroupName" : "my_log_group_4",
 "eventName" : "System",
 "eventLevels" : ["ERROR", "WARNING", "CRITICAL"],
 "monitor" : true
 }
],
 "alarms" : [
 {
 "alarmName" : "my_instance_alarm_1",
 "severity" : "HIGH"
 },
 {
 "alarmName" : "my_instance_alarm_2",
 "severity" : "LOW"
 }
],
 "subComponents" : [
 {
```

```

 "subComponentType" : "AWS::EC2::Volume",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "VolumeQueueLength",
 "monitor" : "true"
 },
 {
 "alarmMetricName" : "VolumeThroughputPercentage",
 "monitor" : "true"
 },
 {
 "alarmMetricName" : "BurstBalance",
 "monitor" : "true"
 }
]
 }
}

```

## Amazon Relational Database Service 实例

以下示例演示适用于 Amazon RDS 实例的 JSON 格式组件配置。

```

{
 "alarmMetrics" : [
 {
 "alarmMetricName" : "BurstBalance",
 "monitor" : true
 }, {
 "alarmMetricName" : "WriteThroughput",
 "monitor" : false
 }
],
 "alarms" : [
 {
 "alarmName" : "my_rds_instance_alarm",
 "severity" : "MEDIUM"
 }
]
}

```

## Amazon Relational Database Service (RDS) Aurora MySQL

以下示例演示适用于 Amazon RDS Aurora MySQL 的 JSON 格式组件配置。

```

{
 "alarmMetrics": [
 {
 "alarmMetricName": "CPUUtilization",
 "monitor": true
 },
 {
 "alarmMetricName": "CommitLatency",
 "monitor": true
 }
],
 "logs": [
 {
 "logType": "MYSQL",
 "monitor": true,
 },
 {
 "logType": "MYSQL_SLOW_QUERY",

```

```
 "monitor": false
 }
]
}
```

## Elastic Load Balancing (ELB)

以下示例演示适用于 Elastic Load Balancing 的 JSON 格式组件配置。

```
{
 "alarmMetrics" : [
 {
 "alarmMetricName" : "EstimatedALBActiveConnectionCount"
 }, {
 "alarmMetricName" : "HTTPCode_Backend_5XX"
 }
],
 "subComponents" : [
 {
 "subComponentType" : "AWS::EC2::Instance",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "CPUUtilization"
 }, {
 "alarmMetricName" : "StatusCheckFailed"
 }
],
 "logs" : [
 {
 "logGroupName" : "my_log_group",
 "logPath" : "C:\\LogFolder*",
 "logType" : "APPLICATION"
 }
],
 "processes" : [
 {
 "processName" : "my_process",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "procstat cpu_usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "procstat memory_rss",
 "monitor" : true
 }
]
 }
]
 }
],
 "windowsEvents" : [
 {
 "logGroupName" : "my_log_group_2",
 "eventName" : "Application",
 "eventLevels" : ["ERROR", "WARNING", "CRITICAL"],
 "monitor" : true
 }
],
 {
 "subComponentType" : "AWS::EC2::Volume",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "VolumeQueueLength"
 }, {
 "alarmMetricName" : "BurstBalance"
 }
]
 }
}
```

```
]
 }
],
"alarms" : [
 {
 "alarmName" : "my_elb_alarm",
 "severity" : "HIGH"
 }
]
}
```

## Application Elastic Load Balancing

以下示例演示适用于 Application Elastic Load Balancing 的 JSON 格式组件配置。

```
{
 "alarmMetrics" : [
 {
 "alarmMetricName" : "ActiveConnectionCount",
 }, {
 "alarmMetricName" : "TargetResponseTime"
 }
],
 "subComponents" : [
 {
 "subComponentType" : "AWS::EC2::Instance",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "CPUUtilization",
 }, {
 "alarmMetricName" : "StatusCheckFailed"
 }
],
 "logs" : [
 {
 "logGroupName" : "my_log_group",
 "logPath" : "C:\\LogFolder*",
 "logType" : "APPLICATION",
 }
],
 "windowsEvents" : [
 {
 "logGroupName" : "my_log_group_2",
 "eventName" : "Application",
 "eventLevels" : ["ERROR", "WARNING", "CRITICAL"]
 }
]
 }, {
 "subComponentType" : "AWS::EC2::Volume",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "VolumeQueueLength",
 }, {
 "alarmMetricName" : "BurstBalance"
 }
]
 }
],
 "alarms" : [
 {
 "alarmName" : "my_alb_alarm",
 "severity" : "LOW"
 }
]
}
```

```
}
]
}
```

## Amazon EC2 Auto Scaling (ASG)

以下示例演示适用于 Amazon EC2 Auto Scaling (ASG) 的 JSON 格式组件配置。

```
{
 "alarmMetrics" : [
 {
 "alarmMetricName" : "CPUCreditBalance"
 }, {
 "alarmMetricName" : "EBSIOBalance%"
 }
],
 "subComponents" : [
 {
 "subComponentType" : "AWS::EC2::Instance",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "CPUUtilization"
 }, {
 "alarmMetricName" : "StatusCheckFailed"
 }
],
 "logs" : [
 {
 "logGroupName" : "my_log_group",
 "logPath" : "C:\\LogFolder*",
 "logType" : "APPLICATION"
 }
],
 "processes" : [
 {
 "processName" : "my_process",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "procstat cpu_usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "procstat memory_rss",
 "monitor" : true
 }
]
 }
]
 }
],
 "windowsEvents" : [
 {
 "logGroupName" : "my_log_group_2",
 "eventName" : "Application",
 "eventLevels" : ["ERROR", "WARNING", "CRITICAL"]
 }
],
 {
 "subComponentType" : "AWS::EC2::Volume",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "VolumeQueueLength"
 }, {
 "alarmMetricName" : "BurstBalance"
 }
]
 }
}
```



```

],
 "alarms" : [
 {
 "alarmName" : "my_asg_alarm",
 "severity" : "LOW"
 }
]
 }

```

## Amazon Simple Queue Service (SQS)

以下示例演示适用于 Amazon Simple Queue Service 的 JSON 格式组件配置。

```

{
 "alarmMetrics" : [
 {
 "alarmMetricName" : "ApproximateAgeOfOldestMessage"
 }, {
 "alarmMetricName" : "NumberOfEmptyReceives"
 }
],
 "alarms" : [
 {
 "alarmName" : "my_sqs_alarm",
 "severity" : "MEDIUM"
 }
]
}

```

## 客户分组的 Amazon EC2 实例

以下示例演示适用于客户分组的 Amazon EC2 实例的 JSON 格式组件配置。

```

{
 "subComponents" : [
 {
 "subComponentType" : "AWS::EC2::Instance",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "CPUUtilization",
 },
 {
 "alarmMetricName" : "StatusCheckFailed"
 }
],
 "logs" : [
 {
 "logGroupName" : "my_log_group",
 "logPath" : "C:\\LogFolder*",
 "logType" : "APPLICATION",
 }
],
 "processes" : [
 {
 "processName" : "my_process",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "procstat cpu_usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "procstat memory_rss",
 "monitor" : true
 }
]
 }
]
 }
]
}

```

```
]
 },
],
 "windowsEvents" : [
 {
 "logGroupName" : "my_log_group_2",
 "eventName" : "Application",
 "eventLevels" : ["ERROR", "WARNING", "CRITICAL"]
 }
]
}, {
 "subComponentType" : "AWS::EC2::Volume",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "VolumeQueueLength",
 }, {
 "alarmMetricName" : "BurstBalance"
 }
]
}
],
"alarms" : [
 {
 "alarmName" : "my_alarm",
 "severity" : "MEDIUM"
 }
]
}
```

## Amazon Lambda 函数

以下示例演示适用于 Amazon Lambda Function 的 JSON 格式组件配置。

```
{
 "alarmMetrics": [
 {
 "alarmMetricName": "Errors",
 "monitor": true
 },
 {
 "alarmMetricName": "Throttles",
 "monitor": true
 },
 {
 "alarmMetricName": "IteratorAge",
 "monitor": true
 },
 {
 "alarmMetricName": "Duration",
 "monitor": true
 }
],
 "logs": [
 {
 "logType": "DEFAULT",
 "monitor": true
 }
]
}
```

## Amazon DynamoDB 表

以下示例演示适用于 Amazon DynamoDB 表的 JSON 格式组件配置。

```
{
 "alarmMetrics": [
 {
 "alarmMetricName": "SystemErrors",
 "monitor": false
 },
 {
 "alarmMetricName": "UserErrors",
 "monitor": false
 },
 {
 "alarmMetricName": "ConsumedReadCapacityUnits",
 "monitor": false
 },
 {
 "alarmMetricName": "ConsumedWriteCapacityUnits",
 "monitor": false
 },
 {
 "alarmMetricName": "ReadThrottleEvents",
 "monitor": false
 },
 {
 "alarmMetricName": "WriteThrottleEvents",
 "monitor": false
 },
 {
 "alarmMetricName": "ConditionalCheckFailedRequests",
 "monitor": false
 },
 {
 "alarmMetricName": "TransactionConflict",
 "monitor": false
 }
],
 "logs": []
}
```

## SQL Always On 可用性组

以下示例演示适用于 SQL Always On 可用性组的 JSON 格式组件配置。

```
{
 "subComponents" : [{
 "subComponentType" : "AWS::EC2::Instance",
 "alarmMetrics" : [{
 "alarmMetricName" : "CPUUtilization",
 "monitor" : true
 }, {
 "alarmMetricName" : "StatusCheckFailed",
 "monitor" : true
 }, {
 "alarmMetricName" : "Processor % Processor Time",
 "monitor" : true
 }, {
 "alarmMetricName" : "Memory % Committed Bytes In Use",
 "monitor" : true
 }, {
 "alarmMetricName" : "Memory Available Mbytes",
 "monitor" : true
 }, {
 "alarmMetricName" : "Paging File % Usage",
 "monitor" : true
 }, {

```

```

 "alarmMetricName" : "System Processor Queue Length",
 "monitor" : true
 }, {
 "alarmMetricName" : "Network Interface Bytes Total/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "PhysicalDisk % Disk Time",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Buffer Manager Buffer cache hit ratio",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Buffer Manager Page life expectancy",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:General Statistics Processes blocked",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:General Statistics User Connections",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Locks Number of Deadlocks/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:SQL Statistics Batch Requests/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica File Bytes Received/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Log Bytes Received/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Log remaining for undo",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Log Send Queue",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Mirrored Write Transaction/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Recovery Queue",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Redo Bytes Remaining",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Redone Bytes/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Total Log requiring undo",
 "monitor" : true
 }, {
 "alarmMetricName" : "SQLServer:Database Replica Transaction Delay",
 "monitor" : true
 }],
 "windowsEvents" : [{
 "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
 "eventName" : "Application",
 "eventLevels" : ["WARNING", "ERROR", "CRITICAL", "INFORMATION"],
 "monitor" : true
 }, {
 "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
 "eventName" : "System",
 "eventLevels" : ["WARNING", "ERROR", "CRITICAL"],

```

```

 "monitor" : true
 }, {
 "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
 "eventName" : "Security",
 "eventLevels" : ["WARNING", "ERROR", "CRITICAL"],
 "monitor" : true
 }],
 "logs" : [{
 "logGroupName" : "SQL_SERVER_ALWAYS_ON_AVAILABILITY_GROUP-<RESOURCE_GROUP_NAME>",
 "logPath" : "C:\\Program Files\\Microsoft SQL Server\\MSSQL**\\MSSQLSERVER\\MSSQL\\Log
\\ERRORLOG",
 "logType" : "SQL_SERVER",
 "monitor" : true,
 "encoding" : "utf-8"
 }]
}, {
 "subComponentType" : "AWS::EC2::Volume",
 "alarmMetrics" : [{
 "alarmMetricName" : "VolumeReadBytes",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeWriteBytes",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeReadOps",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeWriteOps",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeQueueLength",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeThroughputPercentage",
 "monitor" : true
 }, {
 "alarmMetricName" : "BurstBalance",
 "monitor" : true
 }]
}]
}

```

## SQL 故障转移集群实例

以下示例演示适用于 SQL 故障转移集群实例的 JSON 格式组件配置。

```

{
 "subComponents" : [{
 "subComponentType" : "AWS::EC2::Instance",
 "alarmMetrics" : [{
 "alarmMetricName" : "CPUUtilization",
 "monitor" : true
 }, {
 "alarmMetricName" : "StatusCheckFailed",
 "monitor" : true
 }, {
 "alarmMetricName" : "Processor % Processor Time",
 "monitor" : true
 }, {
 "alarmMetricName" : "Memory % Committed Bytes In Use",
 "monitor" : true
 }, {
 "alarmMetricName" : "Memory Available Mbytes",
 "monitor" : true
 }]
 }]
}

```

```

 }, {
 "alarmMetricName" : "Paging File % Usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "System Processor Queue Length",
 "monitor" : true
 }, {
 "alarmMetricName" : "Network Interface Bytes Total/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "PhysicalDisk % Disk Time",
 "monitor" : true
 }, {
 "alarmMetricName" : "Bytes Received/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Normal Messages Queue Length/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Urgent Message Queue Length/se",
 "monitor" : true
 }, {
 "alarmMetricName" : "Reconnect Count",
 "monitor" : true
 }, {
 "alarmMetricName" : "Unacknowledged Message Queue Length/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Messages Outstanding",
 "monitor" : true
 }, {
 "alarmMetricName" : "Messages Sent/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Database Update Messages/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Update Messages/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Flushes/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Crypto Checkpoints Saved/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Crypto Checkpoints Restored/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Registry Checkpoints Restored/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Registry Checkpoints Saved/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Cluster API Calls/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Resource API Calls/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Cluster Handles/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Resource Handles/sec",
 "monitor" : true
 }
], {
 "alarmMetricName" : "Paging File % Usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "System Processor Queue Length",
 "monitor" : true
 }, {
 "alarmMetricName" : "Network Interface Bytes Total/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "PhysicalDisk % Disk Time",
 "monitor" : true
 }, {
 "alarmMetricName" : "Bytes Received/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Normal Messages Queue Length/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Urgent Message Queue Length/se",
 "monitor" : true
 }, {
 "alarmMetricName" : "Reconnect Count",
 "monitor" : true
 }, {
 "alarmMetricName" : "Unacknowledged Message Queue Length/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Messages Outstanding",
 "monitor" : true
 }, {
 "alarmMetricName" : "Messages Sent/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Database Update Messages/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Update Messages/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Flushes/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Crypto Checkpoints Saved/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Crypto Checkpoints Restored/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Registry Checkpoints Restored/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Registry Checkpoints Saved/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Cluster API Calls/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Resource API Calls/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Cluster Handles/sec",
 "monitor" : true
 }, {
 "alarmMetricName" : "Resource Handles/sec",
 "monitor" : true
 }

```

```
 }],
 "windowsEvents" : [{
 "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
 "eventName" : "Application",
 "eventLevels" : ["WARNING", "ERROR", "CRITICAL"],
 "monitor" : true
 }, {
 "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
 "eventName" : "System",
 "eventLevels" : ["WARNING", "ERROR", "CRITICAL", "INFORMATION"],
 "monitor" : true
 }, {
 "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
 "eventName" : "Security",
 "eventLevels" : ["WARNING", "ERROR", "CRITICAL"],
 "monitor" : true
 }],
 "logs" : [{
 "logGroupName" : "SQL_SERVER_FAILOVER_CLUSTER_INSTANCE-<RESOURCE_GROUP_NAME>",
 "logPath" : "\\amznfsxjzmzbykwn.mydomain.aws\\SQLDB\\MSSQL**.*MSSQLSERVER\\MSSQL\\Log
\\ERRORLOG",
 "logType" : "SQL_SERVER",
 "monitor" : true,
 "encoding" : "utf-8"
 }]
 }, {
 "subComponentType" : "AWS::EC2::Volume",
 "alarmMetrics" : [{
 "alarmMetricName" : "VolumeReadBytes",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeWriteBytes",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeReadOps",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeWriteOps",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeQueueLength",
 "monitor" : true
 }, {
 "alarmMetricName" : "VolumeThroughputPercentage",
 "monitor" : true
 }, {
 "alarmMetricName" : "BurstBalance",
 "monitor" : true
 }]
 }]
}
```

## RDS MariaDB 和 RDS MySQL

以下示例演示适用于 RDS MariaDB 和 RDS MySQL 的 JSON 格式组件配置。

```
{
 "alarmMetrics": [
 {
 "alarmMetricName": "CPUUtilization",
 "monitor": true
 }
],
 "logs": [
```

```
{
 "logType": "MYSQL",
 "monitor": true,
},
{
 "logType": "MYSQL_SLOW_QUERY",
 "monitor": false
}
]
```

## RDS PostgreSQL

以下示例演示适用于 RDS PostgreSQL 的 JSON 格式组件配置。

```
{
 "alarmMetrics": [
 {
 "alarmMetricName": "CPUUtilization",
 "monitor": true
 }
],
 "logs": [
 {
 "logType": "POSTGRESQL",
 "monitor": true
 }
]
}
```

## Amazon S3 存储桶

以下示例演示适用于 Amazon S3 存储桶的 JSON 格式组件配置。

```
{
 "alarmMetrics" : [
 {
 "alarmMetricName" : "ReplicationLatency",
 "monitor" : true
 }, {
 "alarmMetricName" : "5xxErrors",
 "monitor" : true
 }, {
 "alarmMetricName" : "BytesDownloaded"
 "monitor" : true
 }
]
}
```

## Amazon Step Functions

以下示例演示适用于 Amazon Step Functions 的 JSON 格式组件配置。

```
{
 "alarmMetrics": [
 {
 "alarmMetricName": "ExecutionsFailed",
 "monitor": true
 },
 {
 "alarmMetricName": "LambdaFunctionsFailed",
 "monitor": true
 }
]
}
```



```
 },
 {
 "alarmMetricName": "ProvisionedRefillRate",
 "monitor": true
 }
],
 "logs": [
 {
 "logGroupName": "/aws/states/HelloWorld-Logs",
 "logType": "STEP_FUNCTION",
 "monitor": true,
 }
]
}
```

## API Gateway REST API 阶段

以下示例演示适用于 API Gateway REST API 阶段的 JSON 格式组件配置。

```
{
 "alarmMetrics" : [
 {
 "alarmMetricName" : "4XXError",
 "monitor" : true
 },
 {
 "alarmMetricName" : "5XXError",
 "monitor" : true
 }
],
 "logs" : [
 {
 "logType" : "API_GATEWAY_EXECUTION",
 "monitor" : true
 },
 {
 "logType" : "API_GATEWAY_ACCESS",
 "monitor" : true
 }
]
}
```

## Java

以下示例演示适用于 Java 的 JSON 格式组件配置。

```
{
 "alarmMetrics" : [{
 "alarmMetricName" : "java_lang_threading_threadcount",
 "monitor" : true
 },
 {
 "alarmMetricName" : "java_lang_memory_heapmemoryusage_used",
 "monitor" : true
 },
 {
 "alarmMetricName" : "java_lang_memory_heapmemoryusage_committed",
 "monitor" : true
 }
],
 "logs" : [],
 "JMXPrometheusExporter": {
 "hostPort": "8686",
 "prometheusPort": "9404"
 }
}
```

```
}
}
```

#### Note

Application Insights 不支持为 Prometheus JMX Exporter 配置身份验证。有关如何设置身份验证的信息，请参阅 [Prometheus JMX Exporter 示例配置](#)。

## RDS Oracle

以下示例演示适用于 RDS Oracle 的 JSON 格式组件配置。

```
{
 "alarmMetrics": [
 {
 "alarmMetricName": "CPUUtilization",
 "monitor": true
 }
],
 "logs": [
 {
 "logType": "ORACLE_ALERT",
 "monitor": true,
 },
 {
 "logType": "ORACLE_LISTENER",
 "monitor": false
 }
]
}
```

## Amazon Elastic Container Service (Amazon ECS)

以下示例演示适用于 Amazon Elastic Container Service (Amazon ECS) 的 JSON 格式组件配置。

```
{
 "alarmMetrics": [
 {
 "alarmMetricName": "CpuUtilized",
 "monitor": true
 },
 {
 "alarmMetricName": "MemoryUtilized",
 "monitor": true
 },
 {
 "alarmMetricName": "NetworkRxBytes",
 "monitor": true
 },
 {
 "alarmMetricName": "NetworkTxBytes",
 "monitor": true
 },
 {
 "alarmMetricName": "RunningTaskCount",
 "monitor": true
 },
 {
 "alarmMetricName": "PendingTaskCount",
 "monitor": true
 },
 {
 "alarmMetricName": "StorageReadBytes",
 "monitor": true
 }
]
}
```

```

 "monitor":true
 },
 {
 "alarmMetricName":"StorageWriteBytes",
 "monitor":true
 }
],
"logs":[
 {
 "logGroupName":"/ecs/my-task-definition",
 "logType":"APPLICATION",
 "monitor":true
 }
],
"subComponents":[
 {
 "subComponentType":"AWS::ElasticLoadBalancing::LoadBalancer",
 "alarmMetrics":[
 {
 "alarmMetricName":"HTTPCode_Backend_4XX",
 "monitor":true
 },
 {
 "alarmMetricName":"HTTPCode_Backend_5XX",
 "monitor":true
 },
 {
 "alarmMetricName":"Latency",
 "monitor":true
 },
 {
 "alarmMetricName":"SurgeQueueLength",
 "monitor":true
 },
 {
 "alarmMetricName":"UnHealthyHostCount",
 "monitor":true
 }
]
 },
 {
 "subComponentType":"AWS::ElasticLoadBalancingV2::LoadBalancer",
 "alarmMetrics":[
 {
 "alarmMetricName":"HTTPCode_Target_4XX_Count",
 "monitor":true
 },
 {
 "alarmMetricName":"HTTPCode_Target_5XX_Count",
 "monitor":true
 },
 {
 "alarmMetricName":"TargetResponseTime",
 "monitor":true
 },
 {
 "alarmMetricName":"UnHealthyHostCount",
 "monitor":true
 }
]
 },
 {
 "subComponentType":"AWS::EC2::Instance",
 "alarmMetrics":[
 {
 "alarmMetricName":"CPUUtilization",

```

```

 "monitor":true
 },
 {
 "alarmMetricName":"StatusCheckFailed",
 "monitor":true
 },
 {
 "alarmMetricName":"disk_used_percent",
 "monitor":true
 },
 {
 "alarmMetricName":"mem_used_percent",
 "monitor":true
 }
],
"logs":[
 {
 "logGroupName":"my_log_group",
 "logPath":"/mylog/path",
 "logType":"APPLICATION",
 "monitor":true
 }
],
"processes" : [
 {
 "processName" : "my_process",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "procstat cpu_usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "procstat memory_rss",
 "monitor" : true
 }
]
 }
],
"windowsEvents":[
 {
 "logGroupName":"my_log_group_2",
 "eventName":"Application",
 "eventLevels":[
 "ERROR",
 "WARNING",
 "CRITICAL"
],
 "monitor":true
 }
]
},
{
 "subComponentType":"AWS::EC2::Volume",
 "alarmMetrics":[
 {
 "alarmMetricName":"VolumeQueueLength",
 "monitor":"true"
 },
 {
 "alarmMetricName":"VolumeThroughputPercentage",
 "monitor":"true"
 },
 {
 "alarmMetricName":"BurstBalance",
 "monitor":"true"
 }
]
}
]

```

```
 }
]
}
```

#### Note

- AWS::EC2::Instance 和 AWS::EC2::Volume 的 subComponents 部分仅适用于 ECS 服务或 ECS 任务在 EC2 启动类型上运行的 Amazon ECS 集群。
- AWS::EC2::Instance 在 subComponents 的 windowsEvents 部分仅适用于 Amazon EC2 实例上运行的 Windows。

## Amazon ECS 服务

以下示例演示适用于 Amazon ECS 服务的 JSON 格式组件配置。

```
{
 "alarmMetrics":[
 {
 "alarmMetricName":"CPUUtilization",
 "monitor":true
 },
 {
 "alarmMetricName":"MemoryUtilization",
 "monitor":true
 },
 {
 "alarmMetricName":"CpuUtilized",
 "monitor":true
 },
 {
 "alarmMetricName":"MemoryUtilized",
 "monitor":true
 },
 {
 "alarmMetricName":"NetworkRxBytes",
 "monitor":true
 },
 {
 "alarmMetricName":"NetworkTxBytes",
 "monitor":true
 },
 {
 "alarmMetricName":"RunningTaskCount",
 "monitor":true
 },
 {
 "alarmMetricName":"PendingTaskCount",
 "monitor":true
 },
 {
 "alarmMetricName":"StorageReadBytes",
 "monitor":true
 },
 {
 "alarmMetricName":"StorageWriteBytes",
 "monitor":true
 }
],
 "logs":[
 {
 "logGroupName":"/ecs/my-task-definition",
 "logType":"APPLICATION",
 "monitor":true
 }
]
}
```

```

 }
],
 "subComponents":[
 {
 "subComponentType":"AWS::ElasticLoadBalancing::LoadBalancer",
 "alarmMetrics":[
 {
 "alarmMetricName":"HTTPCode_Backend_4XX",
 "monitor":true
 },
 {
 "alarmMetricName":"HTTPCode_Backend_5XX",
 "monitor":true
 },
 {
 "alarmMetricName":"Latency",
 "monitor":true
 },
 {
 "alarmMetricName":"SurgeQueueLength",
 "monitor":true
 },
 {
 "alarmMetricName":"UnHealthyHostCount",
 "monitor":true
 }
]
 },
 {
 "subComponentType":"AWS::ElasticLoadBalancingV2::LoadBalancer",
 "alarmMetrics":[
 {
 "alarmMetricName":"HTTPCode_Target_4XX_Count",
 "monitor":true
 },
 {
 "alarmMetricName":"HTTPCode_Target_5XX_Count",
 "monitor":true
 },
 {
 "alarmMetricName":"TargetResponseTime",
 "monitor":true
 },
 {
 "alarmMetricName":"UnHealthyHostCount",
 "monitor":true
 }
]
 },
 {
 "subComponentType":"AWS::EC2::Instance",
 "alarmMetrics":[
 {
 "alarmMetricName":"CPUUtilization",
 "monitor":true
 },
 {
 "alarmMetricName":"StatusCheckFailed",
 "monitor":true
 },
 {
 "alarmMetricName":"disk_used_percent",
 "monitor":true
 },
 {
 "alarmMetricName":"mem_used_percent",

```

```

 "monitor":true
 }
},
"logs":[
 {
 "logGroupName":"my_log_group",
 "logPath":"/mylog/path",
 "logType":"APPLICATION",
 "monitor":true
 }
],
"processes" : [
 {
 "processName" : "my_process",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "procstat cpu_usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "procstat memory_rss",
 "monitor" : true
 }
]
 }
]
},
"windowsEvents":[
 {
 "logGroupName":"my_log_group_2",
 "eventName":"Application",
 "eventLevels":[
 "ERROR",
 "WARNING",
 "CRITICAL"
],
 "monitor":true
 }
]
},
{
 "subComponentType":"AWS::EC2::Volume",
 "alarmMetrics":[
 {
 "alarmMetricName":"VolumeQueueLength",
 "monitor":"true"
 },
 {
 "alarmMetricName":"VolumeThroughputPercentage",
 "monitor":"true"
 },
 {
 "alarmMetricName":"BurstBalance",
 "monitor":"true"
 }
]
}
]
}
}

```

#### Note

- AWS::EC2::Instance 和 AWS::EC2::Volume 的 subComponents 部分仅适用于在 EC2 启动类型上运行的 Amazon ECS。
- AWS::EC2::Instance 在 subComponents 的 windowsEvents 部分仅适用于 Amazon EC2 实例上运行的 Windows。

## Amazon ECS 任务

以下示例演示适用于 Amazon ECS 任务的 JSON 格式组件配置。

```
{
 "logs":[
 {
 "logGroupName":"/ecs/my-task-definition",
 "logType":"APPLICATION",
 "monitor":true
 }
],
 "processes" : [
 {
 "processName" : "my_process",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "procstat cpu_usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "procstat memory_rss",
 "monitor" : true
 }
]
 }
]
}
```

## Amazon EKS 集群

以下示例演示适用于 Amazon EKS 集群的 JSON 格式组件配置。

```
{
 "alarmMetrics":[
 {
 "alarmMetricName":"cluster_failed_node_count",
 "monitor":true
 },
 {
 "alarmMetricName":"node_cpu_reserved_capacity",
 "monitor":true
 },
 {
 "alarmMetricName":"node_cpu_utilization",
 "monitor":true
 },
 {
 "alarmMetricName":"node_filesystem_utilization",
 "monitor":true
 },
 {
 "alarmMetricName":"node_memory_reserved_capacity",
 "monitor":true
 },
 {
 "alarmMetricName":"node_memory_utilization",
 "monitor":true
 },
 {
 "alarmMetricName":"node_network_total_bytes",
 "monitor":true
 },
 {

```



```

 "alarmMetricName": "pod_cpu_reserved_capacity",
 "monitor": true
 },
 {
 "alarmMetricName": "pod_cpu_utilization",
 "monitor": true
 },
 {
 "alarmMetricName": "pod_cpu_utilization_over_pod_limit",
 "monitor": true
 },
 {
 "alarmMetricName": "pod_memory_reserved_capacity",
 "monitor": true
 },
 {
 "alarmMetricName": "pod_memory_utilization",
 "monitor": true
 },
 {
 "alarmMetricName": "pod_memory_utilization_over_pod_limit",
 "monitor": true
 },
 {
 "alarmMetricName": "pod_network_rx_bytes",
 "monitor": true
 },
 {
 "alarmMetricName": "pod_network_tx_bytes",
 "monitor": true
 }
],
"logs": [
 {
 "logGroupName": "/aws/containerinsights/kubernetes/application",
 "logType": "APPLICATION",
 "monitor": true,
 "encoding": "utf-8"
 }
],
"subComponents": [
 {
 "subComponentType": "AWS::EC2::Instance",
 "alarmMetrics": [
 {
 "alarmMetricName": "CPUUtilization",
 "monitor": true
 },
 {
 "alarmMetricName": "StatusCheckFailed",
 "monitor": true
 },
 {
 "alarmMetricName": "disk_used_percent",
 "monitor": true
 },
 {
 "alarmMetricName": "mem_used_percent",
 "monitor": true
 }
],
 "logs": [
 {
 "logGroupName": "APPLICATION-KubernetesClusterOnEC2-IAD",
 "logPath": "",
 "logType": "APPLICATION",

```

```

 "monitor":true,
 "encoding":"utf-8"
 }
],
"processes" : [
 {
 "processName" : "my_process",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "procstat cpu_usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "procstat memory_rss",
 "monitor" : true
 }
]
 }
],
"windowsEvents":[
 {
 "logGroupName":"my_log_group_2",
 "eventName":"Application",
 "eventLevels":[
 "ERROR",
 "WARNING",
 "CRITICAL"
],
 "monitor":true
 }
]
},
{
 "subComponentType":"AWS::AutoScaling::AutoScalingGroup",
 "alarmMetrics":[
 {
 "alarmMetricName":"CPUCreditBalance",
 "monitor":true
 },
 {
 "alarmMetricName":"EBSIOBalance%",
 "monitor":true
 }
]
},
{
 "subComponentType":"AWS::EC2::Volume",
 "alarmMetrics":[
 {
 "alarmMetricName":"VolumeReadBytes",
 "monitor":true
 },
 {
 "alarmMetricName":"VolumeWriteBytes",
 "monitor":true
 },
 {
 "alarmMetricName":"VolumeReadOps",
 "monitor":true
 },
 {
 "alarmMetricName":"VolumeWriteOps",
 "monitor":true
 },
 {
 "alarmMetricName":"VolumeQueueLength",
 "monitor":true
 }
]
}

```

```
 },
 {
 "alarmMetricName": "BurstBalance",
 "monitor": true
 }
]
}
]
```

#### Note

- `AWS::EC2::Instance`、`AWS::EC2::Volume` 和 `AWS::AutoScaling::AutoScalingGroup` 的 `subComponents` 部分仅适用于在 EC2 启动类型上运行的 Amazon EKS 集群。
- `AWS::EC2::Instance` 在 `subComponents` 的 `windowsEvents` 部分仅适用于 Amazon EC2 实例上运行的 Windows。

## Amazon EC2 上的 Kubernetes

以下示例演示适用于 Amazon EC2 上 Kubernetes 的 JSON 格式组件配置。

```
{
 "alarmMetrics": [
 {
 "alarmMetricName": "cluster_failed_node_count",
 "monitor": true
 },
 {
 "alarmMetricName": "node_cpu_reserved_capacity",
 "monitor": true
 },
 {
 "alarmMetricName": "node_cpu_utilization",
 "monitor": true
 },
 {
 "alarmMetricName": "node_filesystem_utilization",
 "monitor": true
 },
 {
 "alarmMetricName": "node_memory_reserved_capacity",
 "monitor": true
 },
 {
 "alarmMetricName": "node_memory_utilization",
 "monitor": true
 },
 {
 "alarmMetricName": "node_network_total_bytes",
 "monitor": true
 },
 {
 "alarmMetricName": "pod_cpu_reserved_capacity",
 "monitor": true
 },
 {
 "alarmMetricName": "pod_cpu_utilization",
 "monitor": true
 },
 {
 "alarmMetricName": "pod_cpu_utilization_over_pod_limit",
```

```

 "monitor":true
 },
 {
 "alarmMetricName":"pod_memory_reserved_capacity",
 "monitor":true
 },
 {
 "alarmMetricName":"pod_memory_utilization",
 "monitor":true
 },
 {
 "alarmMetricName":"pod_memory_utilization_over_pod_limit",
 "monitor":true
 },
 {
 "alarmMetricName":"pod_network_rx_bytes",
 "monitor":true
 },
 {
 "alarmMetricName":"pod_network_tx_bytes",
 "monitor":true
 }
],
"logs":[
 {
 "logGroupName":"/aws/containerinsights/kubernetes/application",
 "logType":"APPLICATION",
 "monitor":true,
 "encoding":"utf-8"
 }
],
"subComponents":[
 {
 "subComponentType":"AWS::EC2::Instance",
 "alarmMetrics":[
 {
 "alarmMetricName":"CPUUtilization",
 "monitor":true
 },
 {
 "alarmMetricName":"StatusCheckFailed",
 "monitor":true
 },
 {
 "alarmMetricName":"disk_used_percent",
 "monitor":true
 },
 {
 "alarmMetricName":"mem_used_percent",
 "monitor":true
 }
],
 "logs":[
 {
 "logGroupName":"APPLICATION-KubernetesClusterOnEC2-IAD",
 "logPath":"",
 "logType":"APPLICATION",
 "monitor":true,
 "encoding":"utf-8"
 }
],
 "processes" : [
 {
 "processName" : "my_process",
 "alarmMetrics" : [
 {

```

```

 "alarmMetricName" : "procstat cpu_usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "procstat memory_rss",
 "monitor" : true
 }
]
}
]
},
{
 "subComponentType": "AWS::EC2::Volume",
 "alarmMetrics": [
 {
 "alarmMetricName": "VolumeReadBytes",
 "monitor": true
 },
 {
 "alarmMetricName": "VolumeWriteBytes",
 "monitor": true
 },
 {
 "alarmMetricName": "VolumeReadOps",
 "monitor": true
 },
 {
 "alarmMetricName": "VolumeWriteOps",
 "monitor": true
 },
 {
 "alarmMetricName": "VolumeQueueLength",
 "monitor": true
 },
 {
 "alarmMetricName": "BurstBalance",
 "monitor": true
 }
]
}
]
}
}

```

## Amazon FSx

以下示例演示适用于 Amazon FSx 的 JSON 格式组件配置。

```

{
 "alarmMetrics": [
 {
 "alarmMetricName": "DataReadBytes",
 "monitor": true
 },
 {
 "alarmMetricName": "DataWriteBytes",
 "monitor": true
 },
 {
 "alarmMetricName": "DataReadOperations",
 "monitor": true
 },
 {
 "alarmMetricName": "DataWriteOperations",
 "monitor": true
 }
]
}

```

```
{
 "alarmMetricName": "MetadataOperations",
 "monitor": true
},
{
 "alarmMetricName": "FreeStorageCapacity",
 "monitor": true
}
]
```

## Amazon SNS 主题

以下示例演示适用于 Amazon SNS 主题的 JSON 格式组件配置。

```
{
 "alarmMetrics": [
 {
 "alarmMetricName": "NumberOfNotificationsFailed",
 "monitor": true
 },
 {
 "alarmMetricName": "NumberOfNotificationsFilteredOut-InvalidAttributes",
 "monitor": true
 },
 {
 "alarmMetricName": "NumberOfNotificationsFilteredOut-NoMessageAttributes",
 "monitor": true
 },
 {
 "alarmMetricName": "NumberOfNotificationsFailedToRedriveToDlq",
 "monitor": true
 }
]
}
```

## Amazon EC2 上的 SAP HANA

以下示例演示适用于 Amazon EC2 上 SAP HANA 的 JSON 格式组件配置。

```
{
 "subComponents": [
 {
 "subComponentType": "AWS::EC2::Instance",
 "alarmMetrics": [
 {
 "alarmMetricName": "hanadb_server_startup_time_variations_seconds",
 "monitor": true
 },
 {
 "alarmMetricName": "hanadb_level_5_alerts_count",
 "monitor": true
 },
 {
 "alarmMetricName": "hanadb_level_4_alerts_count",
 "monitor": true
 },
 {
 "alarmMetricName": "hanadb_out_of_memory_events_count",
 "monitor": true
 },
 {

```

```

 "alarmMetricName": "hanadb_max_trigger_read_ratio_percent",
 "monitor": true
 },
 {
 "alarmMetricName": "hanadb_table_allocation_limit_used_percent",
 "monitor": true
 },
 {
 "alarmMetricName": "hanadb_cpu_usage_percent",
 "monitor": true
 },
 {
 "alarmMetricName": "hanadb_plan_cache_hit_ratio_percent",
 "monitor": true
 },
 {
 "alarmMetricName": "hanadb_last_data_backup_age_days",
 "monitor": true
 }
],
"logs": [
 {
 "logGroupName": "SAP_HANA_TRACE-my-resource-group",
 "logPath": "/usr/sap/HDB/HDB00/*/trace/*.trc",
 "logType": "SAP_HANA_TRACE",
 "monitor": true,
 "encoding": "utf-8"
 },
 {
 "logGroupName": "SAP_HANA_LOGS-my-resource-group",
 "logPath": "/usr/sap/HDB/HDB00/*/trace/*.log",
 "logType": "SAP_HANA_LOGS",
 "monitor": true,
 "encoding": "utf-8"
 }
]
},
"hanaPrometheusExporter": {
 "hanaSid": "HDB",
 "hanaPort": "30013",
 "hanaSecretName": "HANA_DB_CREDS",
 "prometheusPort": "9668"
}
}

```

## Amazon EC2 上的 SAP HANA 高可用性

以下示例演示适用于 Amazon EC2 上 SAP HANA 高可用性的 JSON 格式组件配置。

```

{
 "subComponents": [
 {
 "subComponentType": "AWS::EC2::Instance",
 "alarmMetrics": [
 {
 "alarmMetricName": "hanadb_server_startup_time_variations_seconds",
 "monitor": true
 },
 {
 "alarmMetricName": "hanadb_level_5_alerts_count",
 "monitor": true
 },
 {

```

```
 "alarmMetricName": "hanadb_level_4_alerts_count",
 "monitor": true
 },
 {
 "alarmMetricName": "hanadb_out_of_memory_events_count",
 "monitor": true
 },
 {
 "alarmMetricName": "ha_cluster_pacemaker_stonith_enabled",
 "monitor": true
 }
],
"logs": [
 {
 "logGroupName": "SAP_HANA_TRACE-my-resource-group",
 "logPath": "/usr/sap/HDB/HDB00/*/trace/*.trc",
 "logType": "SAP_HANA_TRACE",
 "monitor": true,
 "encoding": "utf-8"
 },
 {
 "logGroupName": "SAP_HANA_HIGH_AVAILABILITY-my-resource-group",
 "logPath": "/var/log/pacemaker/pacemaker.log",
 "logType": "SAP_HANA_HIGH_AVAILABILITY",
 "monitor": true,
 "encoding": "utf-8"
 }
]
},
"hanaPrometheusExporter": {
 "hanaSid": "HDB",
 "hanaPort": "30013",
 "hanaSecretName": "HANA_DB_CREDS",
 "prometheusPort": "9668"
},
"haClusterPrometheusExporter": {
 "prometheusPort": "9664"
}
}
```

## 使用 CloudFormation 模板创建和配置 CloudWatch Application Insights 监控

您可以直接从 Amazon CloudFormation 模板中将 Application Insights 监控（包括关键指标和遥测）添加到应用程序、数据库和 Web 服务器。

此部分提供 JSON 和 YAML 格式的示例 Amazon CloudFormation 模板，以帮助您创建和配置 Application Insights 监控。

要查看 Amazon CloudFormation 用户指南中的 Application Insights 资源和属性参考，请参阅 [ApplicationInsights 资源类型参考](#)。

### 示例模板

- [为整个 Amazon CloudFormation 堆栈创建 Application Insights 应用程序 \(p. 573\)](#)
- [创建具有详细设置的 Application Insights 应用程序 \(p. 574\)](#)
- [使用 CUSTOM 模式组件配置创建 Application Insights 应用程序 \(p. 576\)](#)
- [使用 DEFAULT 模式组件配置创建 Application Insights 应用程序 \(p. 579\)](#)
- [使用 DEFAULT\\_WITH\\_OVERWRITE 模式组件配置创建 Application Insights 应用程序 \(p. 580\)](#)



## 为整个 Amazon CloudFormation 堆栈创建 Application Insights 应用程序

要应用以下模板，必须创建 Amazon 资源和一个或多个资源组，用于创建 Application Insights 应用程序以监控这些资源。有关更多信息，请参阅 [Amazon Resource Groups 入门](#)。

以下模板的前两部分指定资源和资源组。模板的最后一部分为资源组创建 Application Insights 应用程序，但不配置应用程序或应用监控。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [CreateApplication](#) 命令详细信息。

JSON 格式的模板

```
{
 "AWSTemplateFormatVersion": "2010-09-09",
 "Description": "Test Resource Group stack",
 "Resources": {
 "EC2Instance": {
 "Type": "AWS::EC2::Instance",
 "Properties": {
 "ImageId": "ami-abcd1234efgh5678i",
 "SecurityGroupIds": ["sg-abcd1234"]
 }
 },
 ...
 "ResourceGroup": {
 "Type": "AWS::ResourceGroups::Group",
 "Properties": {
 "Name": "my_resource_group"
 }
 },
 "AppInsightsApp": {
 "Type": "AWS::ApplicationInsights::Application",
 "Properties": {
 "ResourceGroupName": "my_resource_group"
 },
 "DependsOn": "ResourceGroup"
 }
 }
}
```

YAML 格式的模板

```

AWSTemplateFormatVersion: '2010-09-09'
Description: Test Resource Group stack
Resources:
 EC2Instance:
 Type: AWS::EC2::Instance
 Properties:
 ImageId: ami-abcd1234efgh5678i
 SecurityGroupIds:
 - sg-abcd1234
 ...
 ResourceGroup:
 Type: AWS::ResourceGroups::Group
 Properties:
 Name: my_resource_group
 AppInsightsApp:
 Type: AWS::ApplicationInsights::Application
 Properties:
 ResourceGroupName: my_resource_group
```

```
DependsOn: ResourceGroup
```

以下模板部分将默认监控配置应用于 Application Insights 应用程序。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [CreateApplication](#) 命令详细信息。

当 `AutoConfigurationEnabled` 设置为 `true` 时，将使用 `DEFAULT` 应用程序层的建议监控设置来配置应用程序的所有组件。有关这些设置和层的更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [DescribeComponentConfigurationRecommendation](#) 和 [UpdateComponentConfiguration](#)。

JSON 格式的模板

```
{
 "AWSTemplateFormatVersion": "2010-09-09",
 "Description": "Test Application Insights Application stack",
 "Resources": {
 "AppInsightsApp": {
 "Type": "AWS::ApplicationInsights::Application",
 "Properties": {
 "ResourceGroupName": "my_resource_group",
 "AutoConfigurationEnabled": true
 }
 }
 }
}
```

YAML 格式的模板

```

AWSTemplateFormatVersion: '2010-09-09'
Description: Test Application Insights Application stack
Resources:
 AppInsightsApp:
 Type: AWS::ApplicationInsights::Application
 Properties:
 ResourceGroupName: my_resource_group
 AutoConfigurationEnabled: true
```

## 创建具有详细设置的 Application Insights 应用程序

以下模板执行以下操作：

- 创建启用了 CloudWatch Events 通知并启用 OpsCenter 的 Application Insights 应用程序。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [CreateApplication](#) 命令详细信息。
- 使用两个标签来标记应用程序，其中一个标签没有标签值。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [TagResource](#)。
- 创建两个自定义实例组组件。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [CreateComponent](#)。
- 创建两个日志模式集。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [CreateLogPattern](#)。
- 将 `AutoConfigurationEnabled` 设置为 `true`，这会使用 `DEFAULT` 层的建议监控设置来配置应用程序的所有组件。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [DescribeComponentConfigurationRecommendation](#)。

JSON 格式的模板

```
{
 "Type": "AWS::ApplicationInsights::Application",
```

```

"Properties": {
 "ResourceGroupName": "my_resource_group",
 "CWEMonitorEnabled": true,
 "OpsCenterEnabled": true,
 "OpsItemSNSTopicArn": "arn:aws-cn:sns:cn-north-1:123456789012:my_topic",
 "AutoConfigurationEnabled": true,
 "Tags": [
 {
 "Key": "key1",
 "Value": "value1"
 },
 {
 "Key": "key2",
 "Value": ""
 }
],
 "CustomComponents": [
 {
 "ComponentName": "test_component_1",
 "ResourceList": [
 "arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-abcd1234efgh5678i"
]
 },
 {
 "ComponentName": "test_component_2",
 "ResourceList": [
 "arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-abcd1234efgh5678i",
 "arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-abcd1234efgh5678i"
]
 }
],
 "LogPatternSets": [
 {
 "PatternSetName": "pattern_set_1",
 "LogPatterns": [
 {
 "PatternName": "deadlock_pattern",
 "Pattern": ".*\\sDeadlocked\\sSchedulers(([^\\w].*)|($))",
 "Rank": 1
 }
]
 },
 {
 "PatternSetName": "pattern_set_2",
 "LogPatterns": [
 {
 "PatternName": "error_pattern",
 "Pattern": ".*[\\s\\[\\]ERROR[\\s\\]].*",
 "Rank": 1
 },
 {
 "PatternName": "warning_pattern",
 "Pattern": ".*[\\s\\[\\]WARN(ING)?[\\s\\]].*",
 "Rank": 10
 }
]
 }
]
}
}

```

#### YAML 格式的模板

```

Type: AWS::ApplicationInsights::Application

```

```
Properties:
 ResourceGroupName: my_resource_group
 CWEMonitorEnabled: true
 OpsCenterEnabled: true
 OpsItemSNSTopicArn: arn:aws-cn:sns:cn-north-1:123456789012:my_topic
 AutoConfigurationEnabled: true
 Tags:
 - Key: key1
 Value: value1
 - Key: key2
 Value: ''
 CustomComponents:
 - ComponentName: test_component_1
 ResourceList:
 - arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-abcd1234efgh5678i
 - ComponentName: test_component_2
 ResourceList:
 - arn:aws-cn:ec2:cn-north-1:123456789012:instance/i-abcd1234efgh5678i
 LogPatternSets:
 - PatternSetName: pattern_set_1
 LogPatterns:
 - PatternName: deadlock_pattern
 Pattern: ".*\\sDeadlocked\\sSchedulers((^[^\\w].*)|($))"
 Rank: 1
 - PatternSetName: pattern_set_2
 LogPatterns:
 - PatternName: error_pattern
 Pattern: ".*[\\s\\[\\]ERROR[\\s\\]].*"
 Rank: 1
 - PatternName: warning_pattern
 Pattern: ".*[\\s\\[\\]WARN(ING)?[\\s\\]].*"
 Rank: 10
```

## 使用 CUSTOM 模式组件配置创建 Application Insights 应用程序

以下模板执行以下操作：

- 创建 Application Insights 应用程序。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [CreateApplication](#)。
- 组件 my\_component 将 ComponentConfigurationMode 设置为 CUSTOM，这将导致按照 CustomComponentConfiguration 中指定的配置来配置此组件。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [UpdateComponentConfiguration](#)。

JSON 格式的模板

```
{
 "Type": "AWS::ApplicationInsights::Application",
 "Properties": {
 "ResourceGroupName": "my_resource_group",
 "ComponentMonitoringSettings": [
 {
 "ComponentARN": "my_component",
 "Tier": "SQL_SERVER",
 "ComponentConfigurationMode": "CUSTOM",
 "CustomComponentConfiguration": {
 "ConfigurationDetails": {
 "AlarmMetrics": [
 {
 "AlarmMetricName": "StatusCheckFailed"
 },
 ...
]
 }
 }
 }
]
 }
}
```

```

],
 "Logs": [
 {
 "LogGroupName": "my_log_group_1",
 "LogPath": "C:\\LogFolder_1*",
 "LogType": "DOT_NET_CORE",
 "Encoding": "utf-8",
 "PatternSet": "my_pattern_set_1"
 },
 ...
],
 "WindowsEvents": [
 {
 "LogGroupName": "my_windows_event_log_group_1",
 "EventName": "Application",
 "EventLevels": [
 "ERROR",
 "WARNING",
 ...
],
 "Encoding": "utf-8",
 "PatternSet": "my_pattern_set_2"
 },
 ...
],
 "Alarms": [
 {
 "AlarmName": "my_alarm_name",
 "Severity": "HIGH"
 },
 ...
]
},
"SubComponentTypeConfigurations": [
 {
 "SubComponentType": "EC2_INSTANCE",
 "SubComponentConfigurationDetails": {
 "AlarmMetrics": [
 {
 "AlarmMetricName": "DiskReadOps"
 },
 ...
],
 "Logs": [
 {
 "LogGroupName": "my_log_group_2",
 "LogPath": "C:\\LogFolder_2*",
 "LogType": "IIS",
 "Encoding": "utf-8",
 "PatternSet": "my_pattern_set_3"
 },
 ...
],
 "processes" : [
 {
 "processName" : "my_process",
 "alarmMetrics" : [
 {
 "alarmMetricName" : "procstat cpu_usage",
 "monitor" : true
 }, {
 "alarmMetricName" : "procstat memory_rss",
 "monitor" : true
 }
]
 }
]
 }
 }
]

```

```

],
 "WindowsEvents": [
 {
 "LogGroupName": "my_windows_event_log_group_2",
 "EventName": "Application",
 "EventLevels": [
 "ERROR",
 "WARNING",
 ...
],
 "Encoding": "utf-8",
 "PatternSet": "my_pattern_set_4"
 },
 ...
]
 }
}

```

#### YAML 格式的模板

```

Type: AWS::ApplicationInsights::Application
Properties:
 ResourceGroupName: my_resource_group
 ComponentMonitoringSettings:
 - ComponentARN: my_component
 Tier: SQL_SERVER
 ComponentConfigurationMode: CUSTOM
 CustomComponentConfiguration:
 ConfigurationDetails:
 AlarmMetrics:
 - AlarmMetricName: StatusCheckFailed
 ...
 Logs:
 - LogGroupName: my_log_group_1
 LogPath: C:\LogFolder_1*
 LogType: DOT_NET_CORE
 Encoding: utf-8
 PatternSet: my_pattern_set_1
 ...
 WindowsEvents:
 - LogGroupName: my_windows_event_log_group_1
 EventName: Application
 EventLevels:
 - ERROR
 - WARNING
 ...
 Encoding: utf-8
 PatternSet: my_pattern_set_2
 ...
 Alarms:
 - AlarmName: my_alarm_name
 Severity: HIGH
 ...
 SubComponentTypeConfigurations:
 - SubComponentType: EC2_INSTANCE
 SubComponentConfigurationDetails:
 AlarmMetrics:
 - AlarmMetricName: DiskReadOps

```

```
...
Logs:
- LogGroupName: my_log_group_2
 LogPath: C:\LogFolder_2*
 LogType: IIS
 Encoding: utf-8
 PatternSet: my_pattern_set_3
...
Processes:
- ProcessName: my_process
 AlarmMetrics:
 - AlarmMetricName: procstat cpu_usage
 ...
 ...
WindowsEvents:
- LogGroupName: my_windows_event_log_group_2
 EventName: Application
 EventLevels:
 - ERROR
 - WARNING
 ...
 Encoding: utf-8
 PatternSet: my_pattern_set_4
...
```

## 使用 DEFAULT 模式组件配置创建 Application Insights 应用程序

以下模板执行以下操作：

- 创建 Application Insights 应用程序。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [CreateApplication](#)。
- 组件 my\_component 将 ComponentConfigurationMode 设置为 DEFAULT，并将 Tier 设置为 SQL\_SERVER，这将导致使用 Application Insights 为 SQL\_Server 层推荐的配置设置来配置此组件。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [DescribeComponentConfiguration](#) 和 [UpdateComponentConfiguration](#)。

JSON 格式的模板

```
{
 "Type": "AWS::ApplicationInsights::Application",
 "Properties": {
 "ResourceGroupName": "my_resource_group",
 "ComponentMonitoringSettings": [
 {
 "ComponentARN": "my_component",
 "Tier": "SQL_SERVER",
 "ComponentConfigurationMode": "DEFAULT"
 }
]
 }
}
```

YAML 格式的模板

```

Type: AWS::ApplicationInsights::Application
Properties:
 ResourceGroupName: my_resource_group
 ComponentMonitoringSettings:
 - ComponentARN: my_component
```

```
Tier: SQL_SERVER
ComponentConfigurationMode: DEFAULT
```

## 使用 DEFAULT\_WITH\_OVERWRITE 模式组件配置创建 Application Insights 应用程序

以下模板执行以下操作：

- 创建 Application Insights 应用程序。有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [CreateApplication](#)。
- 组件 my\_component 将 ComponentConfigurationMode 设置为 DEFAULT\_WITH\_OVERWRITE，并将 tier 设置为 DOT\_NET\_CORE，这将导致使用 Application Insights 为 DOT\_NET\_CORE 层推荐的配置设置来配置此组件。覆盖的配置设置在 DefaultOverwriteComponentConfiguration 中指定：
  - 在组件级别，AlarmMetrics 设置将被覆盖。
  - 在子组件级别，对于 EC2\_Instance 类型的子组件，将覆盖 Logs 设置。

有关更多信息，请参阅 Amazon CloudWatch Application Insights API 参考中的 [UpdateComponentConfiguration](#)。

JSON 格式的模板

```
{
 "Type": "AWS::ApplicationInsights::Application",
 "Properties": {
 "ResourceGroupName": "my_resource_group",
 "ComponentMonitoringSettings": [
 {
 "ComponentName": "my_component",
 "Tier": "DOT_NET_CORE",
 "ComponentConfigurationMode": "DEFAULT_WITH_OVERWRITE",
 "DefaultOverwriteComponentConfiguration": {
 "ConfigurationDetails": {
 "AlarmMetrics": [
 {
 "AlarmMetricName": "StatusCheckFailed"
 }
]
 },
 "SubComponentTypeConfigurations": [
 {
 "SubComponentType": "EC2_INSTANCE",
 "SubComponentConfigurationDetails": {
 "Logs": [
 {
 "LogGroupName": "my_log_group",
 "LogPath": "C:\\LogFolder*",
 "LogType": "IIS",
 "Encoding": "utf-8",
 "PatternSet": "my_pattern_set"
 }
]
 }
 }
]
 }
]
]
 }
}
```



## YAML 格式的模板

```

Type: AWS::ApplicationInsights::Application
Properties:
 ResourceGroupName: my_resource_group
 ComponentMonitoringSettings:
 - ComponentName: my_component
 Tier: DOT_NET_CORE
 ComponentConfigurationMode: DEFAULT_WITH_OVERWRITE
 DefaultOverwriteComponentConfiguration:
 ConfigurationDetails:
 AlarmMetrics:
 - AlarmMetricName: StatusCheckFailed
 SubComponentTypeConfigurations:
 - SubComponentType: EC2_INSTANCE
 SubComponentConfigurationDetails:
 Logs:
 - LogGroupName: my_log_group
 LogPath: C:\LogFolder*
 LogType: IIS
 Encoding: utf-8
 PatternSet: my_pattern_set
```

## 教程：为 SAP HANA 设置监控

本教程演示了如何配置 CloudWatch Application Insights 来为 SAP HANA 数据库设置监控。您可以使用 CloudWatch Application Insights 自动控制面板可视化呈现问题详细信息、加快故障排除速度及缩短 SAP HANA 数据库的平均解决时间 (MTTR)。

### Application Insights for SAP HANA 主题

- [支持的环境 \(p. 581\)](#)
- [支持的操作系统 \(p. 582\)](#)
- [功能 \(p. 582\)](#)
- [先决条件 \(p. 582\)](#)
- [设置 SAP HANA 数据库以进行监控 \(p. 584\)](#)
- [管理对 SAP HANA 数据库的监控 \(p. 584\)](#)
- [配置告警阈值 \(p. 585\)](#)
- [查看和排查 CloudWatch Application Insights 检测到的问题 \(p. 585\)](#)
- [SAP HANA 的异常检测 \(p. 587\)](#)
- [排查 Application Insights for SAP HANA 的问题 \(p. 588\)](#)

## 支持的环境

CloudWatch Application Insights 支持部署用于以下系统和模式的 Amazon 资源。您提供并安装 SAP HANA 数据库软件和支持的 SAP 应用程序软件。

- 单个 Amazon EC2 实例上的 SAP HANA 数据库 – 采用单节点、纵向扩展架构的 SAP HANA，内存高达 24TB。
- 多个 Amazon EC2 实例上的 SAP HANA 数据库 – 采用多节点、横向扩展架构的 SAP HANA。
- 跨可用区 SAP HANA 数据库高可用性设置 – 使用 SUSE/RHEL 集群在两个可用区中配置了高可用性的 SAP HANA。

## Note

CloudWatch Application Insights 仅支持单个 SID HANA 环境。如果连接了多个 HANA SID，则只会为第一个检测到的 SID 设置监控。

## 支持的操作系统

CloudWatch Application Insights for SAP HANA 在以下操作系统中支持 x86-64 架构：

- SuSE Linux 12 SP4 For SAP
- SuSE Linux 12 SP5 For SAP
- SUSE Linux 15
- SuSE Linux 15 SP1
- SuSE Linux 15 SP2
- SuSE Linux 15 For SAP
- SuSE Linux 15 SP1 For SAP
- SuSE Linux 15 SP2 For SAP
- RedHat Linux 8.2 For SAP ( 包含高可用性和更新服务 )
- RedHat Linux 8.1 For SAP ( 包含高可用性和更新服务 )
- RedHat Linux 7.9 For SAP ( 包含高可用性和更新服务 )

## 功能

CloudWatch Application Insights for SAP HANA 具有以下功能：

- SAP HANA 工作负载自动检测
- 基于静态阈值自动创建 SAP HANA 告警
- 基于异常检测自动创建 SAP HANA 告警
- 自动识别 SAP HANA 日志模式
- SAP HANA 健康控制面板
- SAP HANA 问题控制面板

## 先决条件

您必须完成以下必需任务才能使用 CloudWatch Application Insights 配置 SAP HANA 数据库：

- SAP HANA – 在 Amazon EC2 实例上安装处于运行状态且可访问的 SAP HANA 数据库 2.0 SPS05。
- SAP HANA 数据库用户 — 必须在 SYSTEM 数据库和所有租户中创建具有监控角色的数据库用户。

### 示例

以下 SQL 命令将创建具有监控角色的用户。

```
su - <sid>adm
hdbsql -u SYSTEM -p <SYSTEMDB password> -d SYSTEMDB
CREATE USER CW_HANADB_EXPORTER_USER PASSWORD <Monitoring user password> NO
FORCE_FIRST_PASSWORD_CHANGE;
CREATE ROLE CW_HANADB_EXPORTER_ROLE;
GRANT MONITORING TO CW_HANADB_EXPORTER_ROLE;
GRANT CW_HANADB_EXPORTER_ROLE TO CW_HANADB_EXPORTER_USER;
```

- Python 3.6 – 在操作系统上安装 Python 3.6 或更高版本。使用最新的 Python 版本。如果未在操作系统中检测到 Python，则将安装 Python 3.8。

请参阅 [installation examples](#)。

- Pip3 – 在操作系统中安装安装程序 pip3。如果未在操作系统中检测到 pip3，则安装该程序。
- Amazon CloudWatch 代理 – 确保 Amazon EC2 实例上没有运行预先已有的 Amazon CloudWatch 代理。
- Amazon Systems Manager 启用 – 在实例上安装 SSM Agent，并且实例必须启用了 SSM。有关如何安装 SSM Agent 的信息，请参阅 Amazon Systems Manager 用户指南中的 [使用 SSM Agent](#)。
- Amazon EC2 实例角色 – 要配置数据库，您必须附加以下 Amazon EC2 实例角色。
  - 要启用 Systems Manager，您必须附加 AmazonSSMManagedInstanceCore 角色。有关更多信息，请参阅 [Amazon Systems Manager 基于身份的策略示例](#)。
  - 要启用通过 CloudWatch 发出实例指标和日志，您必须附加 CloudWatchAgentServerPolicy。有关更多信息，请参阅 [创建用于 CloudWatch 代理的 IAM 角色和用户](#)。
  - 要读取存储在 Amazon Secrets Manager 中的密码，您必须将以下 IAM 内联策略附加到 Amazon EC2 实例角色。有关内联策略的更多信息，请参阅 Amazon Identity and Access Management 用户指南中的 [内联策略](#)。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "VisualEditor0",
 "Effect": "Allow",
 "Action": [
 "secretsmanager:GetSecretValue"
],
 "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
 }
]
}
```

- Amazon Resource Groups – 要将应用程序添加到 CloudWatch Application Insights，您必须创建一个包含应用程序堆栈所用的所有相关 Amazon 资源的资源组。这包括运行 SAP HANA 数据库的 Amazon EC2 实例和 Amazon EBS 卷。如果每个账户有多个数据库，我们建议您创建一个资源组，而且该资源组包含每个 SAP HANA 数据库系统的 Amazon 资源。
- IAM 权限 — 对于非管理员用户：
  - 您必须创建允许 Application Insights 创建服务相关角色的 Amazon Identity and Access Management (IAM) 策略，并将其附加到您的用户身份。有关附加策略的步骤，请参阅 [IAM policy \(p. 519\)](#)。
  - 用户必须有在 Amazon Secrets Manager 中创建密钥的权限，以存储数据库用户凭证。有关更多信息，请参阅 [示例：创建密钥的权限](#)。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "secretsmanager:CreateSecret"
],
 "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
 }
]
}
```

- 服务相关角色 – Application Insights 使用 Amazon Identity and Access Management (IAM) 服务相关角色。当您在 Application Insights 控制台中创建首个 Application Insights 应用程序时，将会为您创建服务相关角色。有关更多信息，请参阅在 [CloudWatch Application Insights 中使用服务相关角色 \(p. 871\)](#)。

## 设置 SAP HANA 数据库以进行监控

要为 SAP HANA 数据库设置监控，请执行以下步骤。

1. 导航到 [CloudWatch 控制台](#)。
2. 从左侧导航窗格中，选择 Insights 下的 Application Insights。
3. Application Insights 页面会显示使用 Application Insights 监控的应用程序列表以及每个应用程序的监控状态。在右上角，选择 Add an application (添加应用程序)。
4. 在 Specify application details (指定应用程序详细信息) 页面上，从 Resource group (资源组) 下的下拉列表中选择包含 SAP HANA 数据库资源的 Amazon 资源组。如果尚未为应用程序创建资源组，则可以在 Resource group (资源组) 下拉列表下选择 Create new resource group (创建新资源组) 来创建一个资源组。有关创建资源组的更多信息，请参阅 [Amazon Resource Groups 用户指南](#)。
5. 在 Monitor CloudWatch Events (监控 CloudWatch Events) 下，选中复选框将 Application Insights 监控与 CloudWatch Events 集成，以获取 Amazon EBS、Amazon EC2、Amazon CodeDeploy、Amazon ECS、Amazon Health API 和通知、Amazon RDS、Amazon S3 和 Amazon Step Functions 中的洞察。
6. 在 Integrate with Amazon Systems Manager OpsCenter (与 OpsCenter 集成) 下，选中 Generate Amazon Systems Manager OpsCenter OpsItems for remedial actions (生成 OpsCenter OpsItems 以采取修复措施) 旁边的复选框，以在检测到所选应用程序的问题时查看问题及接收通知。要跟踪为解析与 Amazon 资源相关的操作工作项 (OpsItem) 而执行的操作，请提供 SNS 主题 ARN。
7. 您可以选择输入标签以帮助您标识和整理资源。CloudWatch Application Insights 支持基于标签和基于 Amazon CloudFormation 的资源组 (Application Auto Scaling 组除外)。有关更多信息，请参阅 [Amazon Resource Groups](#) 和 [标签用户指南](#) 中的 [标签编辑器](#)。
8. 选择 Next (下一步) 继续设置监控。
9. Set up monitoring (设置监控) 页面中列出了 CloudWatch Application Insights 自动检测到的受监控组件。选择 Next (下一步)。
10. 在 Specify component details (指定组件详细信息) 页面上，输入用户的用户名和密码。
11. 选择 Next (下一步)。
12. 查看应用程序监控配置，然后选择 Submit (提交)。
13. 此时将打开应用程序详细信息页面，您可以在该页面中查看 Application summary (应用程序摘要)，以及 Monitored components (已监控组件) 和 Unmonitored components (未监控组件) 的列表。选择组件旁边的单选按钮，还可以查看 Configuration history (配置历史记录)、Log patterns (日志模式) 和已创建的任何 Tags (标签)。提交配置时，您的账户会为 SAP HANA 系统部署所有指标和告警，这最多可能需要 2 个小时。

## 管理对 SAP HANA 数据库的监控

您可以执行以下步骤来管理 SAP HANA 数据库的用户凭证、指标和日志路径：

1. 导航到 [CloudWatch 控制台](#)。
2. 从左侧导航窗格中，选择 Insights 下的 Application Insights。
3. Application Insights 页面会显示使用 Application Insights 监控的应用程序列表以及每个应用程序的监控状态。
4. 在 Monitored components (已监控组件) 下，选择组件名称旁边的单选按钮。然后，选择 Manage monitoring (管理监控)。
5. 在 EC2 instance group logs (EC2 实例组日志) 下，您可以更新现有的日志路径、日志模式集和日志组名称。此外，您可以额外添加最多三个 Application logs (应用程序日志)。
6. 在 Metrics (指标) 下，您可以根据自己的要求选择 SAP HANA 指标。SAP HANA 指标名称的前缀是 hanadb。每个组件可以添加最多 40 个指标。

7. 在 HANA configuration ( HANA 配置 ) 下，输入 SAP HANA 数据库的密码和用户名，即 Amazon CloudWatch 代理连接到 SAP HANA 数据库时使用的用户名和密码。
8. 在 Custom alarms ( 自定义告警 ) 下，您可以添加可由 CloudWatch Application Insights 监控的额外告警。
9. 查看应用程序监控配置并选择 Submit ( 提交 )。提交配置时，您的账户会为 SAP HANA 系统更新所有指标和告警，这最多可能需要 2 个小时。

## 配置告警阈值

CloudWatch Application Insights 会自动创建由告警监控的 Amazon CloudWatch 指标，以及该指标的阈值。当该指标在指定数量的评估期内超出阈值，告警将变为 ALARM ( 告警 ) 状态。请注意，Application Insights 不会保留这些设置。

要编辑单个指标的告警，请按以下步骤操作：

1. 导航到 [CloudWatch 控制台](#)。
2. 在左侧导航窗格中，依次选择 Alarms ( 告警 ) 和 All alarms ( 所有告警 )。
3. 选择 CloudWatch Application Insights 自动创建的告警旁边的单选按钮。然后选择 Actions ( 操作 )，并选择下拉菜单中的 Edit ( 编辑 )。
4. 编辑 Metric ( 指标 ) 下的以下参数。
  - a. 在 Statistic ( 统计数据 ) 下，选择其中一个统计数据或预定义百分比值，或指定一个自定义百分比值。例如：p95.45。
  - b. 在 Period ( 时间段 ) 下，选择告警的评估期。评估告警时，每个时间段都聚合到一个数据点。
5. 编辑 Conditions ( 条件 ) 下的以下参数。
  - a. 选择指标是否必须大于、小于或等于阈值。
  - b. 指定阈值。
6. 在 Additional configuration ( 其他配置 ) 下，编辑以下参数：
  - a. 在 Datapoints to alarm ( 触发告警的数据点数 ) 下，指定必须处于 ALARM ( 告警 ) 状态以启动告警的数据点数或评估期。当两个值匹配时，如果超过指定的连续评估期数，将创建一个告警并进入 ALARM ( 告警 ) 状态。要创建“m ( 最大为 n )”告警，则为第一个数据点指定的值应小于为第二个数据点指定的值。有关评估告警的更多信息，请参阅 [Evaluating an alarm](#) ( 评估告警 )。
  - b. 在 Missing data treatment ( 缺失数据处理 ) 下，选择在缺失某些数据点时的告警行为。有关缺失数据处理的更多信息，请参阅 [配置 CloudWatch 告警处理缺失数据的方式](#)。
  - c. 如果警报将百分比值作为监控的统计数据，将显示样本数少的百分比框。选择是评估还是忽略采样率较低的案例。如果选择忽略 ( 保持警报状态 )，在样本大小太小时，将始终保持当前警报状态。有关采样率较低的百分位数的更多信息，请参阅 [基于百分位数的 CloudWatch 告警和小数据样本](#) (p. 156)。
7. 选择 Next ( 下一步 )。
8. 在通知下面，选择一个在警报处于 ALARM、OK 或 INSUFFICIENT\_DATA 状态时通知的 SNS 主题。
9. 选择 Update alarm ( 更新告警 )。

## 查看和排查 CloudWatch Application Insights 检测到的问题

以下各节提供了一些步骤，可帮助您解决在 Application Insights 上为 SAP HANA 配置监控时出现的常见问题排查情况。

故障排除主题

- [SAP HANA 数据库达到内存分配限制](#) (p. 586)

- [磁盘已满事件 \(p. 586\)](#)
- [SAP HANA 备份停止运行 \(p. 587\)](#)

SAP HANA 数据库达到内存分配限制

描述

由于内存压力过大，由 SAP HANA 数据库提供支持的 SAP 应用程序发生故障，导致应用程序性能下降。

解析

您可以检查动态创建的控制面板（显示相关指标和日志文件片段），找出导致问题的应用程序层。在以下示例中，问题可能是 SAP HANA 系统中数据负载过大。

CloudWatch: Application Insights  
Problem id: p-9197469c-e31b-4f35-8577-0ca0fabff84 [Edit configuration](#)

1h 3h 12h 1d 3d 1w custom (4d) Actions

Problem summary

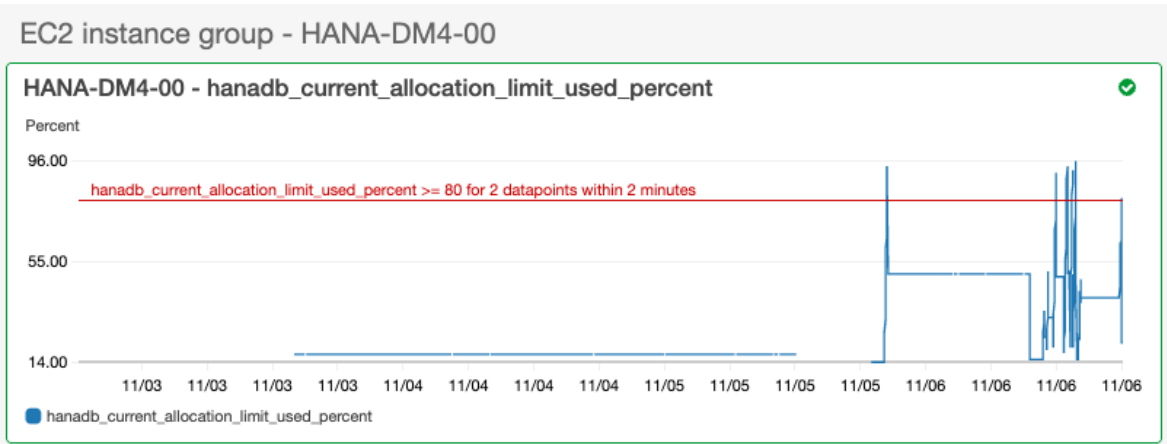
| Severity | Problem summary                                            | Source                                                       | Start-time           | Status      | Resource group     | SSM OpsItem     |
|----------|------------------------------------------------------------|--------------------------------------------------------------|----------------------|-------------|--------------------|-----------------|
| High     | SAP HANA: Allocation limit used (%) exceeded the threshold | saphanacomponent-DM4-00-79ec8266-5692-49c3-8dd8-38163d420087 | 2021-11-03T14:01:21Z | In progress | AI-SUSE-1-Node-DM4 | oi-902e0d35c005 |

Insight 0

Check the current memory utilization. Identify and resolve reasons which are responsible for the used memory coming close to the allocation limit. In addition, examine the CloudWatch Log Insights widget in the problem dashboard below. If your investigation indicates a requirement to have more memory capacity, you can resize your instances to a different EC2 instance type. See <https://aws.amazon.com/sap/instance-types/> for all the SAP certified EC2 instances for SAP HANA.

Help us improve our models: ☐ This insight is useful ☐ This insight is not useful [Submit feedback](#)

已用内存分配超过总内存分配限制的 80% 的阈值。



日志组显示方案 BNR-DATA 和表 IMDBMASTER\_30003 内存不足。此外，日志组会显示问题发生的确切时间、当前全局位置限制、共享内存、代码大小和 OOM 预留分配大小。

Log Group: SAP\_HANA\_TRACE-AI-SUSE-1-Node-DM4, Log Type: SAP\_HANA\_TRACE, AWS:SAPHANA.OutOfMemory

| #  | @timestamp               | @message                                                                                                                                                                                                                                 |
|----|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1  | 2021-11-06T13:31:23.317Z | GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (C9901081728b), SHARED_MEMORY = 567.77mb (C95357696b), CODE_SIZE = 2.94gb (C162598272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (188818752b)                                                   |
| 2  | 2021-11-06T13:31:23.316Z | [2867][311268][22/963854] 2021-11-06 13:08:44.99578 * OOMNotification.Statement.cc(84580) : oom exception occurred at 'indmaster:30003': conn_id=311268, stmt_id=136853818011966, stmt_hash=7elccc2b5f46864c0ea8c9869f0d1, sql=CALL...   |
| 3  | 2021-11-06T13:31:23.316Z | [3803][311513][22/967162] 2021-11-06 13:31:17.163648 * Memory.mmlReportMemoryProblems.cpp(8185) : OUT OF MEMORY occurred.                                                                                                                |
| 4  | 2021-11-06T13:31:23.316Z | Current callstack: 1: 0x00007f8245386d35 in MemoryManager::PoolAllocator::notifyOOMImpl(unsigned long, unsigned long, bool, ltt::allocation_failure_type, bool)<#b1 at mPoolAllocator.cpp:2284 (libhdbasis.so) 2: 0x00007f824524a7ad ... |
| 5  | 2021-11-06T13:31:23.316Z | [2822][1][1-1-1] 2021-11-06 13:31:17.175597 * Memory.mmlReportMemoryProblems.cpp(8185) : OUT OF MEMORY occurred.                                                                                                                         |
| 6  | 2021-11-06T13:31:23.316Z | Current callstack: 1: 0x00007f8245386d35 in MemoryManager::PoolAllocator::notifyOOMImpl(unsigned long, unsigned long, bool, ltt::allocation_failure_type, bool)<#b1 at mPoolAllocator.cpp:2284 (libhdbasis.so) 2: 0x00007f824524a7ad ... |
| 7  | 2021-11-06T13:31:23.316Z | GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (C9901081728b), SHARED_MEMORY = 567.77mb (C95357696b), CODE_SIZE = 2.94gb (C162598272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (188818752b)                                                   |
| 8  | 2021-11-06T13:31:17.316Z | GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (C9901081728b), SHARED_MEMORY = 567.77mb (C95357696b), CODE_SIZE = 2.94gb (C162598272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (188818752b)                                                   |
| 9  | 2021-11-06T13:31:17.317Z | GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (C9901081728b), SHARED_MEMORY = 567.77mb (C95357696b), CODE_SIZE = 2.94gb (C162598272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (188818752b)                                                   |
| 10 | 2021-11-06T13:31:17.317Z | [3803][311513][22/967162] 2021-11-06 13:31:17.180223 * Memory.mmlReportMemoryProblems.cpp(8185) : OUT OF MEMORY occurred.                                                                                                                |
| 11 | 2021-11-06T13:31:17.317Z | GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (C9901081728b), SHARED_MEMORY = 567.77mb (C95357696b), CODE_SIZE = 2.94gb (C162598272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (188818752b)                                                   |
| 12 | 2021-11-06T13:31:17.317Z | [3803][311513][22/967162] 2021-11-06 13:31:17.163648 * Memory.mmlReportMemoryProblems.cpp(8185) : OUT OF MEMORY occurred.                                                                                                                |
| 13 | 2021-11-06T13:31:17.317Z | Current callstack: 1: 0x00007f8245386d35 in MemoryManager::PoolAllocator::notifyOOMImpl(unsigned long, unsigned long, bool, ltt::allocation_failure_type, bool)<#b1 at mPoolAllocator.cpp:2284 (libhdbasis.so) 2: 0x00007f824524a7ad ... |
| 14 | 2021-11-06T13:31:17.317Z | [2822][1][1-1-1] 2021-11-06 13:31:17.175597 * Memory.mmlReportMemoryProblems.cpp(8185) : OUT OF MEMORY occurred.                                                                                                                         |
| 15 | 2021-11-06T13:31:17.317Z | [2822][1][1-1-1] 2021-11-06 13:31:17.175597 * Memory.mmlReportMemoryProblems.cpp(8185) : OUT OF MEMORY occurred.                                                                                                                         |
| 16 | 2021-11-06T13:31:17.317Z | Current callstack: 1: 0x00007f8245386d35 in MemoryManager::PoolAllocator::notifyOOMImpl(unsigned long, unsigned long, bool, ltt::allocation_failure_type, bool)<#b1 at mPoolAllocator.cpp:2284 (libhdbasis.so) 2: 0x00007f824524a7ad ... |
| 17 | 2021-11-06T13:31:17.317Z | GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (C9901081728b), SHARED_MEMORY = 567.77mb (C95357696b), CODE_SIZE = 2.94gb (C162598272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (188818752b)                                                   |
| 18 | 2021-11-06T13:31:16.317Z | GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (C9901081728b), SHARED_MEMORY = 567.77mb (C95357696b), CODE_SIZE = 2.94gb (C162598272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (188818752b)                                                   |
| 19 | 2021-11-06T13:31:16.317Z | GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (C9901081728b), SHARED_MEMORY = 567.77mb (C95357696b), CODE_SIZE = 2.94gb (C162598272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (188818752b)                                                   |

磁盘已满事件

描述



由 SAP HANA 数据库提供支持的 SAP 应用程序停止响应，导致无法访问数据库。

解析

您可以检查动态创建的控制面板（显示相关指标和日志文件片段），找出导致问题的数据库层。在以下示例中，问题可能是管理员无法启用自动日志备份，导致 sap/hana/log 目录填满。

Problem summary

| Severity | Problem summary                             | Source              | Start-time           | Status      | Resource group     | SSM OpsItem    |
|----------|---------------------------------------------|---------------------|----------------------|-------------|--------------------|----------------|
| Medium   | SAP HANA: DISK FULL error has been detected | i-043851dc0a2ab15cc | 2021-11-05T18:07:29Z | In progress | AI-SUSE-1-Node-DM2 | oi-88f4cb8fcf6 |

Insight o

If the HANA database does not accept any of the new requests due to log volume is full. We strongly advise against remove either data files or log files using operating system tools as this will corrupt the database. The recommendation is to follow SAP Note 1679938 to temporarily free up space in the log volume, this way you should be able to start up the database for root cause analysis and problem resolution.

Help us improve our models: ☐ This insight is useful ☐ This insight is not useful

问题控制面板中的日志组小组件显示 DISKFULL 事件。

Log Group: SAP\_HANA\_TRACE-AI-SUSE-1-Node-DM2, Log Type: SAP\_HANA\_TRACE, AWS::SAPHANA.DiskFull

| # | @timestamp               | @message                                                                                                                                |
|---|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 1 | 2021-11-06T18:00:20.072Z | [26768][-1][-1/-1] 2021-11-06 18:00:16.556583 i EventHandler LocalFileCallback.cpp(00517) : [DISKFULL] restarting queue with 1 requests |
|   | @ingestionTime           | 1636221622489                                                                                                                           |
|   | @log                     | [REDACTED]:SAP_HANA_TRACE-AI-SUSE-1-Node-DM2                                                                                            |
|   | @logStream               | i-[REDACTED]                                                                                                                            |
|   | @message                 | [26768][-1][-1/-1] 2021-11-06 18:00:16.556583 i EventHandler LocalFileCallback.cpp(00517) : [DISKFULL] restarting queue with 1 requests |
|   | @timestamp               | 1636221620072                                                                                                                           |

SAP HANA 备份停止运行

描述

由 SAP HANA 数据库提供支持的 SAP 应用程序已停止工作。

解析

您可以检查动态创建的控制面板（显示相关指标和日志文件片段），找出导致问题的数据库层。

问题控制面板中的日志组小组件显示 ACCESS DENIED 事件。其中包括其他信息，如 S3 存储桶、S3 存储桶文件夹和 S3 存储桶区域。

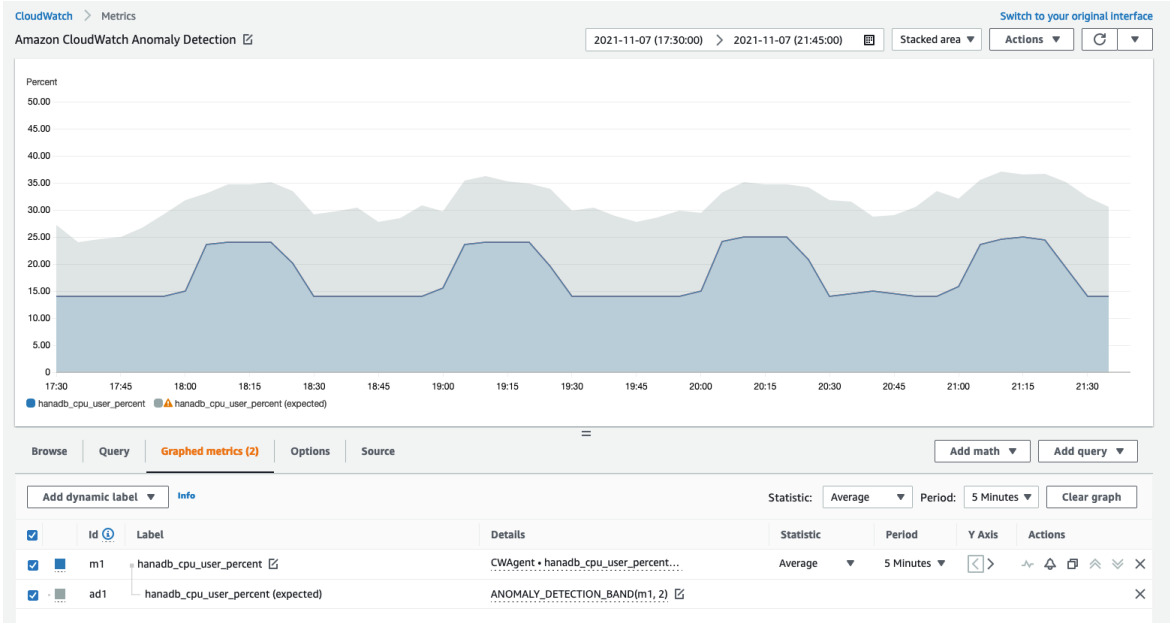
Log Group: SAP\_HANA\_LOGS-AI-SUSE-1-Node-DM3, Log Type: SAP\_HANA\_LOGS, AWS::SAPHANA.BackupErrorAccessDenied

| # | @timestamp               | @message                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 | 2021-11-06T28:34.582Z    | 2021-11-06 20:28:34.493 backup terminated: pid: 21196 exit code: 1 output: exception: exception 1: no.118587 (Backup/Destination/BackupImpl/BackupDestBackup_Executor.cpp:243) Backup exited with exit code 1 instead of 0. console ...                                                                                                                                                                                      |
|   | @ingestionTime           | 7849381381500: SAP_HANA_LOGS-AI-SUSE-1-Node-DM3                                                                                                                                                                                                                                                                                                                                                                              |
|   | @log                     | i-00164a0de25f3231b                                                                                                                                                                                                                                                                                                                                                                                                          |
|   | @logStream               |                                                                                                                                                                                                                                                                                                                                                                                                                              |
|   | @message                 | 2021-11-06 20:28:34.493 backup terminated: pid: 21196 exit code: 1 output: exception: exception 1: no.118587 (Backup/Destination/BackupImpl/BackupDestBackup_Executor.cpp:243) Backup exited with exit code 1 instead of 0. console output: time="2021-11-06T20:28:34Z" level=info msg="Starting execution." time="2021-11-06T20:28:34Z" level=info msg="Loading configuration file /usr/sap/DM3/SYS/global/hdb/opt/hdbconfi |
|   | @timestamp               | 1636220514980                                                                                                                                                                                                                                                                                                                                                                                                                |
| 2 | 2021-11-06T28:27.46.035Z | 2021-11-06 20:27:41.418 backup terminated: pid: 21808 exit code: 1 output: exception: exception 1: no.118587 (Backup/Destination/BackupImpl/BackupDestBackup_Executor.cpp:243) Backup exited with exit code 1 instead of 0. console ...                                                                                                                                                                                      |
| 3 | 2021-11-06T28:27.22.974Z | 2021-11-06 20:27:22.959 backup terminated: pid: 21809 exit code: 1 output: exception: exception 1: no.118587 (Backup/Destination/BackupImpl/BackupDestBackup_Executor.cpp:243) Backup exited with exit code 1 instead of 0. console ...                                                                                                                                                                                      |
| 4 | 2021-11-06T28:26.46.035Z | 2021-11-06 20:26:41.277 backup terminated: pid: 20947 exit code: 1 output: exception: exception 1: no.118587 (Backup/Destination/BackupImpl/BackupDestBackup_Executor.cpp:243) Backup exited with exit code 1 instead of 0. console ...                                                                                                                                                                                      |
| 5 | 2021-11-06T28:26.39.035Z | 2021-11-06 20:26:34.218 backup terminated: pid: 20951 exit code: 1 output: exception: exception 1: no.118587 (Backup/Destination/BackupImpl/BackupDestBackup_Executor.cpp:243) Backup exited with exit code 1 instead of 0. console ...                                                                                                                                                                                      |
| 6 | 2021-11-06T28:26.22.949Z | 2021-11-06 20:26:22.823 backup terminated: pid: 20876 exit code: 1 output: exception: exception 1: no.118587 (Backup/Destination/BackupImpl/BackupDestBackup_Executor.cpp:243) Backup exited with exit code 1 instead of 0. console ...                                                                                                                                                                                      |
| 7 | 2021-11-06T28:25.41.183Z | 2021-11-06 20:25:41.136 backup terminated: pid: 20814 exit code: 1 output: exception: exception 1: no.118587 (Backup/Destination/BackupImpl/BackupDestBackup_Executor.cpp:243) Backup exited with exit code 1 instead of 0. console ...                                                                                                                                                                                      |

SAP HANA 的异常检测

对于特定的 SAP HANA 指标（例如线程数量），CloudWatch 应用统计和机器学习算法来定义阈值。这些算法只需最少的用户干预，即可持续分析 SAP HANA 数据库的指标，确定正常基线和表面异常。算法会生成一个异常检测模型，该模型生成表示正常指标行为的预期值范围。

异常检测算法将指标的季节性变化和趋势变化考虑在内。季节性变化可以是每小时、每天或每周，如下 SAP HANA CPU 使用率示例所示。



创建模型后，CloudWatch 异常检测会持续评估模型并对其进行调整，以确保模型尽可能准确。这包括重新训练模型，以便模型在指标值随着时间的推移而变化或突然发生变化时作出调整。此外还包括用于改进季节性、峰值或稀疏指标模型的预测器。

## 排查 Application Insights for SAP HANA 的问题

本节提供了帮助您解决 Application Insights 控制面板返回的常见错误的步骤。

无法添加超过 60 个监控指标

返回错误：Component cannot have more than 60 monitored metrics.

根本原因：The current metric limit is 60 monitor metrics per component.

解决方法：删除遵守限制所不需要的指标。

载入流程之后未显示 SAP 指标或告警。

返回错误：在 Amazon Systems Manager Run 命令中，AWS-ConfigureAWSPackage 失败。

输出内容显示以下错误：

```
Unable to find a host with system database, for more info rerun using -v
```



### Step 2 - Command description and status

|                               |                                             |                                              |
|-------------------------------|---------------------------------------------|----------------------------------------------|
| Status<br>Failed              | Detailed status<br>Failed                   | Response code<br>1                           |
| Step name<br>configurePackage | Start time<br>Tue, 09 Nov 2021 10:00:47 GMT | Finish time<br>Tue, 09 Nov 2021 10:00:59 GMT |

► Output

▼ Error

The command output displays a maximum of 48,000 characters. You can view the complete command output in either Amazon S3 or CloudWatch Logs, if you specify an S3 bucket or a logs group when you run the command.

```
password=db_creds["password"]
File "./install_utils.py", line 102, in get_master_host
raise InstallationError("Unable to find a host with system database, for more info rerun using -v")
__main__.InstallationError: Unable to find a host with system database, for more info rerun using -v
failed to run commands: exit status 1
Failed to install package; install status Failed
```

CopyDownload

解决方法：用户名和密码可能不正确。验证用户名和密码是否有效，然后重新运行载入过程。

输出内容显示以下安装错误：

```
ERROR: Can not execute `setup.py` since setuptools is not available in the build
environment.
```

或

```
[SSL: CERTIFICATE_VERIFY_FAILED]
```

解决方法：使用以下一条示例 SUSE Linux 命令来安装 Python：

示例 1

```
sudo zypper install -y python36
```

示例 2

安装最新的 Python 3.8 版本。

```
wget https://www.python.org/ftp/python/3.8.<LATEST_RELEASE>/Python-3.8.<LATEST_RELEASE>.tgz
tar xf Python-3.*
cd Python-3.*/
sudo zypper install make gcc-c++ gcc automake autoconf libtool
sudo zypper install zlib-devel
sudo zypper install libopenssl-devel libffi-devel
./configure --with-ensurepip=install
sudo make
sudo make install
sudo su
python3.8 -m pip install --upgrade pip setuptools wheel
```

## 教程：使用 SQL Server 为 .NET 应用程序设置监控

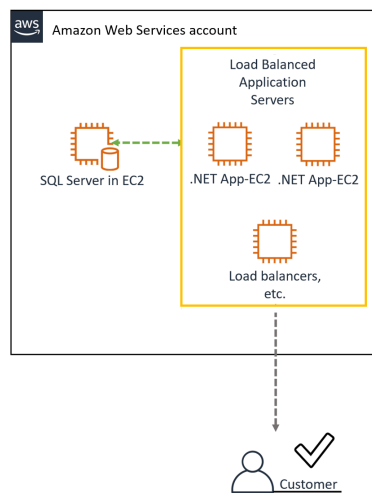
本教程演示如何配置 CloudWatch Application Insights 以监控示例解决方案，然后模拟问题场景以测试解决方案。在此示例中，部署了在后端使用 SQL Server 的负载均衡 Web 应用程序。Web 应用程序和 SQL Server 托管在单独的 EC2 实例上。

#### 教程组件

- [使用案例场景 \(p. 590\)](#)
- [先决条件 \(p. 590\)](#)
- [为示例场景部署资源 \(p. 590\)](#)
- [使用 Amazon CloudWatch Application Insights 设置监控 \(p. 591\)](#)
- [模拟问题场景并查看洞察 \(p. 592\)](#)

## 使用案例场景

在这种场景下，在后端使用 SQL Server 的 .NET 应用程序在 Amazon EC2 实例上运行。部署配置了两个负载均衡的 EC2 实例，托管 Barley Adventure Works 应用程序。这两个实例都访问托管在单独的 EC2 实例上的 SQL Server。监控通过 CloudWatch Application Insights 设置，可快速识别、隔离和解决应用程序问题。



## 先决条件

要完成本教程中的步骤，您必须使用 Amazon 账户。

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

#### 注册 Amazon Web Services 账户

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

在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

## 为示例场景部署资源

提供了一个 CloudFormation 模板，用于自动执行测试的部署场景。该模板部署以下实例：

- 托管 Microsoft SQL Server 数据库的 EC2 实例。
- 两个负载均衡的 EC2 实例。每个负载均衡实例托管 Barley Adventure Works Web 应用程序。

模板部署大约需要 10 分钟完成。

## 部署 CloudFormation 堆栈的步骤

1. 访问 <https://console.aws.amazon.com/cloudformation>，在 Amazon CloudFormation 登录页面上依次选择 Create Stack (创建堆栈) > With new resources (standard) [使用新资源 (标准)]，在位于美国东部 (弗吉尼亚北部) 的账户中启动 CloudFormation 堆栈。此模板仅在 us-east-1 区域中可用。
2. 选择 Create Stack (创建堆栈) 页面上的 Template is ready (模板准备就绪)。
3. 在 Specify template (指定模板) 下，选择 Amazon S3 URL 并输入以下 S3 URL 路径：<https://application-insights-demo-resources.s3.amazonaws.com.cn/SampleApp.yml>。选择 Next (下一步)。

### Note

此 CloudFormation 模板也可以在 aws-samples GitHub 存储库中找到，位置如下：<https://github.com/aws-samples/application-insights-sample-application/blob/master/SampleApp.yml>。

4. 在 Specify Stack details (指定堆栈详细信息) 页面上，输入堆栈名称，如 ApplicationInsightsTest。
5. 审核 Parameters (参数) 下的默认参数并根据自己的偏好修改值。输入 SQLServer 的密码。验证是否遵循 SQL Server 密码复杂性要求，否则应用程序将无法按预期运行。输入现有 EC2 密钥对或在 EC2 控制台中创建新的 EC2 密钥对。选择 Next (下一步)。
6. 在 Configure stack options (配置堆栈选项) 页面的 Tags (标签) 下，您可以选择添加标签以帮助您标识堆栈。选择 Next (下一步)。
7. 在 Review (审核) 页面上审核并确认设置。选中确认模板将创建 Amazon Identity and Access Management (IAM) 资源的复选框。
8. 选择 Create stack (创建堆栈) 以部署堆栈。提醒您只能从 us-east-1 区域部署此模板。
9. 在 CloudFormation 堆栈页面的 Events (事件) 选项卡中监控堆栈部署的状态。当堆栈成功部署后，请继续到下一部分。

## 使用 Amazon CloudWatch Application Insights 设置监控

本部分演示如何从 CloudFormation 模板部署的资源中创建资源组，以及如何将资源组添加到 CloudWatch Application Insights 以进行监控。

### 创建资源组

1. 导航到 [Amazon Resource Groups 控制台](#)，然后选择 Create resource group (创建资源组)。
2. 在 Create query-based group (创建基于查询的组) 页面的 Group type (组类型) 下，选择 CloudFormation stack based (基于 CloudFormation 堆栈)。
3. 在 Group criteria (分组标准) 下，从下拉列表中选择您在上一部分中创建的 CloudFormation 堆栈 (ApplicationInsightsTest)。保留资源类型为 All supported resource types (所有支持的资源类型)。
4. 在 Group details (组详细信息) 下，输入 Group name (组名称) (如 application-insights-resource-group) 以及资源组的描述 (可选)。然后选择 Create group (创建组)。

### 在 CloudWatch Application Insights 上设置资源组监控

1. 导航到 [Amazon CloudWatch 控制台](#)，然后选择左侧导航窗格中 Insights 下的 Application Insights。
2. 选择 View application (查看应用程序) 旁边的 Application Insights。
3. 在 Applications monitored (已监控应用程序) 页面上，选择 Add an application (添加应用程序)。
4. 在 Resource Group select (资源组选择) 下，选择在上一过程中创建的资源组 (application-insights-resource-group)，然后选择 Add application (添加应用程序)。
5. 在 Overview (概览) 页面上，刷新浏览器以显示资源组中的应用程序组件。

6. 选择 Application Load Balancer group ( Application Load Balancer 组 ) , 然后选择 Manage monitoring ( 管理监控 ) 。
7. 在 Manage monitoring ( 管理监控 ) 页面上, 选择 Enable monitoring ( 启用监控 ) 。
8. 选择保存。
9. 若要启用对 SQL Server 实例的监控, 请在 Application components ( 应用程序组件 ) 部分选择 SQL Server EC2 实例, 然后重复上述步骤, 从 Manage monitoring ( 管理监控 ) 页面启用监控。

为 SQL Server 实例监控以下指标：

- CPU 利用率
- StatusCheckFailed
- Memory % Committed Bytes in Use
- Memory Available Mbytes
- Network Interface Bytes Total/sec
- Paging File % Usage
- Physical Disk % Disk Time
- Processor % Processor Time
- SQLServer:Buffer Manager cache hit ratio
- SQLServer:Buffer Manager life expectancy
- SQLServer:General Statistics Processes blocked
- SQLServer:General Statistics User Connections
- SQLServer:Locks Number of Deadlocks/sec
- SQLServer:SQL Statistics Batch Requests/sec
- System Processor Queue Length

为附加到 SQL Server 实例的卷监控以下指标：

- VolumeReadBytes
  - VolumeWriteBytes
  - VolumeReadOps
  - VolumeWriteOps
  - VolumeTotalReadTime
  - VolumeTotalWriteTime
  - VolumeIdleTime
  - VolumeQueueLength
  - VolumeThroughputPercentage
  - VolumeConsumedReadWriteOps
  - BurstBalance
10. 当同时为负载均衡器和 SQL Server 实例启用监控时, 它们所属的资源组将在资源组 Overview ( 概览 ) 页面上显示 Enabled ( 已启用 ) 状态。

## 模拟问题场景并查看洞察

本节说明如何创建 SQL 登录失败、SQL 内存压力事件和 HTTP 500 错误, 以便您可以在 CloudWatch Application Insights 控制面板上查看错误详细信息。

- [模拟高内存压力 \(p. 593\)](#)
- [模拟 HTTP 500 错误 \(p. 593\)](#)

## 模拟 SQL 登录失败

要模拟 SQL 登录失败并从 CloudWatch 控制面板查看问题，请执行以下步骤。

1. 使用您在创建 CloudFormation 堆栈时选择的密钥对登录到为 SQL Server 实例（M4 实例类型）配置的 EC2 实例。
2. 从 Start（启动）菜单中，启动 SQL Management Studio。
3. 输入用户名和不正确的密码，然后选择 Connect（连接）。此时将显示一条消息，指示登录失败。再重复几次此步骤。
4. 在 CloudWatch Application Insights Problems detected（检测到的问题）页面（[CloudWatch 控制台登陆页面](#)的底部）中，该错误应显示在问题摘要下方，显示为 SQL: Login Failure（SQL：登录失败）。要查看有关问题的更多详细信息，请选择问题链接。

## 模拟高内存压力

要模拟可能导致 Web 服务器和负载均衡器上的应用程序性能下降和超时错误的高内存压力事件，请执行以下步骤。

1. 从 SQL 服务器实例启动 SQL Management Studio，并使用 Windows 管理员账户登录。
2. 右键单击数据库服务器，选择 Properties（属性），然后选择 Memory（内存）。
3. 在 Server memory options（服务器内存选项）下，将 Maximum server memory（最大服务器内存）减小到 256KB。
4. 在新的查询窗口中，运行以下 SQL 查询：

```
SELECT COUNT(*)
FROM [AdventureWorks2016].[Sales].[Customer]
```

此时将显示一条消息，指示没有足够的内存来运行该查询。

5. 在 CloudWatch Application Insights Problems detected（检测到的问题）页面（[CloudWatch 控制台登陆页面](#)的底部）中，该错误应显示在问题摘要下方，显示为 SQL: Memory Pressure（SQL：内存压力）。要查看有关问题的更多详细信息，请选择问题链接。

## 模拟 HTTP 500 错误

针对 Web 应用程序未处理 HTTP 请求发起的 HTTP 请求会导致 HTTP 500 错误。要模拟 HTTP 500 错误，请执行以下步骤。

1. 在 Amazon Web Services Management Console 中，导航到 Amazon CloudFormation 控制台。
2. 选择您之前启动的 AppInsightsTest CloudFormation 堆栈的 Output（输出）选项卡。
3. 在 Web 浏览器中打开显示在 Value（值）下的 URL，以打开 AdventureWorks 应用程序。
4. 使用 sampleapp/SalesOrderDetails/edit/5 在前面提到的 URL 后添加后缀，导航到 Barley Adventure Works Web 应用程序的客户详细信息页面。
5. 多次刷新编译的 URL 请求。您的 URL 应该如下所示：http://<YourURL>.us-east-1.elb.amazonaws.com.cn/sampleapp/SalesOrderDetails/edit/5。

将显示一条错误消息，指示找不到该文件或目录。

6. 在 CloudWatch Application Insights Problems detected ( 检测到的问题 ) 页面 ( [CloudWatch 控制台登陆页面](#) 的底部 ) 中, 该错误应显示在问题摘要下方, 显示为 ALBP: Backend 5XX errors ( ALB : 后端 5XX 错误 )。要查看有关问题的更多详细信息, 请选择问题链接。

## 查看和排查 Amazon CloudWatch Application Insights 检测到的问题

本节中的主题介绍 Application Insights 显示的检测到的问题相关详细信息和见解, 并针对检测到的您账户或配置的问题提供了建议解决方法。

故障排除主题

- [CloudWatch 控制台概览](#) (p. 594)
- [Application Insights 问题摘要页面](#) (p. 594)
- [反馈](#) (p. 595)
- [配置错误](#) (p. 595)

## CloudWatch 控制台概览

[CloudWatch 控制台](#) 概览页面中的 CloudWatch Application Insights 窗格下列出了影响已监控应用程序的问题概览。有关更多信息, 请参阅[开始使用 Amazon CloudWatch Application Insights](#) (p. 518)。

CloudWatch Application Insights 概览窗格显示以下内容 :

- 所检测到的问题的严重性 : 高/中/低
- 问题的简短摘要
- 问题来源
- 问题的开始时间
- 问题的解决状态
- 受影响的资源组

要查看特定问题的详细信息, 请在 Problem Summary ( 问题摘要 ) 下选择问题描述。详细控制面板显示问题信息以及相关的指标异常情况和日志错误片段。您可以选择该信息是否有用, 以提供相关性反馈。

如果检测到未配置的新资源, 问题摘要描述将转到 Edit configuration ( 编辑配置 ) 向导以配置新资源。您可以选择详细控制面板右上角的 View/edit configuration ( 查看/编辑配置 ) 来查看或编辑资源组配置。

要返回到概览, 请选择 Back to overview ( 返回概览 ) , 它位于 CloudWatch Application Insights 详细控制面板标题旁边。

## Application Insights 问题摘要页面

Application Insights 问题摘要页面

CloudWatch Application Insights 在问题摘要页面上提供有关检测到的问题的以下信息 :

- 问题的简短摘要
- 问题的开始时间和日期
- 问题严重性 : 高/中/低
- 检测到的问题的状态 : 正在进行/已解决

- 信息：自动生成有关检测到的问题和可能的根本原因的信息
- 信息反馈：您为有关 CloudWatch Application Insights 生成的信息是否有用提供的反馈
- 相关的观察结果：与各种应用程序组件中的问题相关的指标异常情况和相关日志错误片段的详细视图

## 反馈

### 反馈

您可以指定为检测到的问题自动生成的信息是否有用以提供反馈。将使用您对这些信息的反馈以及应用程序诊断（指标异常情况和日志异常情况）改进将来对类似问题的检测。

## 配置错误

CloudWatch Application Insights 使用您的配置为组件创建监控遥测数据。在 Application Insights 检测到您的账户或配置存在问题时，将在应用程序摘要的 Remarks（备注）字段中提供有关如何解决应用程序配置问题的信息。

下表显示了特定备注的建议解决方案。

| 备注                      | 建议的解决方案                                                                                                                                                                                           | 附加说明                                                                                                                                                                                                             |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 已达到 CloudFormation 的配额。 | Application Insights 为每个应用程序创建一个 CloudFormation 堆栈，以管理所有应用程序组件的 CloudWatch 代理安装和配置。默认情况下，每个 Amazon 账户可以具有 2000 个堆栈。请参阅 <a href="#">Amazon CloudFormation 限制</a> 。要解决该问题，请提高 CloudFormation 堆栈的限制。 | 不适用                                                                                                                                                                                                              |
| 在以下实例上没有 SSM 实例角色。      | 要使 Application Insights 能够在应用程序实例上安装和配置 CloudWatch 代理，必须将 AmazonSSMManagedInstanceCore 和 CloudWatchAgentServerPolicy 策略附加到该实例角色。                                                                  | Application Insights 会调用 SSM <a href="#">DescribeInstanceInformation API</a> 以获取具有 SSM 权限的实例列表。在将该角色附加到实例后，SSM 需要一段时间以将实例包含在 DescribeInstanceInformation 结果中。在 SSM 将实例包含在结果中之前，应用程序仍然存在 NO_SSM_INSTANCE_ROLE 错误。 |
| 新组件可能需要进行配置。            | Application Insights 检测到在应用程序资源组中具有新组件。要解决该问题，请相应地配置新组件。                                                                                                                                          | 不适用                                                                                                                                                                                                              |

## Amazon CloudWatch Application Insights 支持的日志和指标

以下列表显示 Amazon CloudWatch Application Insights 支持的日志和指标。



CloudWatch Application Insights 支持以下日志：

- Microsoft Internet 信息服务 (IIS) 日志
- EC2 上的 SQL Server 的错误日志
- 自定义 .NET 应用程序日志，例如 Log4Net
- Windows 事件日志，包括 Windows 日志（系统、应用程序和安全），以及应用程序和服务日志
- 适用于 Amazon Lambda 的 Amazon CloudWatch Logs
- EC2 上 RDS MySQL、Aurora MySQL 和 MySQL 的错误日志和慢速日志
- 适用于 EC2 上的 PostgreSQL RDS 和 PostgreSQL 的 Postgresql 日志
- 适用于 Amazon Step Functions 的 Amazon CloudWatch Logs
- API Gateway REST API 阶段的执行日志和访问日志（JSON、CSV 和 XML，但不包括 CLF）
- Prometheus JMX Exporter 日志 (EMF)
- Amazon RDS 上的 Oracle 和 Amazon EC2 上的 Oracle 的提示日志和侦听器日志
- 使用 [awslogs 日志驱动程序](#) 从 Amazon ECS 容器路由到 CloudWatch 的容器日志。
- 使用 [FireLens 容器日志路由器](#) 从 Amazon ECS 容器路由到 CloudWatch 的容器日志。
- 使用带有 Container Insights 的 [Fluent Bit](#) 或 [Fluentd 日志处理器](#)，从 Amazon EKS 或在 Amazon EC2 上运行的 Kubernetes 路由到 CloudWatch 的容器日志。
- SAP HANA 跟踪和错误日志
- HA Pacemaker 日志
- 使用 [CloudWatch 代理的 proctstat 插件](#) (p. 711) 的 Windows 进程的进程指标

CloudWatch Application Insights 支持以下应用程序组件的指标：

- [Amazon Elastic Compute Cloud \(EC2\)](#) (p. 597)
  - [CloudWatch 内置指标](#) (p. 597)
  - [CloudWatch 代理指标 \(Windows Server\)](#) (p. 598)
  - [CloudWatch 代理进程指标 \(Windows Server\)](#) (p. 601)
  - [CloudWatch 代理指标 \(Linux Server\)](#) (p. 602)
- [Elastic Block Store \(EBS\)](#) (p. 604)
- [Elastic Load Balancer \(ELB\)](#) (p. 605)
- [Application ELB](#) (p. 605)
- [Amazon EC2 Auto Scaling 组](#) (p. 606)
- [Amazon Simple Queue Server \(SQS\)](#) (p. 607)
- [Amazon Relational Database Service \(RDS\)](#) (p. 607)
  - [RDS 数据库实例](#) (p. 607)
  - [RDS 数据库集群](#) (p. 608)
- [Amazon Lambda 函数](#) (p. 609)
- [Amazon DynamoDB 表](#) (p. 609)
- [Amazon S3 存储桶](#) (p. 610)
- [Amazon Step Functions](#) (p. 610)
  - [Execution-level](#) (p. 611)
  - [活动](#) (p. 611)
  - [Lambda 函数](#) (p. 611)
  - [服务集成](#) (p. 612)
  - [Step Functions API](#) (p. 612)
- [API Gateway REST API 阶段](#) (p. 612)



- [SAP HANA \(p. 612\)](#)
- [HA 集群 \(p. 617\)](#)
- [Java \(p. 617\)](#)
- [Amazon Elastic Container Service \(Amazon ECS\) \(p. 618\)](#)
  - [CloudWatch 内置指标 \(p. 618\)](#)
  - [Container Insights 指标 \(p. 618\)](#)
  - [Container Insights Prometheus 指标 \(p. 619\)](#)
- [Amazon 上的 Kubernetes \(p. 620\)](#)
  - [Container Insights 指标 \(p. 620\)](#)
  - [Container Insights Prometheus 指标 \(p. 621\)](#)
- [Amazon FSx \(p. 622\)](#)
- [具有数据点要求的指标 \(p. 622\)](#)
  - [AWS/ApplicationELB \(p. 623\)](#)
  - [AWS/AutoScaling \(p. 623\)](#)
  - [AWS/EC2 \(p. 623\)](#)
  - [Elastic Block Store \(EBS\) \(p. 624\)](#)
  - [AWS/ELB \(p. 625\)](#)
  - [AWS/RDS \(p. 625\)](#)
  - [AWS/Lambda \(p. 626\)](#)
  - [AWS/SQS \(p. 627\)](#)
  - [AWS/CWAgent \(p. 627\)](#)
  - [AWS/DynamoDB \(p. 628\)](#)
  - [AWS/S3 \(p. 629\)](#)
  - [AWS/States \(p. 629\)](#)
  - [AWS/ApiGateway \(p. 630\)](#)
  - [AWS/SNS \(p. 630\)](#)
- [推荐的指标 \(p. 631\)](#)
- [性能计数器指标 \(p. 649\)](#)

## Amazon Elastic Compute Cloud (EC2)

CloudWatch Application Insights 支持以下指标：

### 指标

- [CloudWatch 内置指标 \(p. 597\)](#)
- [CloudWatch 代理指标 \(Windows Server\) \(p. 598\)](#)
- [CloudWatch 代理进程指标 \(Windows Server\) \(p. 601\)](#)
- [CloudWatch 代理指标 \(Linux Server\) \(p. 602\)](#)

## CloudWatch 内置指标

CPUCreditBalance

CPUCreditUsage

CPUSurplusCreditBalance

CPUSurplusCreditsCharged

CPU 利用率

磁盘读取字节数

磁盘读取操作

磁盘写入字节数

磁盘写入操作

EBSByteBalance%

EBSIOBalance%

EBSReadBytes

EBSReadOps

EBSWriteBytes

EBSWriteOps

NetworkIn

网络输出

NetworkPacketsIn

NetworkPacketsOut

StatusCheckFailed

StatusCheckFailed\_Instance

StatusCheckFailed\_System

## CloudWatch 代理指标 (Windows Server)

.NET CLR Exceptions # of Exceps Thrown

.NET CLR Exceptions # of Exceps Thrown/Sec

.NET CLR Exceptions # of Filters/sec

.NET CLR Exceptions # of Finallys/sec

.NET CLR Exceptions Throw to Catch Depth/sec

.NET CLR Interop # of CCWs

.NET CLR Interop # of Stubs

.NET CLR Interop # of TLB exports/sec

.NET CLR Interop # of TLB imports/sec

.NET CLR Interop # of marshaling

.NET CLR Jit % Time in Jit

.NET CLR Jit Standard Jit Failures

.NET CLR Loading % Time Loading

.NET CLR Loading Rate of Load Failures

.NET CLR LocksAndThreads Contention Rate/sec

.NET CLR LocksAndThreads Queue Length/sec

.NET CLR Memory # Total Committed Bytes

.NET CLR Memory % Time in GC

.NET CLR Networking 4.0.0.0 HttpRequest Average Queue Time

.NET CLR Networking 4.0.0.0 HttpWebRequests Aborted/sec

.NET CLR Networking 4.0.0.0 HttpWebRequests Failed/sec

.NET CLR Networking 4.0.0.0 HttpWebRequests Queued/sec

APP\_POOL\_WAS Total Worker Process Ping Failures

ASP.NET Application Restarts

ASP.NET Applications % Managed Processor Time (estimated)

ASP.NET Applications Errors Total/Sec

ASP.NET Applications Errors Unhandled During Execution/sec

ASP.NET Applications Requests in Application Queue

ASP.NET Applications Requests/Sec

ASP.NET Request Wait Time

ASP.NET Requests Queued

HTTP Service Request Queues CurrentQueueSize

LogicalDisk % Free Space

Memory % Committed Bytes In Use

Memory Available Mbytes

Memory Pages/sec

Network Interface Bytes Total/sec

Paging File % Usage

PhysicalDisk % Disk Time

PhysicalDisk Avg. Disk Queue Length

PhysicalDisk Avg. Disk sec/Read

PhysicalDisk Avg. Disk sec/Write

PhysicalDisk Disk Read Bytes/sec

PhysicalDisk Disk Reads/sec

PhysicalDisk Disk Write Bytes/sec

PhysicalDisk Disk Writes/sec

Processor % Idle Time

Processor % Interrupt Time

Processor % Processor Time

Processor % User Time

SQLServer:Access Methods Forwarded Records/sec

SQLServer:Access Methods Full Scans/sec

SQLServer:Access Methods Page Splits/sec

SQLServer:Buffer Manager Buffer cache hit ratio

SQLServer:Buffer Manager Page life expectancy

SQLServer:General Statistics Processes blocked

SQLServer:General Statistics User Connections

SQLServer:Latches Average Latch Wait Time (ms)

SQLServer:Locks Average Wait Time (ms)

SQLServer:Locks Lock Timeouts/sec

SQLServer:Locks Lock Waits/sec

SQLServer:Locks Number of Deadlocks/sec

SQLServer:Memory Manager Memory Grants Pending

SQLServer:SQL Statistics Batch Requests/sec

SQLServer:SQL Statistics SQL Compilations/sec

SQLServer:SQL Statistics SQL Re-Compilations/sec

System Processor Queue Length

TCPv4 Connections Established

TCPv6 Connections Established

W3SVC\_W3WP File Cache Flushes

W3SVC\_W3WP File Cache Misses

W3SVC\_W3WP Requests/Sec  
W3SVC\_W3WP URI Cache Flushes  
W3SVC\_W3WP URI Cache Misses  
Web Service Bytes Received/Sec  
Web Service Bytes Sent/Sec  
Web Service Connection attempts/sec  
Web Service Current Connections  
Web Service Get Requests/sec  
Web Service Post Requests/sec  
Bytes Received/sec  
Normal Messages Queue Length/sec  
Urgent Message Queue Length/sec  
Reconnect Count  
Unacknowledged Message Queue Length/sec  
Messages Outstanding  
Messages Sent/sec  
Database Update Messages/sec  
Update Messages/sec  
Flushes/sec  
Crypto Checkpoints Saved/sec  
Crypto Checkpoints Restored/sec  
Registry Checkpoints Restored/sec  
Registry Checkpoints Saved/sec  
Cluster API Calls/sec  
Resource API Calls/sec  
Cluster Handles/sec  
Resource Handles/sec

### CloudWatch 代理进程指标 (Windows Server)

进程指标使用 [CloudWatch 代理 procstat 插件 \(p. 711\)](#) 收集。只有运行 Windows 工作负载的 Amazon EC2 实例支持进程指标。

procstat cpu\_time\_system

procstat cpu\_time\_user

procstat cpu\_usage

procstat memory\_rss

procstat memory\_vms

procstat read\_bytes

procstat write\_bytes

.procstat read\_count

procstat write\_count

## CloudWatch 代理指标 (Linux Server)

cpu\_time\_active

cpu\_time\_guest

cpu\_time\_guest\_nice

cpu\_time\_idle

cpu\_time\_iowait

cpu\_time\_irq

cpu\_time\_nice

cpu\_time\_softirq

cpu\_time\_steal

cpu\_time\_system

cpu\_time\_user

cpu\_usage\_active

cpu\_usage\_guest

cpu\_usage\_guest\_nice

cpu\_usage\_idle

cpu\_usage\_iowait

cpu\_usage\_irq

cpu\_usage\_nice

cpu\_usage\_softirq

cpu\_usage\_steal

cpu\_usage\_system

cpu\_usage\_user  
disk\_free  
disk\_inodes\_free  
disk\_inodes\_used  
disk\_used  
disk\_used\_percent  
diskio\_io\_time  
diskio\_iops\_in\_progress  
diskio\_read\_bytes  
diskio\_read\_time  
diskio\_reads  
diskio\_write\_bytes  
diskio\_write\_time  
diskio\_writes  
mem\_active  
mem\_available  
mem\_available\_percent  
mem\_buffered  
mem\_cached  
mem\_free  
mem\_inactive  
mem\_used  
mem\_used\_percent  
net\_bytes\_recv  
net\_bytes\_sent  
net\_drop\_in  
net\_drop\_out  
net\_err\_in  
net\_err\_out  
net\_packets\_recv

net\_packets\_sent  
netstat\_tcp\_close  
netstat\_tcp\_close\_wait  
netstat\_tcp\_closing  
netstat\_tcp\_established  
netstat\_tcp\_fin\_wait1  
netstat\_tcp\_fin\_wait2  
netstat\_tcp\_last\_ack  
netstat\_tcp\_listen  
netstat\_tcp\_none  
netstat\_tcp\_syn\_recv  
netstat\_tcp\_syn\_sent  
netstat\_tcp\_time\_wait  
netstat\_udp\_socket  
processes\_blocked  
processes\_dead  
processes\_idle  
processes\_paging  
processes\_running  
processes\_sleeping  
processes\_stopped  
processes\_total  
processes\_total\_threads  
processes\_wait  
processes\_zombies  
swap\_free  
swap\_used  
swap\_used\_percent

## Elastic Block Store (EBS)

CloudWatch Application Insights 支持以下指标：



VolumeReadBytes

VolumeWriteBytes

VolumeReadOps

VolumeWriteOps

VolumeTotalReadTime

VolumeTotalWriteTime

VolumeIdleTime

VolumeQueueLength

VolumeThroughputPercentage

VolumeConsumedReadWriteOps

BurstBalance

## Elastic Load Balancer (ELB)

CloudWatch Application Insights 支持以下指标：

EstimatedALBActiveConnectionCount

EstimatedALBConsumedLCUs

EstimatedALBNewConnectionCount

EstimatedProcessedBytes

HTTPCode\_Backend\_4XX

HTTPCode\_Backend\_5XX

HealthyHostCount

RequestCount

UnHealthyHostCount

## Application ELB

CloudWatch Application Insights 支持以下指标：

EstimatedALBActiveConnectionCount

EstimatedALBConsumedLCUs

EstimatedALBNewConnectionCount

EstimatedProcessedBytes

HTTPCode\_Backend\_4XX

HTTPCode\_Backend\_5XX

HealthyHostCount

延迟

RequestCount

SurgeQueueLength

UnHealthyHostCount

## Amazon EC2 Auto Scaling 组

CloudWatch Application Insights 支持以下指标：

CPUCreditBalance

CPUCreditUsage

CPUSurplusCreditBalance

CPUSurplusCreditsCharged

CPU 利用率

磁盘读取字节数

磁盘读取操作

磁盘写入字节数

磁盘写入操作

EBSByteBalance%

EBSIOBalance%

EBSReadBytes

EBSReadOps

EBSWriteBytes

EBSWriteOps

NetworkIn

网络输出

NetworkPacketsIn

NetworkPacketsOut

StatusCheckFailed

StatusCheckFailed\_Instance

StatusCheckFailed\_System

## Amazon Simple Queue Server (SQS)

CloudWatch Application Insights 支持以下指标：

ApproximateAgeOfOldestMessage

ApproximateNumberOfMessagesDelayed

ApproximateNumberOfMessagesNotVisible

ApproximateNumberOfMessagesVisible

NumberOfEmptyReceives

NumberOfMessagesDeleted

NumberOfMessagesReceived

NumberOfMessagesSent

## Amazon Relational Database Service (RDS)

CloudWatch Application Insights 支持以下指标：

指标

- [RDS 数据库实例 \(p. 607\)](#)
- [RDS 数据库集群 \(p. 608\)](#)

### RDS 数据库实例

BurstBalance

CPUCreditBalance

CPU 利用率

DatabaseConnections

DiskQueueDepth

FailedSQLServerAgentJobsCount

可用存储空间

FreeableMemory

NetworkReceiveThroughput

NetworkTransmitThroughput

读取 IOPS

ReadLatency

读取吞吐量

写入 IOPS

WriteLatency

写入吞吐量

## RDS 数据库集群

ActiveTransactions

AuroraBinlogReplicaLag

AuroraReplicaLag

BackupRetentionPeriodStorageUsed

BinLogDiskUsage

BlockedTransactions

BufferCacheHitRatio

CPU 利用率

CommitLatency

CommitThroughput

DDLlatency

DDLThroughput

DMLLatency

DMLThroughput

DatabaseConnections

死锁数

DeleteLatency

DeleteThroughput

EngineUptime

FreeLocalStorage

FreeableMemory

InsertLatency

InsertThroughput

LoginFailures

NetworkReceiveThroughput

NetworkThroughput

NetworkTransmitThroughput

## 查询

ResultSetCacheHitRatio

SelectLatency

SelectThroughput

SnapshotStorageUsed

TotalBackupStorageBilled

UpdateLatency

UpdateThroughput

VolumeBytesUsed

VolumeReadIOPs

VolumeWriteIOPs

## Amazon Lambda 函数

CloudWatch Application Insights 支持以下指标：

### 错误

DeadLetterErrors

Duration

### 限制

IteratorAge

ProvisionedConcurrencySpilloverInvocations

## Amazon DynamoDB 表

CloudWatch Application Insights 支持以下指标：

SystemErrors

UserErrors

已使用读取容量单位

已使用写入容量单位

ReadThrottleEvents

WriteThrottleEvents

TimeToLiveDeletedItemCount

ConditionalCheckFailedRequests

TransactionConflict

ReturnedRecordsCount

PendingReplicationCount

ReplicationLatency

## Amazon S3 存储桶

CloudWatch Application Insights 支持以下指标：

ReplicationLatency

BytesPendingReplication

OperationsPendingReplication

4xxErrors

5xxErrors

AllRequests

GetRequests

PutRequests

DeleteRequests

HeadRequests

PostRequests

SelectRequests

ListRequests

SelectScannedBytes

SelectReturnedBytes

FirstByteLatency

TotalRequestLatency

BytesDownloaded

BytesUploaded

## Amazon Step Functions

CloudWatch Application Insights 支持以下指标：

指标

- [Execution-level \(p. 611\)](#)
- [活动 \(p. 611\)](#)

- [Lambda 函数](#) (p. 611)
- [服务集成](#) (p. 612)
- [Step Functions API](#) (p. 612)

## Execution-level

ExecutionTime

ExecutionThrottled

ExecutionsFailed

ExecutionsTimedOut

ExecutionsAborted

ExecutionsSucceeded

ExecutionsStarted

## 活动

ActivityRunTime

ActivityScheduleTime

ActivityTime

ActivitiesFailed

ActivitiesHeartbeatTimedOut

ActivitiesTimedOut

ActivitiesScheduled

ActivitiesSucceeded

ActivitiesStarted

## Lambda 函数

LambdaFunctionRunTime

LambdaFunctionScheduleTime

LambdaFunctionTime

LambdaFunctionsFailed

LambdaFunctionsTimedOut

LambdaFunctionsScheduled

LambdaFunctionsSucceeded

LambdaFunctionsStarted

## 服务集成

ServiceIntegrationRunTime

ServiceIntegrationScheduleTime

ServiceIntegrationTime

ServiceIntegrationsFailed

ServiceIntegrationsTimedOut

ServiceIntegrationsScheduled

ServiceIntegrationsSucceeded

ServiceIntegrationsStarted

## Step Functions API

ThrottledEvents

ProvisionedBucketSize

ProvisionedRefillRate

ConsumedCapacity

## API Gateway REST API 阶段

CloudWatch Application Insights 支持以下指标：

4XXError

5XXError

IntegrationLatency

延迟

DataProcessed

CacheHitCount

CacheMissCount

## SAP HANA

### Note

CloudWatch Application Insights 仅支持单个 SID HANA 环境。如果连接了多个 HANA SID，则只会为第一个检测到的 SID 设置监控。

CloudWatch Application Insights 支持以下指标：

hanadb\_every\_service\_started\_status

hanadb\_daemon\_service\_started\_status



hanadb\_preprocessor\_service\_started\_status

hanadb\_webdispatcher\_service\_started\_status

hanadb\_compileservice\_service\_started\_status

hanadb\_nameserver\_service\_started\_status

hanadb\_server\_startup\_time\_variations\_seconds

hanadb\_level\_5\_alerts\_count

hanadb\_level\_4\_alerts\_count

hanadb\_out\_of\_memory\_events\_count

hanadb\_max\_trigger\_read\_ratio\_percent

hanadb\_max\_trigger\_write\_rate\_ratio\_%

hanadb\_log\_switch\_wait\_ratio\_percent

hanadb\_log\_switch\_race\_ratio\_percent

hanadb\_time\_since\_last\_savepoint\_seconds

hanadb\_disk\_usage\_highlevel\_percent

hanadb\_max\_converter\_page\_number\_count

hanadb\_long\_running\_savepoints\_count

hanadb\_failed\_io\_reads\_count

hanadb\_failed\_io\_writes\_count

hanadb\_disk\_data\_unused\_percent

hanadb\_current\_allocation\_limit\_used\_percent

hanadb\_table\_allocation\_limit\_used\_percent

hanadb\_host\_total\_physical\_memory\_mb

hanadb\_host\_physical\_memory\_used\_mb

hanadb\_host\_physical\_memory\_free\_mb

hanadb\_swap\_memory\_free\_mb

hanadb\_swap\_memory\_used\_mb

hanadb\_host\_allocation\_limit\_mb

hanadb\_host\_total\_memory\_used\_mb

hanadb\_host\_total\_peak\_memory\_used\_mb

hanadb\_host\_total\_allocation\_limit\_mb

hanadb\_host\_code\_size\_mb

hanadb\_host\_shared\_memory\_allocation\_mb

hanadb\_cpu\_usage\_percent

hanadb\_cpu\_user\_percent

hanadb\_cpu\_system\_percent

hanadb\_cpu\_waitio\_percent

hanadb\_cpu\_busy\_percent

hanadb\_cpu\_idle\_percent

hanadb\_long\_delta\_merge\_count

hanadb\_unsuccessful\_delta\_merge\_count

hanadb\_successful\_delta\_merge\_count

hanadb\_row\_store\_allocated\_size\_mb

hanadb\_row\_store\_free\_size\_mb

hanadb\_row\_store\_used\_size\_mb

hanadb\_temporary\_tables\_count

hanadb\_large\_non\_compressed\_tables\_count

hanadb\_total\_non\_compressed\_tables\_count

hanadb\_longest\_running\_job\_seconds

hanadb\_average\_commit\_time\_milliseconds

hanadb\_suspended\_sql\_statements\_count

hanadb\_plan\_cache\_hit\_ratio\_percent

hanadb\_plan\_cache\_lookup\_count

hanadb\_plan\_cache\_hit\_count

hanadb\_plan\_cache\_total\_execution\_microseconds

hanadb\_plan\_cache\_cursor\_duration\_microseconds

hanadb\_plan\_cache\_preparation\_microseconds

hanadb\_plan\_cache\_evicted\_count

hanadb\_plan\_cache\_evicted\_microseconds

hanadb\_plan\_cache\_evicted\_preparation\_count

hanadb\_plan\_cache\_evicted\_execution\_count

hanadb\_plan\_cache\_evicted\_preparation\_microseconds  
hanadb\_plan\_cache\_evicted\_cursor\_duration\_microseconds  
hanadb\_plan\_cache\_evicted\_total\_execution\_microseconds  
hanadb\_plan\_cache\_evicted\_plan\_size\_mb  
hanadb\_plan\_cache\_count  
hanadb\_plan\_cache\_preparation\_count  
hanadb\_plan\_cache\_execution\_count  
hanadb\_network\_collision\_rate  
hanadb\_network\_receive\_rate  
hanadb\_network\_transmit\_rate  
hanadb\_network\_packet\_receive\_rate  
hanadb\_network\_packet\_transmit\_rate  
hanadb\_network\_transmit\_error\_rate  
hanadb\_network\_receive\_error\_rate  
hanadb\_time\_until\_license\_expires\_days  
hanadb\_is\_license\_valid\_status  
hanadb\_local\_running\_connections\_count  
hanadb\_local\_idle\_connections\_count  
hanadb\_remote\_running\_connections\_count  
hanadb\_remote\_idle\_connections\_count  
hanadb\_last\_full\_data\_backup\_age\_days  
hanadb\_last\_data\_backup\_age\_days  
hanadb\_last\_log\_backup\_age\_hours  
hanadb\_failed\_data\_backup\_past\_7\_days\_count  
hanadb\_failed\_log\_backup\_past\_7\_days\_count  
hanadb\_oldest\_backup\_in\_catalog\_age\_days  
hanadb\_backup\_catalog\_size\_mb  
hanadb\_hsr\_replication\_status  
hanadb\_hsr\_log\_shipping\_delay\_seconds  
hanadb\_hsr\_secondary\_failover\_count

hanadb\_hsr\_secondary\_reconnect\_count

hanadb\_hsr\_async\_buffer\_used\_mb

hanadb\_hsr\_secondary\_active\_status

hanadb\_handle\_count

hanadb\_ping\_time\_milliseconds

hanadb\_connection\_count

hanadb\_internal\_connection\_count

hanadb\_external\_connection\_count

hanadb\_idle\_connection\_count

hanadb\_transaction\_count

hanadb\_internal\_transaction\_count

hanadb\_external\_transaction\_count

hanadb\_user\_transaction\_count

hanadb\_blocked\_transaction\_count

hanadb\_statement\_count

hanadb\_active\_commit\_id\_range\_count

hanadb\_mvcc\_version\_count

hanadb\_pending\_session\_count

hanadb\_record\_lock\_count

hanadb\_read\_count

hanadb\_write\_count

hanadb\_merge\_count

hanadb\_unload\_count

hanadb\_active\_thread\_count

hanadb\_waiting\_thread\_count

hanadb\_total\_thread\_count

hanadb\_active\_sql\_executor\_count

hanadb\_waiting\_sql\_executor\_count

hanadb\_total\_sql\_executor\_count

hanadb\_data\_write\_size\_mb

hanadb\_data\_write\_time\_milliseconds  
hanadb\_log\_write\_size\_mb  
hanadb\_log\_write\_time\_milliseconds  
hanadb\_data\_read\_size\_mb  
hanadb\_data\_read\_time\_milliseconds  
hanadb\_log\_read\_size\_mb  
hanadb\_log\_read\_time\_milliseconds  
hanadb\_data\_backup\_write\_size\_mb  
hanadb\_data\_backup\_write\_time\_milliseconds  
hanadb\_log\_backup\_write\_size\_mb  
hanadb\_log\_backup\_write\_time\_milliseconds  
hanadb\_mutex\_colision\_count  
hanadb\_read\_write\_lock\_collision\_count  
hanadb\_admission\_control\_admit\_count  
hanadb\_admission\_control\_reject\_count  
hanadb\_admission\_control\_queue\_size\_mb  
hanadb\_admission\_control\_wait\_time\_milliseconds

## HA 集群

CloudWatch Application Insights 支持以下指标：

ha\_cluster\_pacemaker\_stonith\_enabled  
ha\_cluster\_corosync\_quorate  
hanadb\_webdispatcher\_service\_started\_status  
ha\_cluster\_pacemaker\_nodes  
ha\_cluster\_corosync\_ring\_errors  
ha\_cluster\_pacemaker\_fail\_count

## Java

CloudWatch Application Insights 支持以下指标：

java\_lang\_memory\_heapmemoryusage\_used  
java\_lang\_memory\_heapmemoryusage\_committed  
java\_lang\_operatingsystem\_openfiledescriptorcount

java\_lang\_operatingsystem\_maxfiledescriptorcount  
java\_lang\_operatingsystem\_freephysicalmemorysize  
java\_lang\_operatingsystem\_freeswapspacesize  
java\_lang\_threading\_threadcount  
java\_lang\_threading\_daemonthreadcount  
java\_lang\_classloading\_loadedclasscount  
java\_lang\_garbagecollector\_collectiontime\_copy  
java\_lang\_garbagecollector\_collectiontime\_ps\_scavenge  
java\_lang\_garbagecollector\_collectiontime\_parnew  
java\_lang\_garbagecollector\_collectiontime\_marksweepcompact  
java\_lang\_garbagecollector\_collectiontime\_ps\_marksweep  
java\_lang\_garbagecollector\_collectiontime\_concurrentmarksweep  
java\_lang\_garbagecollector\_collectiontime\_g1\_young\_generation  
java\_lang\_garbagecollector\_collectiontime\_g1\_old\_generation  
java\_lang\_garbagecollector\_collectiontime\_g1\_mixed\_generation  
java\_lang\_operatingsystem\_committedvirtualmemorysize

## Amazon Elastic Container Service (Amazon ECS)

CloudWatch Application Insights 支持以下指标：

指标

- [CloudWatch 内置指标 \(p. 618\)](#)
- [Container Insights 指标 \(p. 618\)](#)
- [Container Insights Prometheus 指标 \(p. 619\)](#)

### CloudWatch 内置指标

CPUReservation

CPU 利用率

MemoryReservation

MemoryUtilization

GPUReservation

### Container Insights 指标

ContainerInstanceCount

CpuUtilized  
CpuReserved  
DeploymentCount  
DesiredTaskCount  
MemoryUtilized  
MemoryReserved  
NetworkRxBytes  
NetworkTxBytes  
PendingTaskCount  
RunningTaskCount  
ServiceCount  
StorageReadBytes  
StorageWriteBytes  
TaskCount  
TaskSetCount  
instance\_cpu\_limit  
instance\_cpu\_reserved\_capacity  
instance\_cpu\_usage\_total  
instance\_cpu\_utilization  
instance\_filesystem\_utilization  
instance\_memory\_limit  
instance\_memory\_reserved\_capacity  
instance\_memory\_utilization  
instance\_memory\_working\_set  
instance\_network\_total\_bytes  
instance\_number\_of\_running\_tasks

## Container Insights Prometheus 指标

Java JMX 指标  
java\_lang\_memory\_heapmemoryusage\_used  
java\_lang\_memory\_heapmemoryusage\_committed

java\_lang\_operatingsystem\_openfiledescriptorcount  
java\_lang\_operatingsystem\_maxfiledescriptorcount  
java\_lang\_operatingsystem\_freephysicalmemorysize  
java\_lang\_operatingsystem\_freeswapspacesize  
java\_lang\_threading\_threadcount  
java\_lang\_classloading\_loadedclasscount  
java\_lang\_threading\_daemonthreadcount  
java\_lang\_garbagecollector\_collectiontime\_copy  
java\_lang\_garbagecollector\_collectiontime\_ps\_scavenge  
java\_lang\_garbagecollector\_collectiontime\_parnew  
java\_lang\_garbagecollector\_collectiontime\_marksweepcompact  
java\_lang\_garbagecollector\_collectiontime\_ps\_marksweep  
java\_lang\_garbagecollector\_collectiontime\_concurrentmarksweep  
java\_lang\_garbagecollector\_collectiontime\_g1\_young\_generation  
java\_lang\_garbagecollector\_collectiontime\_g1\_old\_generation  
java\_lang\_garbagecollector\_collectiontime\_g1\_mixed\_generation  
java\_lang\_operatingsystem\_committedvirtualmemorysize

## Amazon 上的 Kubernetes

CloudWatch Application Insights 支持以下指标：

指标

- [Container Insights 指标 \(p. 620\)](#)
- [Container Insights Prometheus 指标 \(p. 621\)](#)

## Container Insights 指标

cluster\_failed\_node\_count  
cluster\_node\_count  
namespace\_number\_of\_running\_pods  
node\_cpu\_limit  
node\_cpu\_reserved\_capacity  
node\_cpu\_usage\_total  
node\_cpu\_utilization



node\_filesystem\_utilization  
node\_memory\_limit  
node\_memory\_reserved\_capacity  
node\_memory\_utilization  
node\_memory\_working\_set  
node\_network\_total\_bytes  
node\_number\_of\_running\_containers  
node\_number\_of\_running\_pods  
pod\_cpu\_reserved\_capacity  
pod\_cpu\_utilization  
pod\_cpu\_utilization\_over\_pod\_limit  
pod\_memory\_reserved\_capacity  
pod\_memory\_utilization  
pod\_memory\_utilization\_over\_pod\_limit  
pod\_network\_rx\_bytes  
pod\_network\_tx\_bytes  
service\_number\_of\_running\_pods

## Container Insights Prometheus 指标

### Java JMX 指标

java\_lang\_memory\_heapmemoryusage\_used  
java\_lang\_memory\_heapmemoryusage\_committed  
java\_lang\_operatingsystem\_openfiledescriptorcount  
java\_lang\_operatingsystem\_maxfiledescriptorcount  
java\_lang\_operatingsystem\_freephysicalmemorysize  
java\_lang\_operatingsystem\_freeswapspacesize  
java\_lang\_threading\_threadcount  
java\_lang\_classloading\_loadedclasscount  
java\_lang\_threading\_daemonthreadcount  
java\_lang\_garbagecollector\_collectiontime\_copy  
java\_lang\_garbagecollector\_collectiontime\_ps\_scavenge

java\_lang\_garbagecollector\_collectiontime\_parnew  
java\_lang\_garbagecollector\_collectiontime\_marksweepcompact  
java\_lang\_garbagecollector\_collectiontime\_ps\_marksweep  
java\_lang\_garbagecollector\_collectiontime\_concurrentmarksweep  
java\_lang\_garbagecollector\_collectiontime\_g1\_young\_generation  
java\_lang\_garbagecollector\_collectiontime\_g1\_old\_generation  
java\_lang\_garbagecollector\_collectiontime\_g1\_mixed\_generation  
java\_lang\_operatingsystem\_committedvirtualmemorysize

## Amazon FSx

CloudWatch Application Insights 支持以下指标：

DataReadBytes

DataWriteBytes

DataReadOperations

DataWriteOperations

MetadataOperations

FreeStorageCapacity

FreeDataStorageCapacity

LogicalDiskUsage

PhysicalDiskUsage

## 具有数据点要求的指标

对于没有触发警报的明显默认阈值的指标，Application Insights 等到指标具有足够的数据点以预测触发警报的合理阈值。CloudWatch Application Insights 在告警创建之前检查的指标数据点要求如下：

- 指标从过去 15 天到过去 2 天期间至少具有 100 个数据点。
- 指标在最后一天至少具有 100 个数据点。

以下指标遵循这些数据点要求。请注意，CloudWatch 代理指标最多需要一小时才能创建警报。

指标

- [AWS/ApplicationELB \(p. 623\)](#)
- [AWS/AutoScaling \(p. 623\)](#)
- [AWS/EC2 \(p. 623\)](#)
- [Elastic Block Store \(EBS\) \(p. 624\)](#)
- [AWS/ELB \(p. 625\)](#)
- [AWS/RDS \(p. 625\)](#)

- [AWS/Lambda \(p. 626\)](#)
- [AWS/SQS \(p. 627\)](#)
- [AWS/CWAgent \(p. 627\)](#)
- [AWS/DynamoDB \(p. 628\)](#)
- [AWS/S3 \(p. 629\)](#)
- [AWS/States \(p. 629\)](#)
- [AWS/ApiGateway \(p. 630\)](#)
- [AWS/SNS \(p. 630\)](#)

## [AWS/ApplicationELB](#)

ActiveConnectionCount

ConsumedLCUs

HTTPCode\_ELB\_4XX\_Count

HTTPCode\_Target\_2XX\_Count

HTTPCode\_Target\_3XX\_Count

HTTPCode\_Target\_4XX\_Count

HTTPCode\_Target\_5XX\_Count

NewConnectionCount

ProcessedBytes

TargetResponseTime

UnHealthyHostCount

## [AWS/AutoScaling](#)

GroupDesiredCapacity

GroupInServiceInstances

GroupMaxSize

GroupMinSize

GroupPendingInstances

GroupStandbyInstances

GroupTerminatingInstances

GroupTotalInstances

## [AWS/EC2](#)

CPUCreditBalance

CPUCreditUsage

CPUSurplusCreditBalance

CPUSurplusCreditsCharged

CPU 利用率

磁盘读取字节数

磁盘读取操作

磁盘写入字节数

磁盘写入操作

EBSByteBalance%

EBSIOBalance%

EBSReadBytes

EBSReadOps

EBSWriteBytes

EBSWriteOps

NetworkIn

网络输出

NetworkPacketsIn

NetworkPacketsOut

## Elastic Block Store (EBS)

VolumeReadBytes

VolumeWriteBytes

VolumeReadOps

VolumeWriteOps

VolumeTotalReadTime

VolumeTotalWriteTime

VolumIdleTime

VolumeQueueLength

VolumeThroughputPercentage

VolumeConsumedReadWriteOps

BurstBalance

## AWS/ELB

EstimatedALBActiveConnectionCount

EstimatedALBConsumedLCUs

EstimatedALBNewConnectionCount

EstimatedProcessedBytes

HTTPCode\_Backend\_4XX

HTTPCode\_Backend\_5XX

HealthyHostCount

延迟

RequestCount

SurgeQueueLength

UnHealthyHostCount

## AWS/RDS

ActiveTransactions

AuroraBinlogReplicaLag

AuroraReplicaLag

BackupRetentionPeriodStorageUsed

BinLogDiskUsage

BlockedTransactions

CPUCreditBalance

CommitLatency

CommitThroughput

DDLlatency

DDLThroughput

DMLlatency

DMLThroughput

DatabaseConnections

死锁数

DeleteLatency

DeleteThroughput

DiskQueueDepth

EngineUptime

FreeLocalStorage

可用存储空间

FreeableMemory

InsertLatency

InsertThroughput

LoginFailures

NetworkReceiveThroughput

NetworkThroughput

NetworkTransmitThroughput

查询

读取 IOPS

读取吞吐量

SelectLatency

SelectThroughput

SnapshotStorageUsed

TotalBackupStorageBilled

UpdateLatency

UpdateThroughput

VolumeBytesUsed

VolumeReadIOPs

VolumeWriteIOPs

写入 IOPS

写入吞吐量

[AWS/Lambda](#)

错误

DeadLetterErrors

Duration

## 限制

IteratorAge

ProvisionedConcurrencySpilloverInvocations

## AWS/SQS

ApproximateAgeOfOldestMessage

ApproximateNumberOfMessagesDelayed

ApproximateNumberOfMessagesNotVisible

ApproximateNumberOfMessagesVisible

NumberOfEmptyReceives

NumberOfMessagesDeleted

NumberOfMessagesReceived

NumberOfMessagesSent

## AWS/CWAgent

LogicalDisk % Free Space

Memory % Committed Bytes In Use

Memory Available Mbytes

Network Interface Bytes Total/sec

Paging File % Usage

PhysicalDisk % Disk Time

PhysicalDisk Avg. Disk sec/Read

PhysicalDisk Avg. Disk sec/Write

PhysicalDisk Disk Read Bytes/sec

PhysicalDisk Disk Reads/sec

PhysicalDisk Disk Write Bytes/sec

PhysicalDisk Disk Writes/sec

Processor % Idle Time

Processor % Interrupt Time

Processor % Processor Time

Processor % User Time

SQLServer:Access Methods Forwarded Records/sec

SQLServer:Access Methods Page Splits/sec

SQLServer:Buffer Manager Buffer cache hit ratio

SQLServer:Buffer Manager Page life expectancy

SQLServer : 接收的数据库副本文件字节数/秒

SQLServer : 接收的数据库副本日志字节数/秒

SQLServer : 保留数据库副本日志以供撤消

SQLServer : 数据库副本日志发送队列

SQLServer : 数据库副本镜像写事务数/秒

SQLServer : 数据库副本恢复队列

SQLServer : 剩余数据库副本重做字节数

SQLServer : 数据库副本已重做的字节数/秒

SQLServer : 需要撤消的数据库副本总日志

SQLServer : 数据库副本事务延迟

SQLServer:General Statistics Processes blocked

SQLServer:SQL Statistics Batch Requests/sec

SQLServer:SQL Statistics SQL Compilations/sec

SQLServer:SQL Statistics SQL Re-Compilations/sec

System Processor Queue Length

TCPv4 Connections Established

TCPv6 Connections Established

## AWS/DynamoDB

已使用读取容量单位

已使用写入容量单位

ReadThrottleEvents

WriteThrottleEvents

TimeToLiveDeletedItemCount

ConditionalCheckFailedRequests

TransactionConflict

ReturnedRecordsCount

PendingReplicationCount



ReplicationLatency

## AWS/S3

ReplicationLatency

BytesPendingReplication

OperationsPendingReplication

4xxErrors

5xxErrors

AllRequests

GetRequests

PutRequests

DeleteRequests

HeadRequests

PostRequests

SelectRequests

ListRequests

SelectScannedBytes

SelectReturnedBytes

FirstByteLatency

TotalRequestLatency

BytesDownloaded

BytesUploaded

## AWS/States

ActivitiesScheduled

ActivitiesStarted

ActivitiesSucceeded

ActivityScheduleTime

ActivityRuntime

ActivityTime

LambdaFunctionsScheduled

LambdaFunctionsStarted

LambdaFunctionsSucceeded  
LambdaFunctionScheduleTime  
LambdaFunctionRuntime  
LambdaFunctionTime  
ServiceIntegrationsScheduled  
ServiceIntegrationsStarted  
ServiceIntegrationsSucceeded  
ServiceIntegrationScheduleTime  
ServiceIntegrationRuntime  
ServiceIntegrationTime  
ProvisionedRefillRate  
ProvisionedBucketSize  
ConsumedCapacity  
ThrottledEvents

## [AWS/ApiGateway](#)

4XXError  
IntegrationLatency  
延迟

DataProcessed  
CacheHitCount  
CacheMissCount

## [AWS/SNS](#)

NumberOfNotificationsDelivered  
NumberOfMessagesPublished  
NumberOfNotificationsFailed  
NumberOfNotificationsFilteredOut  
NumberOfNotificationsFilteredOut-InvalidAttributes  
NumberOfNotificationsFilteredOut-NoMessageAttributes  
NumberOfNotificationsRedrivenToDlq  
NumberOfNotificationsFailedToRedriveToDlq

SMSSuccessRate

推荐的指标

下表列出了每种组件类型的建议指标。

| 组件类型                    | 工作负载类型           | 建议的指标                                                                                                                                                                                                                                                                                                   |
|-------------------------|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EC2 实例 (Windows Server) | 默认/自定义           | CPU 利用率<br><br>StatusCheckFailed<br><br>Processor % Processor Time<br><br>Memory % Committed Bytes In Use<br><br>LogicalDisk % Free Space<br><br>Memory Available Mbytes                                                                                                                                |
|                         | Active Directory | CPU 利用率<br><br>StatusCheckFailed<br><br>Processor % Processor Time<br><br>Memory % Committed Bytes In Use<br><br>Memory Available Mbytes<br>数据库 ==> 实例数据库缓存 % 命中<br><br>DirectoryServices DRA 挂起的复制操作<br><br>DirectoryServices DRA 挂起的复制同步<br><br>DNS 递归查询失败/秒<br><br>LogicalDisk 平均值 Disk Queue Length |
|                         | Java 应用程序        | CPU 利用率<br><br>StatusCheckFailed<br><br>Processor % Processor Time<br><br>Memory % Committed Bytes In Use<br><br>Memory Available Mbytes<br><br>java_lang_threading_threadcount<br><br>java_lang_classloading_loadedclasscount                                                                          |

| 组件类型 | 工作负载类型                    | 建议的指标                                                                                                                                                                                                                                                                                                                                                              |
|------|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      |                           | java_lang_memory_heapmemoryusage_used<br>java_lang_memory_heapmemoryusage_committ<br>java_lang_operatingsystem_freephysicalmemory<br>java_lang_operatingsystem_freeswapspace                                                                                                                                                                                       |
|      | Microsoft IIS/.NET Web 前端 | CPU 利用率<br>StatusCheckFailed<br>Processor % Processor Time<br>Memory % Committed Bytes In Use<br>Memory Available Mbytes<br>.NET CLR Exceptions # of Exceps Thrown/Sec<br>.NET CLR Memory # Total Committed Bytes<br>.NET CLR Memory % Time in GC<br>ASP.NET Applications Requests in Application Queue<br>ASP.NET Requests Queued<br>ASP.NET Application Restarts |

| 组件类型 | 工作负载类型                    | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | Microsoft SQL Server 数据库层 | CPU 利用率<br><br>StatusCheckFailed<br><br>Processor % Processor Time<br><br>Memory % Committed Bytes In Use<br><br>Memory Available Mbytes<br><br>Paging File % Usage<br><br>System Processor Queue Length<br><br>Network Interface Bytes Total/Sec<br><br>PhysicalDisk % Disk Time<br><br>SQLServer:Buffer Manager Buffer Cache Hit ratio<br><br>SQLServer:Buffer Manager Page Life Expectancy<br><br>SQLServer:General Statistics Processes Blocked<br><br>SQLServer:General Statistics User Connections<br><br>SQLServer:Locks Number of Deadlocks/Sec<br><br>SQLServer:SQL Statistics Batch Requests/Sec |
|      | MySQL                     | CPU 利用率<br><br>StatusCheckFailed<br><br>Processor % Processor Time<br><br>Memory % Committed Bytes In Use<br><br>LogicalDisk % Free Space<br><br>Memory Available Mbytes                                                                                                                                                                                                                                                                                                                                                                                                                                   |

| 组件类型 | 工作负载类型         | 建议的指标                                                                                                                                                                                                                                             |
|------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | .NET 工作线程池/中间层 | CPU 利用率<br>StatusCheckFailed<br>Processor % Processor Time<br>Memory % Committed Bytes In Use<br>Memory Available Mbytes<br>.NET CLR Exceptions # of Exceps Thrown/Sec<br>.NET CLR Memory # Total Committed Bytes<br>.NET CLR Memory % Time in GC |
|      | .NET Core 层    | CPU 利用率<br>StatusCheckFailed<br>Processor % Processor Time<br>Memory % Committed Bytes In Use<br>Memory Available Mbytes                                                                                                                          |
|      | Oracle         | CPU 利用率<br>StatusCheckFailed<br>Processor % Processor Time<br>Memory % Committed Bytes In Use<br>LogicalDisk % Free Space<br>Memory Available Mbytes                                                                                              |
|      | Postgres       | CPU 利用率<br>StatusCheckFailed<br>Processor % Processor Time<br>Memory % Committed Bytes In Use<br>LogicalDisk % Free Space<br>Memory Available Mbytes                                                                                              |

| 组件类型 | 工作负载类型     | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | SharePoint | <p>CPU 利用率</p> <p>StatusCheckFailed</p> <p>Processor % Processor Time</p> <p>Memory % Committed Bytes In Use</p> <p>Memory Available Mbytes</p> <p>ASP.NET 应用程序缓存 API 修剪</p> <p>ASP.NET 请求被拒</p> <p>ASP.NET 工作进程重启</p> <p>Memory Pages/sec</p> <p>SharePoint 发布缓存发布缓存刷新/秒</p> <p>SharePoint Foundation 执行时间/页面请求</p> <p>SharePoint 基于磁盘的缓存缓存压缩总数</p> <p>SharePoint 基于磁盘的缓存 Blob 缓存命中率</p> <p>SharePoint 基于磁盘的缓存 Blob 缓存填充率</p> <p>SharePoint 基于磁盘的缓存 Blob 缓存刷新/秒</p> <p>ASP.NET Requests Queued</p> <p>ASP.NET Applications Requests in Application Queue</p> <p>ASP.NET Application Restarts</p> <p>LogicalDisk 平均值 Disk sec/Write</p> <p>LogicalDisk 平均值 Disk sec/Read</p> <p>Processor % Interrupt Time</p> |

| 组件类型                  | 工作负载类型                       | 建议的指标                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EC2 实例 (Linux Server) | 默认/自定义                       | CPU 利用率<br><br>StatusCheckFailed<br><br>disk_used_percent<br><br>mem_used_percent                                                                                                                                                                                                                                                                               |
|                       | Java 应用程序                    | CPU 利用率<br><br>StatusCheckFailed<br><br>disk_used_percent<br><br>mem_used_percent<br><br>java_lang_threading_threadcount<br><br>java_lang_classloading_loadedclasscount<br><br>java_lang_memory_heapmemoryusage_used<br>java_lang_memory_heapmemoryusage_committ<br><br>java_lang_operatingsystem_freephysicalmemory<br>java_lang_operatingsystem_freeswapspace |
|                       | .NET Core 层或 SQL Server 数据库层 | CPU 利用率<br><br>StatusCheckFailed<br><br>disk_used_percent<br><br>mem_used_percent                                                                                                                                                                                                                                                                               |
|                       | Oracle                       | CPU 利用率<br><br>StatusCheckFailed<br><br>disk_used_percent<br><br>mem_used_percent                                                                                                                                                                                                                                                                               |
|                       | Postgres                     | CPU 利用率<br><br>StatusCheckFailed<br><br>disk_used_percent<br><br>mem_used_percent                                                                                                                                                                                                                                                                               |



| 组件类型            | 工作负载类型           | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EC2 实例组         | SAP HANA 多节点或单节点 | <ul style="list-style-type: none"> <li>hanadb_server_startup_time_variations_seconds</li> <li>hanadb_level_5_alerts_count</li> <li>hanadb_level_4_alerts_count</li> <li>hanadb_out_of_memory_events_count</li> <li>hanadb_max_trigger_read_ratio_percent</li> <li>hanadb_max_trigger_write_rate_ratio_percent</li> <li>hanadb_log_switch_race_ratio_percent</li> <li>hanadb_time_since_last_savepoint_seconds</li> <li>hanadb_disk_usage_highlevel_percent</li> <li>hanadb_current_allocation_limit_used_percent</li> <li>hanadb_table_allocation_limit_used_percent</li> <li>hanadb_cpu_usage_percent</li> <li>hanadb_plan_cache_hit_ratio_percent</li> <li>hanadb_last_data_backup_age_days</li> </ul> |
| EBS 卷           | 任何               | VolumeReadBytes<br>VolumeWriteBytes<br>VolumeReadOps<br>VolumeWriteOps<br>VolumeQueueLength<br>VolumeThroughputPercentage<br>VolumeConsumedReadWriteOps<br>BurstBalance                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Classic ELB     | 任何               | HTTPCode_Backend_4XX<br>HTTPCode_Backend_5XX<br>延迟<br>SurgeQueueLength<br>UnHealthyHostCount                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Application ELB | 任何               | HTTPCode_Target_4XX_Count<br>HTTPCode_Target_5XX_Count<br>TargetResponseTime<br>UnHealthyHostCount                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

| 组件类型              | 工作负载类型 | 建议的指标                                                                                                                                                      |
|-------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RDS 数据库实例         | 任何     | CPU 利用率<br>ReadLatency<br>WriteLatency<br>BurstBalance<br>FailedSQLServerAgentJobsCount                                                                    |
| RDS 数据库集群         | 任何     | CPU 利用率<br>CommitLatency<br>DatabaseConnections<br>死锁数<br>FreeableMemory<br>NetworkThroughput<br>VolumeBytesUsed                                           |
| Lambda 函数         | 任何     | Duration<br>错误<br>IteratorAge<br>ProvisionedConcurrencySpilloverInvocations<br>限制                                                                          |
| SQS Queue         | 任何     | ApproximateAgeOfOldestMessage<br>ApproximateNumberOfMessagesVisible<br>NumberOfMessagesSent                                                                |
| Amazon DynamoDB 表 | 任何     | SystemErrors<br>UserErrors<br>已使用读取容量单位<br>已使用写入容量单位<br>ReadThrottleEvents<br>WriteThrottleEvents<br>ConditionalCheckFailedRequests<br>TransactionConflict |

| 组件类型                  | 工作负载类型 | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Amazon S3 存储桶         | 任何     | <p>如果启用了带有复制时间控制 (RTC) 的复制配置：</p> <p>ReplicationLatency</p> <p>BytesPendingReplication</p> <p>OperationsPendingReplication</p> <p>如果启用了请求指标：</p> <p>5xxErrors</p> <p>4xxErrors</p> <p>BytesDownloaded</p> <p>BytesUploaded</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Amazon Step Functions | 任何     | <p>常规</p> <ul style="list-style-type: none"> <li>• ExecutionThrottled</li> <li>• ExecutionsAborted</li> <li>• ProvisionedBucketSize</li> <li>• ProvisionedRefillRate</li> <li>• ConsumedCapacity</li> </ul> <p>如果状态机类型为 <b>EXPRESS</b> 或日志组级别为 <b>OFF</b></p> <ul style="list-style-type: none"> <li>• ExecutionsFailed</li> <li>• ExecutionsTimedOut</li> </ul> <p>如果状态机有 Lambda 函数</p> <ul style="list-style-type: none"> <li>• LambdaFunctionsFailed</li> <li>• LambdaFunctionsTimedOut</li> </ul> <p>如果状态机有活动</p> <ul style="list-style-type: none"> <li>• ActivitiesFailed</li> <li>• ActivitiesTimedOut</li> <li>• ActivitiesHeartbeatTimedOut</li> </ul> <p>如果状态机有服务集成</p> <ul style="list-style-type: none"> <li>• ServiceIntegrationsFailed</li> <li>• ServiceIntegrationsTimedOut</li> </ul> |

| 组件类型                    | 工作负载类型 | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| API Gateway REST API 阶段 | 任何     | <ul style="list-style-type: none"><li>• 4xxErrors</li><li>• 5xxErrors</li><li>• 延迟</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| ECS 集群                  | 任何     | CpuUtilized<br><br>MemoryUtilized<br><br>NetworkRxBytes<br><br>NetworkTxBytes<br><br>RunningTaskCount<br><br>PendingTaskCount<br><br>StorageReadBytes<br><br>StorageWriteBytes<br><br>CPUReservation ( 仅限 EC2 启动类型 )<br><br>CPUUtilization ( 仅限 EC2 启动类型 )<br><br>MemoryReservation ( 仅限 EC2 启动类型 )<br><br>MemoryUtilization ( 仅限 EC2 启动类型 )<br><br>GPUReservation ( 仅限 EC2 启动类型 )<br><br>instance_cpu_utilization ( 仅限 EC2 启动类型 )<br><br>instance_filesystem_utilization ( 仅限 EC2 启动类型 )<br><br>instance_memory_utilization ( 仅限 EC2 启动类型 )<br><br>instance_network_total_bytes ( 仅限 EC2 启动类型 ) |

| 组件类型 | 工作负载类型    | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | Java 应用程序 | CpuUtilized<br>MemoryUtilized<br>NetworkRxBytes<br>NetworkTxBytes<br>RunningTaskCount<br>PendingTaskCount<br>StorageReadBytes<br>StorageWriteBytes<br>CPUReservation ( 仅限 EC2 启动类型 )<br>CPUUtilization ( 仅限 EC2 启动类型 )<br>MemoryReservation ( 仅限 EC2 启动类型 )<br>MemoryUtilization ( 仅限 EC2 启动类型 )<br>GPUReservation ( 仅限 EC2 启动类型 )<br>instance_cpu_utilization ( 仅限 EC2 启动类型 )<br>instance_filesystem_utilization ( 仅限 EC2 启动类型 )<br>instance_memory_utilization ( 仅限 EC2 启动类型 )<br>instance_network_total_bytes ( 仅限 EC2 启动类型 )<br>java_lang_threading_threadcount<br>java_lang_classloading_loadedclasscount<br>java_lang_memory_heapmemoryusage_used<br>java_lang_memory_heapmemoryusage_committ<br>java_lang_operatingsystem_freephysicalmemory<br>java_lang_operatingsystem_freeswapspace |

| 组件类型   | 工作负载类型    | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ECS 服务 | 任何        | CPU 利用率<br>MemoryUtilization<br>CpuUtilized<br>MemoryUtilized<br>NetworkRxBytes<br>NetworkTxBytes<br>RunningTaskCount<br>PendingTaskCount<br>StorageReadBytes<br>StorageWriteBytes                                                                                                                                                                                                                                                               |
|        | Java 应用程序 | CPU 利用率<br>MemoryUtilization<br>CpuUtilized<br>MemoryUtilized<br>NetworkRxBytes<br>NetworkTxBytes<br>RunningTaskCount<br>PendingTaskCount<br>StorageReadBytes<br>StorageWriteBytes<br>java_lang_threading_threadcount<br>java_lang_classloading_loadedclasscount<br>java_lang_memory_heapmemoryusage_used<br>java_lang_memory_heapmemoryusage_committ<br>java_lang_operatingsystem_freephysicalmemory<br>java_lang_operatingsystem_freeswapspace |

| 组件类型   | 工作负载类型 | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EKS 集群 | 任何     | cluster_failed_node_count<br><br>node_cpu_reserved_capacity<br><br>node_cpu_utilization<br><br>node_filesystem_utilization<br><br>node_memory_reserved_capacity<br><br>node_memory_utilization<br><br>node_network_total_bytes<br><br>pod_cpu_reserved_capacity<br><br>pod_cpu_utilization<br><br>pod_cpu_utilization_over_pod_limit<br><br>pod_memory_reserved_capacity<br><br>pod_memory_utilization<br><br>pod_memory_utilization_over_pod_limit<br><br>pod_network_rx_bytes<br><br>pod_network_tx_bytes |

| 组件类型 | 工作负载类型    | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | Java 应用程序 | cluster_failed_node_count<br>node_cpu_reserved_capacity<br>node_cpu_utilization<br>node_filesystem_utilization<br>node_memory_reserved_capacity<br>node_memory_utilization<br>node_network_total_bytes<br>pod_cpu_reserved_capacity<br>pod_cpu_utilization<br>pod_cpu_utilization_over_pod_limit<br>pod_memory_reserved_capacity<br>pod_memory_utilization<br>pod_memory_utilization_over_pod_limit<br>pod_network_rx_bytes<br>pod_network_tx_bytes<br>java_lang_threading_threadcount<br>java_lang_classloading_loadedclasscount<br>java_lang_memory_heapmemoryusage_used<br>java_lang_memory_heapmemoryusage_committ<br>java_lang_operatingsystem_freephysicalmemory<br>java_lang_operatingsystem_freeswapspace |



| 组件类型                 | 工作负载类型 | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EC2 上的 Kubernetes 集群 | 任何     | cluster_failed_node_count<br><br>node_cpu_reserved_capacity<br><br>node_cpu_utilization<br><br>node_filesystem_utilization<br><br>node_memory_reserved_capacity<br><br>node_memory_utilization<br><br>node_network_total_bytes<br><br>pod_cpu_reserved_capacity<br><br>pod_cpu_utilization<br><br>pod_cpu_utilization_over_pod_limit<br><br>pod_memory_reserved_capacity<br><br>pod_memory_utilization<br><br>pod_memory_utilization_over_pod_limit<br><br>pod_network_rx_bytes<br><br>pod_network_tx_bytes |

| 组件类型 | 工作负载类型    | 建议的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | Java 应用程序 | cluster_failed_node_count<br><br>node_cpu_reserved_capacity<br>node_cpu_utilization<br>node_filesystem_utilization<br>node_memory_reserved_capacity<br>node_memory_utilization<br>node_network_total_bytes<br>pod_cpu_reserved_capacity<br>pod_cpu_utilization<br>pod_cpu_utilization_over_pod_limit<br>pod_memory_reserved_capacity<br>pod_memory_utilization<br>pod_memory_utilization_over_pod_limit<br>pod_network_rx_bytes<br>pod_network_tx_bytes<br>java_lang_threading_threadcount<br>java_lang_classloading_loadedclasscount<br>java_lang_memory_heapmemoryusage_used<br>java_lang_memory_heapmemoryusage_committ<br>java_lang_operatingsystem_freephysicalmemory<br>java_lang_operatingsystem_freeswapspace |

下表列出了每种组件类型的建议流程和流程指标。CloudWatch Application Insights 不建议对不在实例上运行的流程进行流程监控。

| 组件类型                    | 工作负载类型                    | 建议的流程 | 建议的指标                                                                                |
|-------------------------|---------------------------|-------|--------------------------------------------------------------------------------------|
| EC2 实例 (Windows Server) | Microsoft IIS/.NET Web 前端 | w3wp  | procstat<br>cpu_usage,<br><br>procstat<br>memory_rss,<br><br>procstat<br>memory_vms, |

| 组件类型 | 工作负载类型                    | 建议的流程                    | 建议的指标                                                                                                                                              |
|------|---------------------------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
|      | Microsoft SQL Server 数据库层 |                          | procstat<br>read_bytes,<br><br>procstat<br>write_bytes                                                                                             |
|      |                           | SQLAgent                 | procstat<br>cpu_usage,<br><br>procstat<br>memory_rss,<br><br>procstat<br>memory_vms,<br><br>procstat<br>read_bytes,<br><br>procstat<br>write_bytes |
|      |                           | sqlservr                 | procstat<br>cpu_usage,<br><br>procstat<br>memory_rss,<br><br>procstat<br>memory_vms,<br><br>procstat<br>read_bytes,<br><br>procstat<br>write_bytes |
|      |                           | sqlwriter                | procstat<br>cpu_usage,<br><br>procstat<br>memory_rss                                                                                               |
|      |                           | ReportingServicesService | procstat<br>cpu_usage,<br><br>procstat<br>memory_rss                                                                                               |

| 组件类型 | 工作负载类型         | 建议的流程      | 建议的指标                                                                                                                                              |
|------|----------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
|      |                | MsDtsServr | procstat<br>cpu_usage,<br><br>procstat<br>memory_rss,<br><br>procstat<br>memory_vms,<br><br>procstat<br>read_bytes,<br><br>procstat<br>write_bytes |
|      |                | Msmdsrv    | procstat<br>cpu_usage,<br><br>procstat<br>memory_rss,<br><br>procstat<br>memory_vms,<br><br>procstat<br>read_bytes,<br><br>procstat<br>write_bytes |
|      | .NET 工作线程池/中间层 | w3wp       | procstat<br>cpu_usage,<br><br>procstat<br>memory_rss,<br><br>procstat<br>memory_vms,<br><br>procstat<br>read_bytes,<br><br>procstat<br>write_bytes |

| 组件类型 | 工作负载类型      | 建议的流程 | 建议的指标                                                                                                                                              |
|------|-------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------|
|      | .NET Core 层 | w3wp  | procstat<br>cpu_usage,<br><br>procstat<br>memory_rss,<br><br>procstat<br>memory_vms,<br><br>procstat<br>read_bytes,<br><br>procstat<br>write_bytes |

## 性能计数器指标

仅当 Windows 实例上安装了相应的性能计数器集时，才建议对实例使用性能计数器指标。

| 性能计数器指标名称                                    | 性能计数器集名称                 |
|----------------------------------------------|--------------------------|
| .NET CLR Exceptions # of Exceps Thrown       | .NET CLR Exceptions      |
| .NET CLR Exceptions # of Exceps Thrown/Sec   | .NET CLR Exceptions      |
| .NET CLR Exceptions # of Filters/Sec         | .NET CLR Exceptions      |
| .NET CLR Exceptions # of Finallys/Sec        | .NET CLR Exceptions      |
| .NET CLR Exceptions Throw to Catch Depth/Sec | .NET CLR Exceptions      |
| .NET CLR Interop # of CCWs                   | .NET CLR Interop         |
| .NET CLR Interop # of Stubs                  | .NET CLR Interop         |
| .NET CLR Interop # of TLB exports/Sec        | .NET CLR Interop         |
| .NET CLR Interop # of TLB imports/Sec        | .NET CLR Interop         |
| .NET CLR Interop # of Marshaling             | .NET CLR Interop         |
| .NET CLR Jit % Time in Jit                   | .NET CLR Jit             |
| .NET CLR Jit Standard Jit Failures           | .NET CLR Jit             |
| .NET CLR Loading % Time Loading              | .NET CLR Loading         |
| .NET CLR Loading Rate of Load Failures       | .NET CLR Loading         |
| .NET CLR LocksAndThreads Contention Rate/Sec | .NET CLR LocksAndThreads |
| .NET CLR LocksAndThreads Queue Length/Sec    | .NET CLR LocksAndThreads |
| .NET CLR Memory # Total Committed Bytes      | .NET CLR Memory          |
| .NET CLR Memory % Time in GC                 | .NET CLR Memory          |

| 性能计数器指标名称                                                  | 性能计数器集名称                    |
|------------------------------------------------------------|-----------------------------|
| .NET CLR Networking 4.0.0.0 HttpRequest Average Queue Time | .NET CLR Networking 4.0.0.0 |
| .NET CLR Networking 4.0.0.0 HttpRequest Aborted/Sec        | .NET CLR Networking 4.0.0.0 |
| .NET CLR Networking 4.0.0.0 HttpRequest Failed/Sec         | .NET CLR Networking 4.0.0.0 |
| .NET CLR Networking 4.0.0.0 HttpRequest Queued/Sec         | .NET CLR Networking 4.0.0.0 |
| APP_POOL_WAS Total Worker Process Ping Failures            | APP_POOL_WAS                |
| ASP.NET Application Restarts                               | ASP.NET                     |
| ASP.NET 请求被拒                                               | ASP.NET                     |
| ASP.NET 工作进程重启                                             | ASP.NET                     |
| ASP.NET 应用程序缓存 API 修剪                                      | ASP.NET Applications        |
| ASP.NET Applications % Managed Processor Time (estimated)  | ASP.NET Applications        |
| ASP.NET Applications Errors Total/Sec                      | ASP.NET Applications        |
| ASP.NET Applications Errors Unhandled During Execution/Sec | ASP.NET Applications        |
| ASP.NET Applications Requests in Application Queue         | ASP.NET Applications        |
| ASP.NET Applications Requests/Sec                          | ASP.NET Applications        |
| ASP.NET Request Wait Time                                  | ASP.NET                     |
| ASP.NET Requests Queued                                    | ASP.NET                     |
| 数据库 ==> 实例数据库缓存 % 命中                                       | 数据库 ==> 实例                  |
| 数据库 ==> 实例 I/O 数据库读取平均延迟                                   | 数据库 ==> 实例                  |
| 数据库 ==> 实例 I/O 数据库读取/秒                                     | 数据库 ==> 实例                  |
| 数据库 ==> 实例 I/O 日志写入平均延迟                                    | 数据库 ==> 实例                  |
| DirectoryServices DRA 挂起的复制操作                              | DirectoryServices           |
| DirectoryServices DRA 挂起的复制同步                              | DirectoryServices           |
| DirectoryServices LDAP 绑定时间                                | DirectoryServices           |
| DNS 递归查询/秒                                                 | DNS                         |
| DNS 递归查询失败/秒                                               | DNS                         |
| DNS TCP 接收查询/秒                                             | DNS                         |

| 性能计数器指标名称                                    | 性能计数器集名称                    |
|----------------------------------------------|-----------------------------|
| DNS 总接收查询/秒                                  | DNS                         |
| DNS 总发送响应/秒                                  | DNS                         |
| DNS UDP 接收查询/秒                               | DNS                         |
| HTTP Service Request Queues CurrentQueueSize | HTTP Service Request Queues |
| LogicalDisk % Free Space                     | LogicalDisk                 |
| LogicalDisk 平均值 Disk sec/Write               | LogicalDisk                 |
| LogicalDisk 平均值 Disk sec/Read                | LogicalDisk                 |
| LogicalDisk 平均值 Disk Queue Length            | LogicalDisk                 |
| Memory % Committed Bytes In Use              | 内存                          |
| Memory Available Mbytes                      | 内存                          |
| Memory Pages/Sec                             | 内存                          |
| 内存长期平均备用缓存寿命 ( 秒 )                           | 内存                          |
| Network Interface Bytes Total/Sec            | 网络接口                        |
| 网络接口接收的字节数/秒                                 | 网络接口                        |
| 网络接口发送的字节数/秒                                 | 网络接口                        |
| 网络接口当前带宽                                     | 网络接口                        |
| Paging File % Usage                          | Paging File                 |
| PhysicalDisk % Disk Time                     | PhysicalDisk                |
| PhysicalDisk Avg. Disk Queue Length          | PhysicalDisk                |
| PhysicalDisk Avg. Disk Sec/Read              | PhysicalDisk                |
| PhysicalDisk Avg. Disk Sec/Write             | PhysicalDisk                |
| PhysicalDisk Disk Read Bytes/Sec             | PhysicalDisk                |
| PhysicalDisk Disk Reads/Sec                  | PhysicalDisk                |
| PhysicalDisk Disk Write Bytes/Sec            | PhysicalDisk                |
| PhysicalDisk Disk Writes/Sec                 | PhysicalDisk                |
| Processor % Idle Time                        | 处理器                         |
| Processor % Interrupt Time                   | 处理器                         |
| Processor % Processor Time                   | 处理器                         |
| Processor % User Time                        | 处理器                         |
| SharePoint 基于磁盘的缓存 Blob 缓存填充率                | SharePoint 基于磁盘的缓存          |
| SharePoint 基于磁盘的缓存 Blob 缓存刷新/秒               | SharePoint 基于磁盘的缓存          |

| 性能计数器指标名称                                       | 性能计数器集名称                     |
|-------------------------------------------------|------------------------------|
| SharePoint 基于磁盘的缓存 Blob 缓存命中率                   | SharePoint 基于磁盘的缓存           |
| SharePoint 基于磁盘的缓存缓存压缩总数                        | SharePoint 基于磁盘的缓存           |
| SharePoint Foundation 执行时间/页面请求                 | SharePoint Foundation        |
| SharePoint 发布缓存发布缓存刷新/秒                         | SharePoint 发布缓存              |
| 安全系统范围的统计信息 Kerberos 身份验证                       | 安全系统范围的统计                    |
| 安全系统范围的统计信息 NTLM 身份验证                           | 安全系统范围的统计                    |
| SQLServer:Access Methods Forwarded Records/Sec  | SQLServer:Access Methods     |
| SQLServer:Access Methods Full Scans/Sec         | SQLServer:Access Methods     |
| SQLServer:Access Methods Page Splits/Sec        | SQLServer:Access Methods     |
| SQLServer:Buffer Manager Buffer cache hit Ratio | SQLServer:Buffer Manager     |
| SQLServer:Buffer Manager Page life Expectancy   | SQLServer:Buffer Manager     |
| SQLServer : 接收的数据库副本文件字节数/秒                     | SQLServer : 数据库副本            |
| SQLServer : 接收的数据库副本日志字节数/秒                     | SQLServer : 数据库副本            |
| SQLServer : 保留数据库副本日志以供撤消                       | SQLServer : 数据库副本            |
| SQLServer : 数据库副本日志发送队列                         | SQLServer : 数据库副本            |
| SQLServer : 数据库副本镜像写事务数/秒                       | SQLServer : 数据库副本            |
| SQLServer : 数据库副本恢复队列                           | SQLServer : 数据库副本            |
| SQLServer : 剩余数据库副本重做字节数                        | SQLServer : 数据库副本            |
| SQLServer : 数据库副本已重做的字节数/秒                      | SQLServer : 数据库副本            |
| SQLServer : 需要撤消的数据库副本总日志                       | SQLServer : 数据库副本            |
| SQLServer : 数据库副本事务延迟                           | SQLServer : 数据库副本            |
| SQLServer:General Statistics Processes Blocked  | SQLServer:General Statistics |
| SQLServer:General Statistics User Connections   | SQLServer:General Statistics |
| SQLServer:Latches Average Latch Wait Time (ms)  | SQLServer:Latches            |
| SQLServer:Locks Average Wait Time (ms)          | SQLServer:Locks              |
| SQLServer:Locks Lock Timeouts/Sec               | SQLServer:Locks              |
| SQLServer:Locks Lock Waits/Sec                  | SQLServer:Locks              |
| SQLServer:Locks Number of Deadlocks/Sec         | SQLServer:Locks              |
| SQLServer:Memory Manager Memory Grants Pending  | SQLServer:Memory Manager     |
| SQLServer:SQL Statistics Batch Requests/Sec     | SQLServer:SQL Statistics     |



| 性能计数器指标名称                                        | 性能计数器集名称                 |
|--------------------------------------------------|--------------------------|
| SQLServer:SQL Statistics SQL Compilations/Sec    | SQLServer:SQL Statistics |
| SQLServer:SQL Statistics SQL Re-Compilations/Sec | SQLServer:SQL Statistics |
| System Processor Queue Length                    | 系统                       |
| TCPv4 Connections Established                    | TCPv4                    |
| TCPv6 Connections Established                    | TCPv6                    |
| W3SVC_W3WP File Cache Flushes                    | W3SVC_W3WP               |
| W3SVC_W3WP File Cache Misses                     | W3SVC_W3WP               |
| W3SVC_W3WP Requests/Sec                          | W3SVC_W3WP               |
| W3SVC_W3WP URI Cache Flushes                     | W3SVC_W3WP               |
| W3SVC_W3WP URI Cache Misses                      | W3SVC_W3WP               |
| Web Service Bytes Received/Sec                   | Web 服务                   |
| Web Service Bytes Sent/Sec                       | Web 服务                   |
| Web Service Connection Attempts/Sec              | Web 服务                   |
| Web Service Current Connections                  | Web 服务                   |
| Web Service Get Requests/Sec                     | Web 服务                   |
| Web Service Post Requests/Sec                    | Web 服务                   |

# 跨账户跨区域的 CloudWatch 控制台

您可以向 CloudWatch 控制台添加跨账户功能。利用此功能，您可以跨账户查看控制面板、警报、指标和自动控制面板，而无需登录和注销不同的账户。

然后，您可以创建控制面板以将来自多个 Amazon 账户和多个 Amazon 区域的 CloudWatch 数据汇总到单个控制面板中。您还可以在某个账户中创建告警，以监控不同账户中的指标。

许多组织都将其 Amazon 资源部署到多个账户中来提供账单和安全边界。在这种情况下，我们建议您将一个或多个账户指定为监控账户，并在这些账户中构建跨账户控制面板。

跨账户功能与 Amazon Organizations 集成，可帮助您高效地构建跨账户控制面板。

## 跨区域功能

跨区域功能现已自动内置。您无需执行任何额外步骤即可在同一图表或同一控制面板上的单个账户中显示来自不同区域的指标。由于告警不支持跨区域功能，您无法在某个区域中创建用以监视其他区域内指标的告警。

## 主题

- 在 [CloudWatch 中启用跨账户功能](#) (p. 654)
- ( 可选 ) 与 [Amazon Organizations 集成](#) (p. 656)
- 对 [CloudWatch 跨账户设置问题进行故障排除](#) (p. 657)
- [使用跨账户后禁用和清理](#) (p. 657)

## 在 CloudWatch 中启用跨账户功能

要在 CloudWatch 控制台中设置跨账户功能，请使用 CloudWatch 控制台设置您的共享账户和监控账户。

### 设置共享账户

您必须在每个账户中启用共享，以使数据对监控账户可用。

这将对所有在您共享的账户中查看跨账户控制面板的用户授予您在步骤 5 中选择的只读权限（如果该用户在您共享的账户中具有相应权限）。

### 使您的账户能够与其他账户共享 CloudWatch 数据

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Settings (设置)，然后选择 Configure (配置)。
3. 选择 Share data (共享数据)。
4. 对于 Sharing (共享)，选择 Specific accounts (特定账户)，然后输入要与之共享数据的账户的 ID。

您在此处指定的任何账户都可以查看您的账户的 CloudWatch 数据。仅指定您知道并信任的账户的 ID。

5. 对于 Permissions (权限)，使用下列选项之一来指定共享数据的方式：
  - Provide read-only access to your CloudWatch metrics, dashboards, and alarms (提供对 CloudWatch 指标、控制面板和告警的只读访问权限)。此选项使监控账户能够创建跨账户控制面板，这些控制面板具有包含您账户中的 CloudWatch 数据的小组件。
  - 包括 CloudWatch 自动控制面板。如果选择此选项，则监控账户中的用户还可以查看此账户的自动控制面板中的信息。有关更多信息，请参阅 [Amazon CloudWatch 入门](#) (p. 23)。

- Include X-Ray read-only access for ServiceLens (包括 ServiceLens 的 X-Ray 只读访问权限)。如果您选择此选项, 监控账户中的用户还可以查看 ServiceLens 服务地图, 以及此账户中的 X-Ray 跟踪信息。有关更多信息, 请参阅 [使用 ServiceLens 监控应用程序的运行状况](#) (p. 194)。
  - Full read-only access to everything in your account (对账户中所有内容的完全只读访问权)。此选项使您用于共享的账户能够创建跨账户控制面板, 这些控制面板具有包含您账户中的 CloudWatch 数据的小组件。它还使这些账户能够更深入地查看您的账户, 并在其他 Amazon 服务的控制台中查看您账户的数据。
6. 选择 Launch CloudFormation template (启动 CloudFormation 模板)。

在确认屏幕中, 键入 **Confirm**, 并选择 Launch template (启动模板)。
  7. 选中 I acknowledge...(我确认...) 复选框, 然后选择 Create stack (创建堆栈)。

### 与整个企业共享

通过完成上述过程, 创建一个 IAM 角色来使您的账户能够与某个账户共享数据。您可以创建或编辑一个 IAM 角色, 该角色将与企业中所有的账户共享您的数据。仅在您知道并信任组织中所有的账户的情况下执行此操作。

如果用户在与您共享的帐户中具有相应权限, 这将向在您的共享账户中查看跨账户控制面板的所有用户授予在上一过程第 5 步所示策略中列出的只读权限。

### 与企业中所有的账户共享您的 CloudWatch 账户数据

1. 如果尚未执行此操作, 请完成上述过程来与某个 Amazon 账户共享您的数据。
2. 登录 Amazon Web Services Management Console, 然后通过以下网址打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
3. 在导航窗格中, 选择角色。
4. 在角色列表中, 选择 CloudWatch-CrossAccountSharingRole。
5. 选择信任关系, 然后选择编辑信任关系。

您将看到与以下内容类似的策略:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::123456789012:root"
 },
 "Action": "sts:AssumeRole"
 }
]
}
```

6. 将策略更改为以下内容, 并将 **org-id** 替换为您组织的 ID。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": "*"
 },
 "Action": "sts:AssumeRole",
 "Condition": {
 "StringEquals": {

```

```
 "aws:PrincipalOrgID": "org-id"
 }
 }
 }
]
```

7. 选择 Update Trust Policy。

#### 设置监控帐户

如果要查看跨账户 CloudWatch 数据，请启用每个监控账户。

完成以下过程后，CloudWatch 将创建一个服务相关角色，CloudWatch 在监控账户中使用该角色来访问从其他账户共享的数据。此服务相关角色称为 AWSServiceRoleForCloudWatchCrossAccount。有关更多信息，请参阅 [为 CloudWatch 使用服务相关角色 \(p. 863\)](#)。

#### 使您的账户能够查看跨账户 CloudWatch 数据

1. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Settings ( 设置 )，然后在 Cross-account cross-region ( 跨账户跨区域 ) 部分中，选择 Configure ( 配置 )。
3. 在 View cross-account cross-region ( 查看跨账户跨区域 ) 部分，选择 Enable ( 启用 )，然后选择 Show selector in the console ( 在控制台中显示选择器 ) 复选框以在您绘制指标图表或创建告警时，让账户选择器显示在 CloudWatch 控制台中。
4. 在 View cross-account cross-region ( 跨账户跨区域查看 ) 中，选择下列选项之一：
  - Account Id Input ( 账户 ID 输入 )。在查看跨账户的数据时，每当您要切换账户时，此选项都会提示您手动输入账户 ID。
  - Amazon Organization account selector ( Amazon Organizations 账户选择器 )。此选项可显示您在完成与 Organizations 的跨账户集成时指定的账户。在您下次使用控制台时，CloudWatch 会为您显示这些账户的下拉列表，供您在查看跨账户数据时进行选择。

为此，您必须已先使用企业管理账户允许 CloudWatch 查看组织中的账户列表。有关更多信息，请参阅 [\( 可选 \) 与 Amazon Organizations 集成 \(p. 656\)](#)。

  - Custom account selector ( 自定义账户选择器 )。此选项提示您输入账户 ID 的列表。在您下次使用控制台时，CloudWatch 会为您显示这些账户的下拉列表，供您在查看跨账户数据时进行选择。

此外，您可为其中每个账户输入一个标签，以便在选择要查看的账户时帮助您识别它们。

用户在此处进行的账户选择器设置仅为该用户保留，而不为监视账户中的所有其他用户保留。

5. 请选择 Enable。

完成此设置后，您可以创建跨账户控制面板。有关更多信息，请参阅 [跨账户跨区域的控制面板 \(p. 30\)](#)。

## ( 可选 ) 与 Amazon Organizations 集成

如果要跨账户功能与 Amazon Organizations 集成，您必须使组织中的所有账户对监控账户可用。

#### 启用跨账户 CloudWatch 功能以访问企业中的所有账户

1. 登录到企业的管理账户。
2. 访问 <https://console.aws.amazon.com/cloudwatch/>，打开 CloudWatch 控制台。

3. 在导航窗格中，选择 Settings (设置)，然后选择 Configure (配置)。
4. 对于 Grant permission to view the list of accounts in the organization (授予查看组织中账户列表的权限)，选择要提示输入账户 ID 列表的 Specific accounts (特定账户)。您组织中的账户列表仅与您在此处指定的账户共享。
5. 选择 Share organization account list (共享组织账户列表)。
6. 选择 Launch CloudFormation template (启动 CloudFormation 模板)。

在确认屏幕中，键入 **Confirm**，并选择 Launch template (启动模板)。

## 对 CloudWatch 跨账户设置问题进行故障排除

这部分包含有关 CloudWatch 中跨账户控制台部署的故障排除提示。

我在显示跨账户的数据时收到访问被拒绝错误

请检查以下事项：

- 您的监控账户应具有一个名为 AWSServiceRoleForCloudWatchCrossAccount 的角色。如果没有此角色，则需要创建它。有关更多信息，请参阅 [Set Up a Monitoring Account \(p. 656\)](#)。
- 每个共享账户应具有一个名为 CloudWatch-CrossAccountSharingRole 的角色。如果没有此角色，则需要创建它。有关更多信息，请参阅 [Set Up A Sharing Account \(p. 654\)](#)。
- 此共享角色必须信任监控账户。

确认是否已为 CloudWatch 跨账户控制台正确设置您的角色

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择角色。
3. 在角色列表中，确保需要的角色已存在。在共享账户中，查找 CloudWatch-CrossAccountSharingRole。在监控账户中，查找 AWSServiceRoleForCloudWatchCrossAccount。
4. 如果您正在使用共享账户，并且 CloudWatch-CrossAccountSharingRole 已存在，请选择 CloudWatch-CrossAccountSharingRole。
5. 选择信任关系，然后选择编辑信任关系。
6. 确认策略已列出监控账户的账户 ID 或监控账户所属组织的组织 ID。

控制台中未显示账户下拉菜单

首先，检查您是否已创建正确的 IAM 角色，如前面的故障排除部分中所述。如果已正确设置这些角色，请确保您已启用此账户来查看跨账户的数据，如[Enable Your Account to View Cross-Account Data \(p. 656\)](#)中所述。

## 使用跨账户后禁用和清理

要禁用 CloudWatch 的跨账户功能，请执行下列步骤。

步骤 1：删除跨账户堆栈或角色

最佳方法是删除用于启用跨账户功能的 Amazon CloudFormation 堆栈。

- 在每个共享账户中，删除 CloudWatch-CrossAccountSharingRole 堆栈。
- 如果您使用 Amazon Organizations 为企业中的所有账户启用跨账户功能，请删除企业管理账户中的 CloudWatch-CrossAccountListAccountsRole 堆栈。

如果您没有使用 Amazon CloudFormation 堆栈以启用跨账户功能，请执行以下操作：

- 在每个共享账户中，删除 CloudWatch-CrossAccountSharingRole IAM 角色。
- 如果您使用 Amazon Organizations 为企业中的所有账户启用跨账户功能，请删除企业管理账户中的 CloudWatch-CrossAccountSharing-ListAccountsRole IAM 角色。

步骤 2：删除服务相关角色

在监控账户中，删除 AWSServiceRoleForCloudWatchCrossAccount 服务相关的 IAM 角色。

# 使用 CloudWatch 代理从 Amazon EC2 实例和本地部署服务器中收集指标和日志

您可以通过统一 CloudWatch 代理执行以下操作：

- 跨操作系统从 Amazon EC2 实例中收集内部系统级指标。除了 EC2 实例的指标之外，这些指标还可以包括来宾中的指标。可以收集的其他指标列在[CloudWatch 代理收集的指标 \(p. 729\)](#)中。
- 从本地服务器中收集系统级别指标。这些服务器可能包括混合环境中的服务器以及不是由 Amazon 管理的服务器。
- 使用 StatsD 和 collectd 协议从应用程序或服务中检索自定义指标。StatsD 在 Linux 服务器和运行 Windows Server 的服务器上都受支持。collectd 仅在 Linux 服务器上受支持。
- 从运行 Linux 或 Windows Server 的 Amazon EC2 实例和本地部署服务器收集日志。

## Note

CloudWatch 代理不支持从 FIFO 管道收集日志。

您可以在 CloudWatch 中存储和查看使用 CloudWatch 代理收集的指标，就像任何其他 CloudWatch 指标一样。CloudWatch 代理收集的指标的默认命名空间为 `cwAgent`，不过您可以在配置该代理时指定其他命名空间。

由统一 CloudWatch 代理收集的日志在 Amazon CloudWatch Logs 中处理和存储，就像较旧的 CloudWatch Logs 代理收集的日志一样。有关 CloudWatch Logs 定价的信息，请参阅[Amazon CloudWatch 定价](#)。

CloudWatch 代理收集的指标按自定义指标计费。有关 CloudWatch 指标定价的信息，请参阅[Amazon CloudWatch 定价](#)。

CloudWatch 代理是 MIT 许可证下的开源代理，并且托管在[GitHub](#)上。如果您想要构建、自定义或贡献 CloudWatch 代理，请参阅 GitHub 存储库以获取最新说明。如果您认为您发现了潜在的安全问题，请勿将其发布到 GitHub 或任何公共论坛上。请按照[漏洞报告](#)中的说明进行操作，或者[直接向 Amazon 安全发送电子邮件](#)。

本节中的步骤介绍如何在 Amazon EC2 实例和本地部署服务器上安装统一 CloudWatch 代理。有关 CloudWatch 代理能够收集的指标的更多信息，请参阅[CloudWatch 代理收集的指标 \(p. 729\)](#)。

## 支持的操作系统

在以下操作系统的 x86-64 架构上支持 CloudWatch 代理：

- Amazon Linux 2014.03.02 或更高版本
- Amazon Linux 2
- Ubuntu Server 版本 20.04、18.04、16.04 和 14.04
- CentOS 版本 8 stream 和版本 7
- Red Hat Enterprise Linux ( RHEL ) 版本 8.4、8.3、8.2、8.1、8.0、7.7、7.6、7.5、7.4、7.2 和 7.0
- Debian 版本 10
- SUSE Linux Enterprise Server (SLES) 版本 15 和版本 12
- Oracle Linux 版本 8.6、8.4、8.3、8.2、8.1、7.8、7.6 和 7.5

- macOS，包括 EC2 Mac1 实例
- 64 位版本的 Windows Server 2022、Windows Server 2019、Windows Server 2016 和 Windows Server 2012 R2
- 在 x86-64 计算机上运行的 64 位 Windows 10

在以下操作系统的 ARM64 架构上支持该代理：

- Amazon Linux 2
- Ubuntu Server 版本 20.04 和 18.04
- Red Hat Enterprise Linux (RHEL) 版本 7.6
- SUSE Linux Enterprise Server 15

#### 安装过程概览

您可以使用命令行手动下载并安装 CloudWatch 代理，也可以将其与 SSM 集成。使用这两种方法之一安装 CloudWatch 代理的一般流程如下所示：

1. 创建使代理能够从服务器中收集指标且（可选）与 Amazon Systems Manager 集成的 IAM 角色或用户。
2. 下载代理软件包。
3. 修改 CloudWatch 代理配置文件并指定要收集的指标。
4. 在服务器上安装并启动代理。当您在 EC2 实例上安装代理时，将会附加在步骤 1 中创建的 IAM 角色。当您在本地部署服务器上安装代理时，您会指定一个命名的配置文件，其中包含您在步骤 1 中创建的 IAM 用户的凭证。

#### 目录

- [安装 CloudWatch 代理 \(p. 660\)](#)
- [创建 CloudWatch 代理配置文件 \(p. 682\)](#)
- [CloudWatch 代理收集的指标 \(p. 729\)](#)
- [CloudWatch 代理对 OpenTelemetry 的支持 \(p. 737\)](#)
- [CloudWatch 代理的常见场景 \(p. 741\)](#)
- [CloudWatch 代理故障排除 \(p. 748\)](#)

## 安装 CloudWatch 代理

您可以使用命令行以及 Amazon S3 下载链接、使用 SSM 或使用 Amazon CloudFormation 模板来下载并安装 CloudWatch 代理。

#### 目录

- [使用命令行安装 CloudWatch 代理 \(p. 660\)](#)
- [使用 Amazon Systems Manager 安装 CloudWatch 代理 \(p. 669\)](#)
- [验证 CloudWatch 代理软件包的签名 \(p. 681\)](#)

## 使用命令行安装 CloudWatch 代理

使用以下主题来下载、配置和安装 CloudWatch 代理软件包。

#### 主题

- [使用命令行下载和配置 CloudWatch 代理 \(p. 661\)](#)



- [创建 IAM 角色和用户以用于 CloudWatch 代理 \(p. 663\)](#)
- [在服务器上安装和运行 CloudWatch 代理 \(p. 665\)](#)

## 使用命令行下载和配置 CloudWatch 代理

使用以下步骤来下载 CloudWatch 代理软件包，创建 IAM 角色或用户，以及（可选）根据需要修改常见配置文件。

### 下载 CloudWatch 代理软件包

您可以使用 Amazon S3 下载链接下载 CloudWatch 代理软件包。根据您的架构和平台，从该表中选择下载链接。

| 架构    | 平台                            | 下载链接                                                                                                                                                                                                                                                  | 签名文件链接                                                                                                                                                                                                                                                        |
|-------|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| amd64 | Amazon Linux 和 Amazon Linux 2 | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm</a> | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig</a> |
| amd64 | Centos                        | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm</a>             | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig</a>             |
| amd64 | Redhat                        | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm</a>             | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig</a>             |
| amd64 | SUSE                          | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm</a>                 | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig</a>                 |
| amd64 | Debian                        | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb</a>             | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig</a>             |
| amd64 | Ubuntu                        | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb</a>             | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig</a>             |
| amd64 | Windows                       | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi</a>           | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig</a>           |

| 架构    | 平台             | 下载链接                                                                                                                                                                                                                                                  | 签名文件链接                                                                                                                                                                                                                                                        |
|-------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ARM64 | Amazon Linux 2 | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm</a> | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig</a> |
| ARM64 | Redhat         | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm</a>             | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig</a>             |
| ARM64 | Ubuntu         | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb</a>             | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig</a>             |

## 使用命令行下载和安装 CloudWatch 代理软件包

### 1. 下载 CloudWatch 代理。

在 Linux 服务器上，输入以下命令。对于 `download-link`，请使用上表中的相应下载链接。

```
wget download-link
```

在运行 Windows Server 的服务器上，下载以下文件：

- 在下载该软件包后，您可以选择验证软件包签名。有关更多信息，请参阅[验证 CloudWatch 代理软件包的签名 \(p. 681\)](#)。
- 安装 软件包。如果已在 Linux 服务器上下载 RPM 程序包，请更改为包含程序包的目录，并输入以下内容：

```
sudo rpm -U ./amazon-cloudwatch-agent.rpm
```

如果已在 Linux 服务器上下载 DEB 程序包，请更改为包含程序包的目录，并输入以下内容：

```
sudo dpkg -i -E ./amazon-cloudwatch-agent.deb
```

如果已运行在 Windows Server 的服务器上下载 MSI 程序包，请更改为包含程序包的目录，并输入以下内容：

```
msiexec /i amazon-cloudwatch-agent.msi
```

此命令还可在 PowerShell 中发挥作用。有关 MSI 命令选项的更多信息，请参阅 Microsoft Windows 文档中的[命令行选项](#)。

## 创建和修改代理配置文件

在下载 CloudWatch 代理后，您必须先创建配置文件，然后再在任何服务器上启动该代理。有关更多信息，请参阅[创建 CloudWatch 代理配置文件 \(p. 682\)](#)。

## 创建 IAM 角色和用户以用于 CloudWatch 代理

要访问 Amazon 资源，需要具有相应的权限。您创建 IAM 角色和/或 IAM 用户，以授予 CloudWatch 代理将指标写入 CloudWatch 所需的权限。如果您打算在 Amazon EC2 实例上使用代理，则必须创建 IAM 角色。如果您打算在本地部署服务器上使用代理，则必须创建 IAM 用户。

### Note

我们最近修改了以下过程，具体来说就是使用由 Amazon 创建的新 CloudWatchAgentServerPolicy 和 CloudWatchAgentAdminPolicy 策略，而不需要客户自行创建这些策略。将文件写入 Parameter Store 和从 Parameter Store 中下载文件时，由 Amazon 创建的策略仅支持名称以 AmazonCloudWatch- 开头的文件。如果您的 CloudWatch 代理配置文件的文件名不以 AmazonCloudWatch- 开头，则这些策略不能用于将文件写入 Parameter Store 或从 Parameter Store 下载文件。

如果您打算在 Amazon EC2 实例上运行 CloudWatch 代理，请使用以下步骤创建必要的 IAM 角色。该角色提供从实例中读取信息并将其写入到 CloudWatch 的权限。

### 创建在 EC2 实例上运行 CloudWatch 代理所需的 IAM 角色

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧的导航窗格中，选择 Roles (角色)，然后选择 Create role (创建角色)。
3. 确保在 Trusted entity type (受信任实体的类型) 下选择了 Amazon service (服务)。
4. 对于 Use case (使用案例)，在 Common use cases (常见使用案例) 下选择 EC2，
5. 选择 Next (下一步)。
6. 在策略列表中，选中 CloudWatchAgentServerPolicy 旁边的复选框。如有必要，请使用搜索框查找该策略。
7. 选择 Next (下一步)。
8. 在 Role name (角色名称) 中，输入角色的名称，例如 **CloudWatchAgentServerRole**。(可选) 为其指定说明。然后选择 Create role (创建角色)。

将立即创建该角色。

9. (可选) 如果代理将向 CloudWatch Logs 发送日志，并且您想要代理能够为这些日志组设置保留策略，则需要将 logs:PutRetentionPolicy 权限添加到角色。有关更多信息，请参阅[允许 CloudWatch 代理设置日志保留策略 \(p. 664\)](#)。

如果您打算在本地部署服务器上运行 CloudWatch 代理，请使用以下步骤创建必要的 IAM 用户。

### 创建 CloudWatch 代理在本地部署服务器上运行所需的 IAM 用户

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧的导航窗格中，选择 Users (用户)，然后选择 Add user (添加用户)。
3. 为新用户输入用户名。
4. 选择 Access key - Programmatic access (访问密钥 - 编程访问)，然后选择 Next: Permissions (下一步: 权限)。
5. 选择 Attach existing policies directly (直接附上现有策略)。
6. 在策略列表中，选中 CloudWatchAgentServerPolicy 旁边的复选框。如有必要，请使用搜索框查找该策略。
7. 请选择下一步：标签。

8. 可以选择为新 IAM 用户创建标签，然后选择 Next: Review ( 下一步: 审核 )。
9. 确认列出了正确的策略，然后选择 Create user ( 创建用户 )。
10. 在新用户的名称旁边，选择显示。将访问密钥和私有密钥复制到一个文件以便在安装该代理时使用。选择 Close ( 关闭 )。

## 允许 CloudWatch 代理设置日志保留策略

您可以配置 CloudWatch 代理，以便为接收其发送的日志事件的日志组设置保留策略。如果您这样做，您必须向代理使用的 IAM 角色或用户授予 `logs:PutRetentionPolicy`。代理使用 IAM 角色在 Amazon EC2 实例上运行，使用本地部署服务器的 IAM 用户。

向 CloudWatch 代理的 IAM 角色授予设置日志保留策略的权限

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧导航窗格中，选择 Roles (角色)。
3. 在搜索框中，键入 CloudWatch 代理的 IAM 角色名称的开头部分。您在创建角色时选择了此名称。其可能被命名为 `CloudWatchAgentServerRole`。

当您看到角色时，选择角色的名称。

4. 在 Permissions ( 权限 ) 选项卡中，依次选择 Add permissions ( 添加权限 ) 和 Create inline policy ( 创建内联策略 )。
5. 选择 JSON 选项卡并将以下策略复制到框中，替换框中默认的 JSON:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "logs:PutRetentionPolicy",
 "Resource": "*"
 }
]
}
```

6. 选择 Review policy ( 查看策略 )。
7. 对于 Name ( 名称 )，输入 **CloudWatchAgentPutLogsRetention** 或类似内容，然后选择 Create policy ( 创建策略 )。

向 CloudWatch 代理的 IAM 用户授予设置日志保留策略的权限

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧导航窗格中，选择 Users (用户)。
3. 在搜索框中，键入 CloudWatch 代理的 IAM 用户名称的开头部分。您在创建用户时选择了此名称。

当您看到用户时，选择用户的名称。

4. 在权限选项卡中，选择 添加内联策略。
5. 选择 JSON 选项卡并将以下策略复制到框中，替换框中默认的 JSON:

```
{
 "Version": "2012-10-17",
 "Statement": [
```

```
{
 "Effect": "Allow",
 "Action": "logs:PutRetentionPolicy",
 "Resource": "*"
}
```

6. 选择 Review policy (查看策略)。
7. 对于 Name (名称)，输入 **CloudWatchAgentPutLogsRetention** 或类似内容，然后选择 Create policy (创建策略)。

## 在服务器上安装和运行 CloudWatch 代理

在创建所需的代理配置文件并创建 IAM 角色或 IAM 用户后，使用该配置执行以下步骤在服务器上安装和运行代理。首先，将 IAM 角色或 IAM 用户附加到将要运行代理的服务器上。然后，在该服务器上，下载代理软件包并使用您创建的代理配置文件启动它。

### 使用 S3 下载链接下载 CloudWatch 代理软件包

在将要运行代理的每台服务器上，下载代理软件包。根据您的架构和平台，从该表中选择下载链接。

| 架构    | 平台                            | 下载链接                                                                                                                                                                                                                                                  | 签名文件链接                                                                                                                                                                                                                                                        |
|-------|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| amd64 | Amazon Linux 和 Amazon Linux 2 | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm</a> | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig</a> |
| amd64 | Centos                        | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm</a>             | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig</a>             |
| amd64 | Redhat                        | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm</a>             | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig</a>             |
| amd64 | SUSE                          | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm</a>                 | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig</a>                 |
| amd64 | Debian                        | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb</a>             | <a href="https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig">https://s3.cn-north-1.amazonaws.com.cn/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig</a>             |
| amd64 | Ubuntu                        | <a href="https://s3.cn-north-1.amazonaws.com.cn/">https://s3.cn-north-1.amazonaws.com.cn/</a>                                                                                                                                                         | <a href="https://s3.cn-north-1.amazonaws.com.cn/">https://s3.cn-north-1.amazonaws.com.cn/</a>                                                                                                                                                                 |

| 架构    | 平台                | 下载链接                                                                                                                                | 签名文件链接                                                                                                                                      |
|-------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
|       |                   | amazoncloudwatch-agent/<br>ubuntu/amd64/latest/amazon-<br>cloudwatch-agent.deb                                                      | amazoncloudwatch-agent/<br>ubuntu/amd64/latest/amazon-<br>cloudwatch-agent.deb.sig                                                          |
| amd64 | Windows           | https://s3.cn-<br>north-1.amazonaws.com.cn/<br>amazoncloudwatch-agent/<br>windows/amd64/latest/amazon-<br>cloudwatch-agent.msi      | https://s3.cn-<br>north-1.amazonaws.com.cn/<br>amazoncloudwatch-agent/<br>windows/amd64/latest/amazon-<br>cloudwatch-agent.msi.sig          |
| ARM64 | Amazon Linux<br>2 | https://s3.cn-<br>north-1.amazonaws.com.cn/<br>amazoncloudwatch-agent/<br>amazon_linux/arm64/latest/<br>amazon-cloudwatch-agent.rpm | https://s3.cn-<br>north-1.amazonaws.com.cn/<br>amazoncloudwatch-agent/<br>amazon_linux/arm64/<br>latest/amazon-cloudwatch-<br>agent.rpm.sig |
| ARM64 | Redhat            | https://s3.cn-<br>north-1.amazonaws.com.cn/<br>amazoncloudwatch-agent/<br>redhat/arm64/latest/amazon-<br>cloudwatch-agent.rpm       | https://s3.cn-<br>north-1.amazonaws.com.cn/<br>amazoncloudwatch-agent/<br>redhat/arm64/latest/amazon-<br>cloudwatch-agent.rpm.sig           |
| ARM64 | Ubuntu            | https://s3.cn-<br>north-1.amazonaws.com.cn/<br>amazoncloudwatch-agent/<br>ubuntu/arm64/latest/amazon-<br>cloudwatch-agent.deb       | https://s3.cn-<br>north-1.amazonaws.com.cn/<br>amazoncloudwatch-agent/<br>ubuntu/arm64/latest/amazon-<br>cloudwatch-agent.deb.sig           |

### 使用命令行在 Amazon EC2 实例上安装 CloudWatch 代理

1. 下载 CloudWatch 代理。对于 Linux 服务器，请输入以下命令。对于 [download-link](#)，请使用上表中的相应下载链接。

```
wget download-link
```

对于运行 Windows Server 的服务器，请下载以下文件：

2. 在下载该软件包后，您可以选择验证软件包签名。有关更多信息，请参阅[验证 CloudWatch 代理软件包的签名 \(p. 681\)](#)。
3. 安装 软件包。如果已在 Linux 服务器上下载 RPM 程序包，请更改为包含程序包的目录，并输入以下内容：

```
sudo rpm -U ./amazon-cloudwatch-agent.rpm
```

如果已在 Linux 服务器上下载 DEB 程序包，请更改为包含程序包的目录，并输入以下内容：

```
sudo dpkg -i -E ./amazon-cloudwatch-agent.deb
```

如果已运行在 Windows Server 的服务器上下载 MSI 程序包，请更改为包含程序包的目录，并输入以下内容：

```
msiexec /i amazon-cloudwatch-agent.msi
```

此命令还可在 PowerShell 中发挥作用。有关 MSI 命令选项的更多信息，请参阅 Microsoft Windows 文档中的[命令行选项](#)。

## ( 在 EC2 实例上安装 ) 附加 IAM 角色

要使 CloudWatch 代理能够从实例发送数据，您必须将 IAM 角色附加到实例。要附加的角色是 CloudWatchAgentServerRole。

有关将 IAM 角色附加到实例的更多信息，请参阅适用于 Windows 实例的 Amazon EC2 用户指南中的[将 IAM 角色附加到实例](#)。

## ( 安装在本地部署服务器上 ) 指定 IAM 凭证和 Amazon 区域

要使 CloudWatch 代理能够从本地部署服务器发送数据，您必须指定先前创建的 IAM 用户的访问密钥和私有密钥。有关创建此用户的更多信息，请参阅[创建 IAM 角色和用户以用于 CloudWatch 代理](#) (p. 663)。

您还必须使用 Amazon 配置文件的 [AmazonCloudWatchAgent] 部分中的 region 字段指定要将指标发送到的 Amazon 区域，如下示例中所示。

```
[profile AmazonCloudWatchAgent]
region = us-west-1
```

下面是使用 aws configure 命令为 CloudWatch 代理创建命名配置文件的示例。该示例假设您使用默认配置文件名称 AmazonCloudWatchAgent。

为 CloudWatch 代理创建 AmazonCloudWatchAgent 配置文件

1. 如果您还没有这样做，请在服务器上安装 Amazon Command Line Interface。有关更多信息，请参阅[安装 Amazon CLI](#)。
2. 在 Linux 服务器上，输入以下命令并按照提示进行操作：

```
sudo aws configure --profile AmazonCloudWatchAgent
```

在 Windows Server 上，以管理员身份打开 PowerShell，输入以下命令并按照提示进行操作。

```
aws configure --profile AmazonCloudWatchAgent
```

## 验证互联网访问权限

您的 Amazon EC2 实例必须具有出站互联网访问权限，才能将数据发送到 CloudWatch 或 CloudWatch Logs。有关如何配置互联网访问权限的更多信息，请参阅 Amazon VPC 用户指南中的[互联网网关](#)。

要在您的代理上配置的终端节点和端口如下所示：

- 如果要使用代理收集指标，则必须将相应区域的 CloudWatch 端点加入白名单。这些端点在 Amazon Web Services 一般参考中的[Amazon CloudWatch](#)中列出。
- 如果要使用代理收集日志，则必须将相应区域的 CloudWatch Logs 端点加入白名单。这些端点在 Amazon Web Services 一般参考中的[Amazon CloudWatch Logs](#)中列出。
- 如果使用 Systems Manager 安装代理或使用 Parameter Store 存储配置文件，您必须将相应区域的 Systems Manager 端点加入白名单。这些端点在 Amazon Web Services 一般参考中的[Amazon Systems Manager](#)中列出。



## ( 可选 ) 修改代理的通用配置或区域信息

CloudWatch 代理包含一个名为 `common-config.toml` 的配置文件。您可以 ( 可选 ) 使用该文件指定代理和区域信息。

在运行 Linux 的服务器上, 该文件位于 `/opt/aws/amazon-cloudwatch-agent/etc` 目录中。在运行 Windows Server 的服务器上, 该文件位于 `C:\ProgramData\Amazon\AmazonCloudWatchAgent` 目录中。

默认 `common-config.toml` 如下所示。

```
This common-config is used to configure items used for both ssm and cloudwatch access

Configuration for shared credential.
Default credential strategy will be used if it is absent here:
Instance role is used for EC2 case by default.
AmazonCloudWatchAgent profile is used for the on-premises case by default.
[credentials]
shared_credential_profile = "{profile_name}"
shared_credential_file= "{file_name}"

Configuration for proxy.
System-wide environment-variable will be read if it is absent here.
i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
Note: system-wide environment-variable is not accessible when using ssm run-command.
Absent in both here and environment-variable means no proxy will be used.
[proxy]
http_proxy = "{http_url}"
https_proxy = "{https_url}"
no_proxy = "{domain}"
```

最初将注释所有行。要设置凭证配置文件或代理设置, 请从该行中删除 # 并指定一个值。您可以手动编辑该文件, 或者使用 Systems Manager 中的 RunShellScript Run Command 执行该操作:

- `shared_credential_profile` – 对于本地部署服务器, 此行指定 IAM 用户凭证配置文件, 以用于将数据发送到 CloudWatch。如果您将此行注释掉, 则会使用 `AmazonCloudWatchAgent`。有关创建此配置文件的更多信息, 请参阅 ( [安装在本地部署服务器上](#) ) 指定 IAM 凭证和 [Amazon 区域](#) (p. 667)。

在 EC2 实例上, 您可以使用此行让 CloudWatch 代理将数据从该实例发送到不同 Amazon 区域中的 CloudWatch。要执行此操作, 请指定一个包含 `region` 字段的命名配置文件, 该字段指定要发送到的区域的名称。

如果指定 `shared_credential_profile`, 您还必须从 `[credentials]` 行开头删除 #。

- `shared_credential_file` – 要让代理在位于默认路径以外的路径中的文件中查找凭证, 请在此处指定完整的路径和文件名。在 Linux 上, 默认路径为 `/root/.aws`; 在 Windows Server 上, 默认路径为 `C:\Users\Administrator\aws`。

下面的第一个示例显示对 Linux 服务器有效的 `shared_credential_file` 行的语法, 第二个示例对 Windows Server 有效。在 Windows Server 上, 您必须转义 \ 字符。

```
shared_credential_file= "/usr/username/credentials"
```

```
shared_credential_file= "C:\\Documents and Settings\\username\\aws\\credentials"
```

如果指定 `shared_credential_file`, 您还必须从 `[credentials]` 行开头删除 #。

- 代理设置 – 如果您的服务器使用 HTTP 或 HTTPS 代理联系 Amazon 服务, 请在 `http_proxy` 和 `https_proxy` 字段中指定这些代理。如果应从代理中排除某些 URL, 请在 `no_proxy` 字段中指定这些 URL 并以逗号分隔。



## 使用命令行启动 CloudWatch 代理

可以执行以下步骤以使用命令行在服务器上启动 CloudWatch 代理。

### 使用命令行在服务器上启动 CloudWatch 代理

1. 将要使用的代理配置文件复制到要运行代理的服务器。记下将其复制到的路径名。
2. 此命令中，`-a fetch-config` 会使代理加载最新版本的 CloudWatch 代理配置文件，`-s` 则会启动该代理。

输入下列命令之一。将 `configuration-file-path` 替换为指向代理配置文件的路径。如果您使用向导创建此文件，则此文件命名为 `config.json`；如果您手动创建该文件，则该文件可能命名为 `amazon-cloudwatch-agent.json`。

在运行 Linux 的 EC2 实例上，输入以下命令。

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:configuration-file-path
```

在运行 Linux 的本地服务器上，输入以下内容：

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m onPremise -s -c file:configuration-file-path
```

在运行 Windows Server 的 EC2 实例上，从 PowerShell 控制台中输入以下内容：

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m ec2 -s -c file:configuration-file-path
```

在运行 Windows Server 的本地服务器上，从 PowerShell 控制台中输入以下内容：

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m onPremise -s -c file:configuration-file-path
```

## 使用 Amazon Systems Manager 安装 CloudWatch 代理

可以按照以下主题使用 Amazon Systems Manager 安装和运行 CloudWatch 代理。

### 主题

- [创建 IAM 角色和用户以用于 CloudWatch 代理 \(p. 669\)](#)
- [下载和配置 CloudWatch 代理 \(p. 672\)](#)
- [使用代理配置在 EC2 实例上安装 CloudWatch 代理 \(p. 674\)](#)
- [在本地部署服务器上安装 CloudWatch 代理 \(p. 677\)](#)

## 创建 IAM 角色和用户以用于 CloudWatch 代理

要访问 Amazon 资源，需要具有相应的权限。您可以创建包含所需权限的 IAM 角色和用户，以使 CloudWatch 代理将指标写入到 CloudWatch 中，以及使 CloudWatch 代理与 Amazon EC2 和 Amazon Systems Manager 进行通信。您在 Amazon EC2 实例上使用 IAM 角色，在本地部署服务器上使用 IAM 用户。

一个角色或用户能让 CloudWatch 代理安装到服务器上，并将指标发送到 CloudWatch。需要使用另一个角色或用户以在 Systems Manager Parameter Store 中存储 CloudWatch 代理配置。Parameter Store 允许多个服务器使用一个 CloudWatch 代理配置。

写入 Parameter Store 是一项广泛而强大的权限。您应该仅在需要时使用它，并且不应将其附加到部署中的多个实例。如果在 Parameter Store 中存储 CloudWatch 代理配置，我们建议您执行以下操作：

- 设置一个实例以在其中执行该配置。
- 使用仅在该实例上有限写入到 Parameter Store 的 IAM 角色。
- 使用仅在处理并保存 CloudWatch 代理配置文件时，有权限写入到 Parameter Store 的 IAM 角色。

#### Note

我们最近修改了以下过程，具体来说就是使用由 Amazon 创建的新 `CloudWatchAgentServerPolicy` 和 `CloudWatchAgentAdminPolicy` 策略，而不需要客户自行创建这些策略。要使用这些策略将代理配置文件写入 Parameter Store，然后从 Parameter Store 中下载该文件，您的代理配置文件的名称必须以 `AmazonCloudWatch-` 开头。如果您的 CloudWatch 代理配置文件的文件名不以 `AmazonCloudWatch-` 开头，则这些策略不能用于将文件写入 Parameter Store 或从 Parameter Store 下载文件。

## 在 Amazon EC2 实例上创建要与 CloudWatch 代理一起使用的 IAM 角色

第一个过程会创建一个 IAM 角色，您必须将其附加到运行 CloudWatch 代理的每个 Amazon EC2 实例。该角色提供从实例中读取信息并将其写入到 CloudWatch 的权限。

第二个过程会创建一个 IAM 角色，您必须将其附加到用于创建 CloudWatch 代理配置文件的 Amazon EC2 实例。如果要将该文件存储在 Systems Manager Parameter Store 中，以便其他服务器可以使用该文件，则需要执行该步骤。除了从实例中读取信息并将其写入到 CloudWatch 的权限以外，该角色还提供写入到 Parameter Store 的权限。该角色包含足以运行 CloudWatch 代理以及写入到 Parameter Store 的权限。

#### Note

Parameter Store 支持标准层和高级层中的参数。这些参数层与 CloudWatch 代理预定义指标集提供的“基本”、“标准”和“高级”详细信息级别无关。

### 创建每个服务器运行 CloudWatch 代理所需的 IAM 角色

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Roles，然后选择 Create role。
3. 在 Select type of trusted entity (选择受信任实体的类型) 下，选择 Amazon service (Amazon 服务)。
4. 在 Common use cases (常见使用案例) 下，选择 EC2，然后选择 Next: Permissions (下一步：权限)。
5. 在策略列表中，选中 `CloudWatchAgentServerPolicy` 旁边的复选框。如有必要，请使用搜索框查找该策略。
6. 要使用 Systems Manager 安装或配置 CloudWatch 代理，请选中 `AmazonSSMManagedInstanceCore` 旁边的框。此 Amazon 托管策略使实例能够使用 Systems Manager 服务核心功能。如有必要，请使用搜索框查找该策略。如果要仅通过命令行启动和配置代理，则不需要此策略。
7. 选择 下一步：标签。
8. (可选) 添加一个或多个标签键值对，以组织、跟踪或控制此角色的访问，然后选择 Next: Review (下一步：审核)。
9. 对于角色名称，请输入新角色的名称，如 `CloudWatchAgentServerRole` 或所需的其他名称。
10. (可选) 对于角色描述，请输入描述。
11. 确认 `CloudWatchAgentServerPolicy` 和 (可选) `AmazonSSMManagedInstanceCore` 是否显示在 Policies (策略) 旁边。

## 12. 选择 Create role ( 创建角色 )。

将立即创建该角色。

以下过程创建也可以写入到 Parameter Store 的 IAM 角色。您可以使用该角色在 Parameter Store 中存储代理配置文件，以便其他服务器检索该文件。

写入到 Parameter Store 的权限提供了广泛的访问。此角色不应附加到您的所有服务器，只有管理员才能使用它。在创建代理配置文件并将其复制到 Parameter Store 后，应将该角色从实例中分离并改用 CloudWatchAgentServerRole。

为管理员创建 IAM 角色以写入到 Parameter Store

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Roles，然后选择 Create role。
3. 在 Select type of trusted entity (选择受信任实体的类型) 下，选择 Amazon service (Amazon 服务)。
4. 在紧靠选择将使用此角色的服务下面，选择 EC2，然后选择下一步: 权限。
5. 在策略列表中，选中 CloudWatchAgentAdminPolicy 旁边的复选框。如有必要，请使用搜索框查找该策略。
6. 要使用 Systems Manager 安装或配置 CloudWatch 代理，请选中 AmazonSSMManagedInstanceCore 旁边的框。此 Amazon 托管策略使实例能够使用 Systems Manager 服务核心功能。如有必要，请使用搜索框查找该策略。如果要仅通过命令行启动和配置代理，则不需要此策略。
7. 选择 下一步: 标签。
8. ( 可选 ) 添加一个或多个标签键值对，以组织、跟踪或控制此角色的访问，然后选择 Next: Review (下一步: 审核)。
9. 对于角色名称，请输入新角色的名称，如 **CloudWatchAgentAdminRole** 或所需的其他名称。
10. ( 可选 ) 对于角色描述，请输入描述。
11. 确认 CloudWatchAgentAdminPolicy 和 ( 可选 ) AmazonSSMManagedInstanceCore 是否显示在 Policies ( 策略 ) 旁边。
12. 选择 Create role ( 创建角色 )。

将立即创建该角色。

## 创建 IAM 用户以用于本地部署服务器上的 CloudWatch 代理

第一个过程会创建运行 CloudWatch 代理所需的 IAM 用户。该用户提供将数据发送到 CloudWatch 的权限。

第二个过程会创建一个 IAM 用户，您可以在创建 CloudWatch 代理配置文件时使用该用户。可以使用此过程将该文件存储在 Systems Manager Parameter Store 中，以便其他服务器使用该文件。除了将数据写入到 CloudWatch 的权限以外，该用户还提供了写入到 Parameter Store 的权限。

### Note

Parameter Store 支持标准层和高级层中的参数。这些参数层与 CloudWatch 代理预定义指标集提供的“基本”、“标准”和“高级”详细信息级别无关。

创建 CloudWatch 代理将数据写入 CloudWatch 所需的 IAM 用户

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Users，然后选择 Add user。
3. 为新用户输入用户名。
4. 对于访问类型，请选择编程访问，然后选择下一步: 权限。

5. 对于设置权限，请选择直接附加现有策略。
6. 在策略列表中，选中 CloudWatchAgentServerPolicy 旁边的复选框。如有必要，请使用搜索框查找该策略。
7. 要使用 Systems Manager 安装或配置 CloudWatch 代理，请选中 AmazonSSMManagedInstanceCore 旁边的框。此 Amazon 托管策略使实例能够使用 Systems Manager 服务核心功能。（如有必要，请使用搜索框查找策略。如果仅通过命令行启动和配置该代理，则不需要使用该策略。）
8. 选择 下一步：标签。
9. （可选）添加一个或多个标签键值对，以组织、跟踪或控制此角色的访问，然后选择 Next: Review (下一步：审核)。
10. 确认列出了正确的策略，然后选择创建用户。
11. 在新用户的行中，选择显示。将访问密钥和私有密钥复制到一个文件以便在安装该代理时使用。选择 Close (关闭)。

以下过程会创建也可以写入到 Parameter Store 的 IAM 用户。如果要在 Parameter Store 中存储代理配置文件以使其他服务器可以使用该文件，您需要使用该 IAM 用户。该 IAM 用户提供写入到 Parameter Store 的权限。该用户还提供从实例中读取信息以及将其写入到 CloudWatch 的权限。写入到 Systems Manager Parameter Store 的权限提供了广泛的访问。该 IAM 用户不应附加到您的所有服务器，仅管理员应该使用该用户。只有在 Parameter Store 中存储代理配置文件时，才应使用该 IAM 用户。

创建将配置文件存储到 Parameter Store 中以及向 CloudWatch 发送信息所需的 IAM 用户

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Users，然后选择 Add user。
3. 为新用户输入用户名。
4. 对于访问类型，请选择编程访问，然后选择下一步：权限。
5. 对于设置权限，请选择直接附加现有策略。
6. 在策略列表中，选中 CloudWatchAgentAdminPolicy 旁边的复选框。如有必要，请使用搜索框查找该策略。
7. 要使用 Systems Manager 安装或配置 CloudWatch 代理，请选中 AmazonSSMManagedInstanceCore 旁边的复选框。此 Amazon 托管策略使实例能够使用 Systems Manager 服务核心功能。（如有必要，请使用搜索框查找策略。如果仅通过命令行启动和配置该代理，则不需要使用该策略。）
8. 选择 下一步：标签。
9. （可选）添加一个或多个标签键值对，以组织、跟踪或控制此角色的访问，然后选择 Next: Review (下一步：审核)。
10. 确认列出了正确的策略，然后选择创建用户。
11. 在新用户的行中，选择显示。将访问密钥和私有密钥复制到一个文件以便在安装该代理时使用。选择 Close (关闭)。

## 下载和配置 CloudWatch 代理

本部分介绍了如何使用 Systems Manager 下载代理，然后如何创建您的代理配置文件。您必须确保为 Systems Manager 正确配置实例，然后才能使用 Systems Manager 下载代理。

### 安装或更新 SSM Agent

在 Amazon EC2 实例上，CloudWatch 代理要求实例运行 2.2.93.0 或更高版本。在安装 CloudWatch 代理之前，如果还没有在实例上更新或安装 SSM Agent，请执行该操作。

有关在运行 Linux 的实例上安装或更新 SSM Agent 的信息，请参阅 Amazon Systems Manager 用户指南中的在 [Linux 实例上安装和配置 SSM Agent](#)。

有关安装或更新 SSM Agent 的更多信息，请参阅 Amazon Systems Manager 用户指南中的 [使用 SSM Agent](#)。

## ( 可选 ) 验证 Systems Manager 的先决条件

在使用 Systems Manager Run Command 安装和配置 CloudWatch 代理之前，请确保您的实例满足 Systems Manager 的最低要求。有关更多信息，请参阅 Amazon Systems Manager 用户指南中的 [Systems Manager 的先决条件](#)。

## 验证 Internet 访问权限

您的 Amazon EC2 实例必须具有出站互联网访问权限，才能将数据发送到 CloudWatch 或 CloudWatch Logs。有关如何配置互联网访问权限的更多信息，请参阅 Amazon VPC 用户指南中的 [互联网网关](#)。

要在您的代理上配置的终端节点和端口如下所示：

- 如果要使用代理收集指标，则必须允许列出相应区域的 CloudWatch 端点。这些端点在 Amazon Web Services 一般参考中的 [Amazon CloudWatch](#) 中列出。
- 如果要使用代理收集日志，则必须允许列出相应区域的 CloudWatch Logs 端点。这些端点在 Amazon Web Services 一般参考中的 [Amazon CloudWatch Logs](#) 中列出。
- 如果使用 Systems Manager 安装代理或使用 Parameter Store 存储配置文件，您必须允许列出相应区域的 Systems Manager 端点。这些端点在 Amazon Web Services 一般参考中的 [Amazon Systems Manager](#) 中列出。

使用以下步骤来使用 Systems Manager 下载 CloudWatch 代理软件包。

## 使用 Systems Manager 下载 CloudWatch 代理

1. 在 <https://console.aws.amazon.com/systems-manager/> 处打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

–或者–

如果打开了 Amazon Systems Manager 主页，请向下滚动并选择 Explore Run Command ( 浏览 Run Command )。

3. 选择 Run command (运行命令)。
4. 在 Command document ( 命令文档 ) 列表中，选择 AWS-ConfigureAWSPackage。
5. 在 Targets ( 目标 ) 区域中，选择要在其上安装 CloudWatch 代理的实例。如果您没有看到特定实例，则可能未将其配置为可使用 Systems Manager 的托管式实例。有关更多信息，请参阅 Amazon Systems Manager 用户指南中的 [为混合环境设置 Amazon Systems Manager](#)。
6. 在操作列表中，选择安装。
7. 在 Name (名称) 字段中，输入 **AmazonCloudWatchAgent**。
8. 将 Version (版本) 设置为 latest (最新) 以安装最新版本的代理。
9. 选择 Run ( 运行 )。
10. ( 可选 ) 在 Targets and outputs ( 目标和输出 ) 区域中，选择实例名称旁边的按钮，然后选择 View output ( 查看输出 )。Systems Manager 将显示已成功安装该代理。

## 创建和修改代理配置文件

在下载 CloudWatch 代理后，您必须先创建配置文件，然后再在任何服务器上启动该代理。

如果您打算在 Systems Manager Parameter Store 中保存代理配置文件，则必须使用 EC2 实例以保存到 Parameter Store。此外，您必须首先使用 CloudWatchAgentAdminRole IAM 角色附加到该实例。有关附加角色的更多信息，请参阅适用于 Windows 实例的 Amazon EC2 用户指南中的 [将 IAM 角色附加到实例](#)。



有关创建 CloudWatch 代理配置文件的更多信息，请参阅 [创建 CloudWatch 代理配置文件 \(p. 682\)](#)。

## 使用代理配置在 EC2 实例上安装 CloudWatch 代理

在 Parameter Store 中保存 CloudWatch 代理配置后，您可以在其他服务器上安装该代理时使用该配置。

### 主题

- [将 IAM 角色附加到实例 \(p. 674\)](#)
- [在 Amazon EC2 实例上下载 CloudWatch 代理软件包 \(p. 674\)](#)
- ( 可选 ) [修改 CloudWatch 代理的通用配置和命名配置文件 \(p. 675\)](#)
- [启动 CloudWatch 代理 \(p. 676\)](#)

### 将 IAM 角色附加到实例

您必须将 CloudWatchAgentServerRole IAM 角色附加到 EC2 实例，才能在该实例上运行 CloudWatch 代理。该角色允许 CloudWatch 代理在实例上执行操作。

有关更多信息，请参阅适用于 Windows 实例的 Amazon EC2 用户指南中的[将 IAM 角色附加到实例](#)。

### 在 Amazon EC2 实例上下载 CloudWatch 代理软件包

#### Note

当您安装或更新 CloudWatch 代理时，仅支持 Uninstall and reinstall ( 卸载并重新安装 ) 选项。您无法使用 In-place update ( 就地更新 ) 选项。

您可以使用 Systems Manager Run Command 或 Amazon S3 下载链接下载 CloudWatch 代理程序包。有关使用 Amazon S3 下载链接的信息，请参阅 [下载 CloudWatch 代理软件包 \(p. 661\)](#)。

### 使用 Systems Manager 在 Amazon EC2 实例上下载 CloudWatch 代理

您必须确保为 Systems Manager 正确配置实例，然后才能使用 Systems Manager 安装 CloudWatch 代理。

#### 安装或更新 SSM Agent

在 Amazon EC2 实例上，CloudWatch 代理要求实例运行 2.2.93.0 或更高版本。在安装 CloudWatch 代理之前，如果还没有在实例上更新或安装 SSM Agent，请执行该操作。

有关在运行 Linux 的实例上安装或更新 SSM Agent 的信息，请参阅 Amazon Systems Manager 用户指南中的[在 Linux 实例上安装和配置 SSM Agent](#)。

有关在运行 Windows 服务器的实例上安装或更新 SSM Agent 的信息，请参阅 Amazon Systems Manager 用户指南中的[在 Windows 实例上安装和配置 SSM Agent](#)。

#### ( 可选 ) 验证 Systems Manager 的先决条件

在使用 Systems Manager Run Command 安装和配置 CloudWatch 代理之前，请确保您的实例满足 Systems Manager 的最低要求。有关更多信息，请参阅 Amazon Systems Manager 用户指南中的[设置 Amazon Systems Manager](#)。

#### 验证互联网访问权限

您的 Amazon EC2 实例必须具有出站互联网访问权限，才能将数据发送到 CloudWatch 或 CloudWatch Logs。有关如何配置互联网访问权限的更多信息，请参阅 Amazon VPC 用户指南中的[互联网网关](#)。

#### 下载 CloudWatch 代理软件包

Systems Manager Run Command 使您能够管理实例配置。您可以指定一个 Systems Manager 文档，指定一些参数，然后在一个或多个实例上执行命令。实例上的 SSM Agent 负责处理命令并按指定方式配置实例。

## 使用 Run Command 下载 CloudWatch 代理

1. 在 <https://console.aws.amazon.com/systems-manager/> 处打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

–或者–

如果打开了 Amazon Systems Manager 主页，请向下滚动并选择 Explore Run Command (浏览 Run Command)。

3. 选择 Run command (运行命令)。
4. 在 Command document (命令文档) 列表中，选择 AWS-ConfigureAWSPackage。
5. 在 Targets (目标) 区域中，选择要在其上安装 CloudWatch 代理的实例。如果未看到特定的实例，则可能没有为 Run Command 配置该实例。有关更多信息，请参阅 Amazon Systems Manager 用户指南中的[为混合环境设置 Amazon Systems Manager](#)。
6. 在操作列表中，选择安装。
7. 在 Name (名称) 框中，输入 **AmazonCloudWatchAgent**。
8. 将 Version (版本) 设置为 latest (最新) 以安装最新版本的代理。
9. 选择 Run (运行)。
10. (可选) 在 Targets and outputs (目标和输出) 区域中，选择实例名称旁边的按钮，然后选择 View output (查看输出)。Systems Manager 将显示已成功安装该代理。

## (可选) 修改 CloudWatch 代理的通用配置和命名配置文件

CloudWatch 代理包含一个名为 common-config.toml 的配置文件。您可以使用该文件指定代理和区域信息 (可选)。

在运行 Linux 的服务器上，该文件位于 /opt/aws/amazon-cloudwatch-agent/etc 目录中。在运行 Windows Server 的服务器上，该文件位于 C:\ProgramData\Amazon\AmazonCloudWatchAgent 目录中。

默认 common-config.toml 如下所示：

```
This common-config is used to configure items used for both ssm and cloudwatch access

Configuration for shared credential.
Default credential strategy will be used if it is absent here:
Instance role is used for EC2 case by default.
AmazonCloudWatchAgent profile is used for onPremise case by default.
[credentials]
shared_credential_profile = "{profile_name}"
shared_credential_file= "{file_name}"

Configuration for proxy.
System-wide environment-variable will be read if it is absent here.
i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
Note: system-wide environment-variable is not accessible when using ssm run-command.
Absent in both here and environment-variable means no proxy will be used.
[proxy]
http_proxy = "{http_url}"
https_proxy = "{https_url}"
no_proxy = "{domain}"
```

最初将注释所有行。要设置凭证配置文件或代理设置，请从该行中删除 # 并指定一个值。您可以手动编辑该文件，或者使用 Systems Manager 中的 RunShellScript Run Command 执行该操作：

- `shared_credential_profile` – 对于本地部署服务器，此行指定 IAM 用户凭证配置文件，以用于将数据发送到 CloudWatch。如果您将此行注释掉，则会使用 `AmazonCloudWatchAgent`。

在 EC2 实例上，您可以使用此行让 CloudWatch 代理将数据从该实例发送到不同 Amazon 区域中的 CloudWatch。要执行此操作，请指定一个包含 `region` 字段的命名配置文件，该字段指定要发送到的区域的名称。

如果指定 `shared_credential_profile`，您还必须从 `[credentials]` 行开头删除 `#`。

- `shared_credential_file` – 要让代理在位于默认路径以外的路径中的文件中查找凭证，请在此处指定完整的路径和文件名。在 Linux 上，默认路径为 `/root/.aws`；在 Windows Server 上，默认路径为 `C:\Users\Administrator\aws`。

下面的第一个示例显示对 Linux 服务器有效的 `shared_credential_file` 行的语法，第二个示例对 Windows Server 有效。在 Windows Server 上，您必须转义 `\` 字符。

```
shared_credential_file= "/usr/username/credentials"
```

```
shared_credential_file= "C:\\Documents and Settings\\username\\aws\\credentials"
```

如果指定 `shared_credential_file`，您还必须从 `[credentials]` 行开头删除 `#`。

- 代理设置 – 如果您的服务器使用 HTTP 或 HTTPS 代理联系 Amazon 服务，请在 `http_proxy` 和 `https_proxy` 字段中指定这些代理。如果应从代理中排除某些 URL，请在 `no_proxy` 字段中指定这些 URL 并以逗号分隔。

## 启动 CloudWatch 代理

您可以使用 Systems Manager Run Command 或命令行来启动代理。

### 使用 Systems Manager Run Command 启用 CloudWatch 代理

使用 Systems Manager Run Command 执行以下步骤来启动代理。

#### 使用 Run Command 启动 CloudWatch 代理

1. 在 <https://console.aws.amazon.com/systems-manager/> 处打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

–或者–

如果打开了 Amazon Systems Manager 主页，请向下滚动并选择 Explore Run Command (浏览 Run Command)。

3. 选择 Run command (运行命令)。
4. 在命令文档列表中，选择 `AmazonCloudWatch-ManagedAgent`。
5. 在 Targets (目标) 区域中，选择安装了 CloudWatch 代理的实例。
6. 在操作列表中，选择配置。
7. 在可选的配置源列表中，选择 `ssm`。
8. 在 Optional Configuration Location (可选的配置位置) 框中，输入您创建并保存到 Systems Manager Parameter Store 的代理配置文件的名称，如 [创建 CloudWatch 代理配置文件 \(p. 682\)](#) 中所述。
9. 在完成这些步骤后，在可选的重新启动列表中选择是以启动该代理。
10. 选择 Run (运行)。
11. (可选) 在 Targets and outputs (目标和输出) 区域中，选择实例名称旁边的按钮，然后选择 View output (查看输出)。Systems Manager 将显示已成功启动该代理。



## 使用命令行在 Amazon EC2 实例上启动 CloudWatch 代理

可以执行以下步骤以使用命令行在 Amazon EC2 实例上安装 CloudWatch 代理。

### 使用命令行在 Amazon EC2 实例上启动 CloudWatch 代理

- 此命令中，`-a fetch-config` 会使代理加载最新版本的 CloudWatch 代理配置文件，`-s` 则会启动该代理。

Linux 和 macOS：如果在 Systems Manager Parameter Store 中保存了配置文件，请输入以下命令：

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c ssm:configuration-parameter-store-name
```

Linux 和 macOS：如果在本地电脑上保存了配置文件，请输入以下命令。将 `configuration-file-path` 替换为指向代理配置文件的路径。如果您使用向导创建此文件，则此文件命名为 `config.json`；如果您手动创建该文件，则该文件可能命名为 `amazon-cloudwatch-agent.json`。

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:configuration-file-path
```

Windows Server：如果在 Systems Manager Parameter Store 中保存代理配置文件，请从 PowerShell 控制台中输入以下命令：

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m ec2 -s -c ssm:configuration-parameter-store-name
```

Windows Server：如果在本地计算机上保存代理配置文件，请从 PowerShell 控制台中输入以下命令：

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m ec2 -s -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent\config.json"
```

## 在本地部署服务器上安装 CloudWatch 代理

如果已在一台电脑上下载了 CloudWatch 代理并创建了所需的代理配置文件，则可以使用该配置文件在其他本地部署服务器上安装该代理。

### 在本地部署服务器上下载 CloudWatch 代理

您可以使用 Systems Manager Run Command 或 Amazon S3 下载链接下载 CloudWatch 代理程序包。有关使用 Amazon S3 下载链接的信息，请参阅 [下载 CloudWatch 代理软件包 \(p. 661\)](#)。

#### 使用 Systems Manager 下载

要使用 Systems Manager Run Command，您必须通过 Amazon EC2 Systems Manager 注册您的本地部署服务器。有关更多信息，请参阅 Amazon Systems Manager 用户指南中的 [在混合环境中设置 Systems Manager](#)。

如果已注册您的服务器，请将 SSM Agent 更新为最新版本。

有关在运行 Linux 的服务器上更新 SSM Agent 的信息，请参阅 Amazon Systems Manager 用户指南中的 [为混合环境安装 SSM Agent \(Linux\)](#)。

有关在运行 Windows Server 的服务器上更新 SSM Agent 的信息，请参阅 Amazon Systems Manager 用户指南中的 [为混合环境安装 SSM Agent \(Windows\)](#)。

使用 SSM Agent 在本地部署服务器上下载 CloudWatch 代理软件包

1. 在 <https://console.aws.amazon.com/systems-manager/> 处打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

–或者–

如果打开了 Amazon Systems Manager 主页，请向下滚动并选择 Explore Run Command (浏览 Run Command)。

3. 选择 Run command (运行命令)。
4. 在 Command document (命令文档) 列表中，选择 AWS-ConfigureAWSPackage 旁边的按钮。
5. 在 Targets (目标) 区域中，选择要在其上安装 CloudWatch 代理的服务器。如果未看到特定的服务器，则可能没有为 Run Command 配置该服务器。有关更多信息，请参阅 Amazon Systems Manager 用户指南中的[为混合环境设置 Amazon Systems Manager](#)。
6. 在操作列表中，选择安装。
7. 在 Name (名称) 框中，输入 `AmazonCloudWatchAgent`。
8. 将 Version (版本) 保留空白以安装最新版本的代理。
9. 选择 Run (运行)。

将下载代理软件包，后续步骤是配置并启动该代理。

### (安装在本地部署服务器上) 指定 IAM 凭证和 Amazon 区域

要使 CloudWatch 代理能够从本地部署服务器发送数据，您必须指定先前创建的 IAM 用户的访问密钥和私有密钥。有关创建此用户的更多信息，请参阅[创建 IAM 角色和用户以用于 CloudWatch 代理](#) (p. 669)。

您还必须使用 `region` 字段指定要将指标发送到的 Amazon 区域。

以下为该文件的示例。

```
[AmazonCloudWatchAgent]
aws_access_key_id=my_access_key
aws_secret_access_key=my_secret_key
region = us-west-1
```

对于 `my_access_key` 和 `my_secret_key`，请使用来自没有写入到 Systems Manager Parameter Store 权限的 IAM 用户的密钥。有关 CloudWatch 代理所需的 IAM 用户的更多信息，请参阅[创建 IAM 用户以用于本地部署服务器上的 CloudWatch 代理](#) (p. 671)。

如果您将此配置文件命名为 `AmazonCloudWatchAgent`，则无需执行任何操作。(可选) 您可以为其指定一个不同的名称，并将该名称指定为 `common-config.toml` 文件中的 `shared_credential_profile` 值(将在下一节中介绍)。

下面是使用 `aws configure` 命令为 CloudWatch 代理创建命名配置文件的示例。该示例假设您使用默认配置文件名称 `AmazonCloudWatchAgent`。

为 CloudWatch 代理创建 `AmazonCloudWatchAgent` 配置文件

1. 如果您还没有这样做，请在服务器上安装 Amazon Command Line Interface。有关更多信息，请参阅[安装 Amazon CLI](#)。
2. 在 Linux 服务器上，输入以下命令并按照提示进行操作：

```
sudo aws configure --profile AmazonCloudWatchAgent
```

在 Windows Server 上，以管理员身份打开 PowerShell，输入以下命令并按照提示进行操作。

```
aws configure --profile AmazonCloudWatchAgent
```

## ( 可选 ) 修改 CloudWatch 代理的通用配置和命名配置文件

CloudWatch 代理包含一个名为 `common-config.toml` 的配置文件。您可以 ( 可选 ) 使用该文件指定代理和区域信息。

在运行 Linux 的服务器上，该文件位于 `/opt/aws/amazon-cloudwatch-agent/etc` 目录中。在运行 Windows Server 的服务器上，该文件位于 `C:\ProgramData\Amazon\AmazonCloudWatchAgent` 目录中。

默认 `common-config.toml` 如下所示：

```
This common-config is used to configure items used for both ssm and cloudwatch access

Configuration for shared credential.
Default credential strategy will be used if it is absent here:
Instance role is used for EC2 case by default.
AmazonCloudWatchAgent profile is used for onPremise case by default.
[credentials]
shared_credential_profile = "{profile_name}"
shared_credential_file= "{file_name}"

Configuration for proxy.
System-wide environment-variable will be read if it is absent here.
i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
Note: system-wide environment-variable is not accessible when using ssm run-command.
Absent in both here and environment-variable means no proxy will be used.
[proxy]
http_proxy = "{http_url}"
https_proxy = "{https_url}"
no_proxy = "{domain}"
```

最初将注释所有行。要设置凭证配置文件或代理设置，请从该行中删除 `#` 并指定一个值。您可以手动编辑该文件，或者使用 Systems Manager 中的 RunShellScript Run Command 执行该操作：

- `shared_credential_profile` – 对于本地部署服务器，此行指定 IAM 用户凭证配置文件，以用于将数据发送到 CloudWatch。如果您将此行注释掉，则会使用 `AmazonCloudWatchAgent`。有关创建此配置文件的更多信息，请参阅 ( [安装在本地部署服务器上](#) ) [指定 IAM 凭证和 Amazon 区域](#) (p. 678)。

在 EC2 实例上，您可以使用此行让 CloudWatch 代理将数据从该实例发送到不同 Amazon 区域中的 CloudWatch。要执行此操作，请指定一个包含 `region` 字段的命名配置文件，该字段指定要发送到的区域的名称。

如果指定 `shared_credential_profile`，您还必须从 `[credentials]` 行开头删除 `#`。

- `shared_credential_file` – 要让代理在位于默认路径以外的路径中的文件中查找凭证，请在此处指定完整的路径和文件名。在 Linux 上，默认路径为 `/root/.aws`；在 Windows Server 上，默认路径为 `C:\Users\Administrator\aws`。

下面的第一个示例显示对 Linux 服务器有效的 `shared_credential_file` 行的语法，第二个示例对 Windows Server 有效。在 Windows Server 上，您必须转义 `\` 字符。

```
shared_credential_file= "/usr/username/credentials"
```

```
shared_credential_file= "C:\\Documents and Settings\\username\\aws\\credentials"
```

如果指定 `shared_credential_file`，您还必须从 `[credentials]` 行开头删除 `#`。

- 代理设置 – 如果您的服务器使用 HTTP 或 HTTPS 代理联系 Amazon 服务，请在 `http_proxy` 和 `https_proxy` 字段中指定这些代理。如果应从代理中排除某些 URL，请在 `no_proxy` 字段中指定这些 URL 并以逗号分隔。

## 启动 CloudWatch 代理

您可以使用 Systems Manager Run Command 或命令行启动 CloudWatch 代理。

使用 SSM Agent 在本地部署服务器上启动 CloudWatch 代理

1. 在 <https://console.aws.amazon.com/systems-manager/> 处打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

–或者–

如果打开了 Amazon Systems Manager 主页，请向下滚动并选择 Explore Run Command (浏览 Run Command)。

3. 选择 Run command (运行命令)。
4. 在命令文档列表中，选择 AmazonCloudWatch-ManagedAgent 旁边的按钮。
5. 在目标区域，选择安装了该代理的实例。
6. 在操作列表中，选择配置。
7. 在模式列表中，选择 onPremise。
8. 在 Optional Configuration Location (可选的配置位置) 框中，输入您使用向导创建并存储在 Parameter Store 中的代理配置文件的名称。
9. 选择 Run (运行)。

该代理使用在配置文件中指定的配置启动。

使用命令行在本地部署服务器上启动 CloudWatch 代理

- 此命令中，`-a fetch-config` 会使代理加载最新版本的 CloudWatch 代理配置文件，`-s` 则会启动该代理。

Linux：如果在 Systems Manager Parameter Store 中保存了配置文件，请输入以下命令：

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m onPremise -s -c ssm:configuration-parameter-store-name
```

Linux：如果在本地计算机上保存了配置文件，请输入以下命令。将 `configuration-file-path` 替换为指向代理配置文件的路径。如果您使用向导创建此文件，则此文件命名为 `config.json`；如果您手动创建该文件，则该文件可能命名为 `amazon-cloudwatch-agent.json`。

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m onPremise -s -c file:configuration-file-path
```

Windows Server：如果在 Systems Manager Parameter Store 中保存代理配置文件，请从 PowerShell 控制台中输入以下命令：

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m onPremise -s -c ssm:configuration-parameter-store-name
```

Windows Server：如果在本地计算机上保存了代理配置文件，请从 PowerShell 控制台中输入以下命令。将 `configuration-file-path` 替换为指向代理配置文件的路径。如果您使用向导创建此文件，则此文件命名为 `config.json`；如果您手动创建该文件，则该文件可能命名为 `amazon-cloudwatch-agent.json`。

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a
fetch-config -m onPremise -s -c file:configuration-file-path
```

## 验证 CloudWatch 代理软件包的签名

包含 GPG 签名文件以用于 Linux 服务器上的 CloudWatch 代理软件包。您可以使用公有密钥验证该代理下载文件是否为原始的而未进行修改。

对于 Windows Server，您可以使用 MSI 验证签名。

对于 macOS 电脑，代理下载软件包中包含签名。

要查找正确的签名文件，请参阅下表。对于每个架构和操作系统，有一个常规链接以及对应于每个区域的链接。例如，对于 Amazon Linux 和 Amazon Linux 2 以及 x86-64 架构，三个有效的链接如下：

- [https://s3.amazonaws.com/amazoncloudwatch-agent/amazon\\_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig](https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig)
- [https://s3.us-east-1.amazonaws.com/amazoncloudwatch-agent-us-east-1/amazon\\_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig](https://s3.us-east-1.amazonaws.com/amazoncloudwatch-agent-us-east-1/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig)
- [https://s3.eu-central-1.amazonaws.com/amazoncloudwatch-agent-eu-central-1/amazon\\_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig](https://s3.eu-central-1.amazonaws.com/amazoncloudwatch-agent-eu-central-1/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig)

### 验证 Linux 服务器上的 CloudWatch 代理软件包

1. 下载公有密钥。
2. 将公有密钥导入到您的密钥环中。

```
shell$ gpg --import amazon-cloudwatch-agent.gpg
gpg: key 3B789C72: public key "Amazon CloudWatch Agent" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

请记住密钥值，因为需要在下一步中使用该值。在上一示例中，密钥值为 3B789C72。

3. 通过运行以下命令，将 `key-value` 替换为上一步中的值来验证指纹：

```
shell$ gpg --fingerprint key-value
pub 2048R/3B789C72 2017-11-14
 Key fingerprint = 9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72
uid Amazon CloudWatch Agent
```

指纹字符串应等于以下内容：

9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72

如果指纹字符串不匹配，请不要安装该代理。请联系 Amazon Web Services。

在验证指纹后，您可以使用该指纹验证 CloudWatch 代理软件包的签名。

4. 使用 `wget` 下载软件包签名文件。要确定正确的签名文件，请参阅前一个表。

```
wget Signature File Link
```

5. 要验证签名，请运行 `gpg --verify`。

```
shell$ gpg --verify signature-filename agent-download-filename
gpg: Signature made Wed 29 Nov 2017 03:00:59 PM PST using RSA key ID 3B789C72
gpg: Good signature from "Amazon CloudWatch Agent"
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72
```

如果输出包含短语 `BAD signature`，则检查是否正确执行了此过程。如果您继续获得该响应，请与 Amazon Web Services 联系，并避免使用已下载的文件。

请注意有关信任的警告。只有当您或您信任的某个人对密钥进行了签名，密钥才是可信的。这并不意味着签名无效，只是您尚未验证公有密钥而已。

## 创建 CloudWatch 代理配置文件

在任何服务器上运行 CloudWatch 代理之前，您都必须创建 CloudWatch 代理配置文件。

代理配置文件是一个 JSON 文件，它指定了该代理要收集的指标和日志，包括自定义指标。您可以使用向导创建该文件，或者从 Scratch 自行创建该文件。您还可以最初使用向导创建配置文件，然后手动对其进行修改。如果手动创建或修改该文件，该过程更复杂一些，但您可以更好地控制收集的指标，并且可以指定不通过向导提供的指标。

每次更改代理配置文件时，您必须重新启动该代理以使更改生效。要重新启动该代理，请按照[启动 CloudWatch 代理 \(p. 676\)](#)中的说明进行操作。

在创建配置文件后，可以将其手动保存为 JSON 文件，然后在服务器上安装代理时使用该文件。或者，如果您打算在服务器上安装代理时使用 Systems Manager，则可以将该文件存储在 Systems Manager Parameter Store 中。

### 目录

- [使用向导创建 CloudWatch 代理配置文件 \(p. 682\)](#)
- [手动创建或编辑 CloudWatch 代理配置文件 \(p. 686\)](#)

## 使用向导创建 CloudWatch 代理配置文件

代理配置文件向导 `amazon-cloudwatch-agent-config-wizard` 会提出一系列问题，包括下列内容：

- 是在 Amazon EC2 实例上还是在本地部署服务器上安装该代理？
- 服务器是运行 Linux 还是 Windows Server？
- 是否希望该代理将日志文件也发送到 CloudWatch Logs？如果是，您目前是否具有 CloudWatch Logs 代理配置文件？如果是，CloudWatch 代理可以使用此文件来确定要从服务器收集的日志。
- 如果代理将日志文件发送到 CloudWatch Logs，您希望这些日志文件的保留期限是多长时间？-1 的默认值会将日志项目设置为永不过期。
- 如果要从服务器收集指标，那么是要监控一组默认指标之一，还是要自定义收集的指标列表？
- 您是否要使用 StatsD 或 collectd 从应用程序或服务中收集自定义指标？
- 是否要从现有的 SSM Agent 中迁移？



如果在启动向导之前已具有 Amazon 凭证和配置文件，向导可以自动检测要使用的凭证和 Amazon 区域。有关这些文件的更多信息，请参阅 Amazon Systems Manager 用户指南中的[配置和凭证文件](#)。

在 Amazon 凭证文件中，向导会检查默认凭证，还会查找 AmazonCloudWatchAgent 部分，如下所示：

```
[AmazonCloudWatchAgent]
aws_access_key_id = my_access_key
aws_secret_access_key = my_secret_key
```

向导将显示默认凭证、来自 AmazonCloudWatchAgent 的凭证以及 Others 选项。您可以选择要使用的凭证。如果您选择 Others，则可以输入凭证。

在 Amazon 配置文件中，您可以指定代理将指标发送到的区域（如果与 [default] 部分不同）。默认设置为将指标发布到 Amazon EC2 实例所在的区域。如果要將指标发布到不同的区域，请在此处指定该区域。在以下示例中，指标将发布到 us-west-1 区域。

```
[AmazonCloudWatchAgent]
region = us-west-1
```

## CloudWatch 代理预定义指标集

向导配置了具有不同详细信息级别的预定义指标集。下表显示了这些指标集。有关这些指标的更多信息，请参阅[CloudWatch 代理收集的指标 \(p. 729\)](#)。

### Note

Parameter Store 支持标准层和高级层中的参数。这些参数层与这些表中描述的“基本”、“标准”和“高级”指标详细信息级别无关。

运行 Linux 的 Amazon EC2 实例

| 详细信息级别 | 包含的指标                                                                                                                                                                                                                                                                                                 |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 基本     | Mem : mem_used_percent<br><br>Disk : disk_used_percent<br><br>disk 指标（如 disk_used_percent）具有一个用于 Partition 的维度，这意味着生成的自定义指标的数量取决于与您的实例关联的分区数量。您拥有的磁盘分区数取决于您使用的 AMI 以及您附加到服务器的 Amazon EBS 卷数。                                                                                                          |
| 标准     | CPU : cpu_usage_idle、cpu_usage_iowait、cpu_usage_user、cpu_usage_system<br><br>Disk : disk_used_percent、disk_inodes_free<br><br>Diskio : diskio_io_time<br><br>Mem : mem_used_percent<br><br>Swap : swap_used_percent                                                                                   |
| 高级     | CPU : cpu_usage_idle、cpu_usage_iowait、cpu_usage_user、cpu_usage_system<br><br>Disk : disk_used_percent、disk_inodes_free<br><br>Diskio : diskio_io_time、diskio_write_bytes、diskio_read_bytes、diskio_writes<br><br>Mem : mem_used_percent<br><br>Netstat : netstat_tcp_established、netstat_tcp_time_wait |

| 详细信息级别 | 包含的指标                    |
|--------|--------------------------|
|        | Swap : swap_used_percent |

#### 运行 Linux 的本地服务器

| 详细信息级别 | 包含的指标                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 基本     | Disk : disk_used_percent<br><br>Diskio : diskio_write_bytes、diskio_read_bytes、diskio_writes、diskio_reads<br><br>Mem : mem_used_percent<br><br>Net : net_bytes_sent、net_bytes_recv、net_packets_sent、net_packets_recv<br><br>Swap : swap_used_percent                                                                                                                                                                                    |
| 标准     | CPU : cpu_usage_idle、cpu_usage_iowait<br><br>Disk : disk_used_percent、disk_inodes_free<br><br>Diskio : diskio_io_time、diskio_write_bytes、diskio_read_bytes、diskio_writes<br><br>Mem : mem_used_percent<br><br>Net : net_bytes_sent、net_bytes_recv、net_packets_sent、net_packets_recv<br><br>Swap : swap_used_percent                                                                                                                    |
| 高级     | CPU :<br>cpu_usage_guest、cpu_usage_idle、cpu_usage_iowait、cpu_usage_steal、cpu_usage_system<br><br>Disk : disk_used_percent、disk_inodes_free<br><br>Diskio : diskio_io_time、diskio_write_bytes、diskio_read_bytes、diskio_writes<br><br>Mem : mem_used_percent<br><br>Net : net_bytes_sent、net_bytes_recv、net_packets_sent、net_packets_recv<br><br>Netstat : netstat_tcp_established、netstat_tcp_time_wait<br><br>Swap : swap_used_percent |

#### 运行 Windows Server 的 Amazon EC2 实例

| 详细信息级别 | 包含的指标                                                                                  |
|--------|----------------------------------------------------------------------------------------|
| 基本     | Memory : Memory % Committed Bytes In Use<br><br>LogicalDisk : LogicalDisk % Free Space |
| 标准     | Memory : Memory % Committed Bytes In Use<br><br>Paging : Paging File % Usage           |



| 详细信息级别 | 包含的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | <p>Processor : Processor % Idle Time、Processor % Interrupt Time、Processor % User Time</p> <p>PhysicalDisk : PhysicalDisk % Disk Time</p> <p>LogicalDisk : LogicalDisk % Free Space</p>                                                                                                                                                                                                                                                                                          |
| 高级     | <p>Memory : Memory % Committed Bytes In Use</p> <p>Paging : Paging File % Usage</p> <p>Processor : Processor % Idle Time、Processor % Interrupt Time、Processor % User Time</p> <p>LogicalDisk : LogicalDisk % Free Space</p> <p>PhysicalDisk : PhysicalDisk % Disk Time、PhysicalDisk Disk Write Bytes/sec、PhysicalDisk Disk Read Bytes/sec、PhysicalDisk Disk Writes/sec、PhysicalDisk Disk Reads/sec</p> <p>TCP : TCPv4 Connections Established、TCPv6 Connections Established</p> |

#### 运行 Windows Server 的本地服务器

| 详细信息级别 | 包含的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 基本     | <p>Paging : Paging File % Usage</p> <p>Processor : Processor % Processor Time</p> <p>LogicalDisk : LogicalDisk % Free Space</p> <p>PhysicalDisk : 、PhysicalDisk Disk Write Bytes/sec、PhysicalDisk Disk Read Bytes/sec、PhysicalDisk Disk Writes/sec、PhysicalDisk Disk Reads/sec</p> <p>Memory : Memory % Committed Bytes In Use</p> <p>Network Interface : Network Interface Bytes Sent/sec、Network Interface Bytes Received/sec、Network Interface Packets Sent/sec、Network Interface Packets Received/sec</p> |
| 标准     | <p>Paging : Paging File % Usage</p> <p>Processor : Processor % Processor Time、Processor % Idle Time、Processor % Interrupt Time</p> <p>LogicalDisk : LogicalDisk % Free Space</p> <p>PhysicalDisk : PhysicalDisk % Disk Time、PhysicalDisk Disk Write Bytes/sec、PhysicalDisk Disk Read Bytes/sec、PhysicalDisk Disk Writes/sec、PhysicalDisk Disk Reads/sec</p> <p>Memory : Memory % Committed Bytes In Use</p>                                                                                                   |

| 详细信息级别 | 包含的指标                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | Network Interface : Network Interface Bytes Sent/sec、Network Interface Bytes Received/sec、Network Interface Packets Sent/sec、Network Interface Packets Received/sec                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 高级     | Paging : Paging File % Usage<br><br>Processor : Processor % Processor Time、Processor % Idle Time、Processor % Interrupt Time、Processor % User Time<br><br>LogicalDisk : LogicalDisk % Free Space<br><br>PhysicalDisk : PhysicalDisk % Disk Time、PhysicalDisk Disk Write Bytes/sec、PhysicalDisk Disk Read Bytes/sec、PhysicalDisk Disk Writes/sec、PhysicalDisk Disk Reads/sec<br><br>Memory : Memory % Committed Bytes In Use<br><br>Network Interface : Network Interface Bytes Sent/sec、Network Interface Bytes Received/sec、Network Interface Packets Sent/sec、Network Interface Packets Received/sec<br><br>TCP : TCPv4 Connections Established、TCPv6 Connections Established |

## 运行 CloudWatch 代理配置向导

### 创建 CloudWatch 代理配置文件

1. 输入以下命令以启动 CloudWatch 代理配置向导：

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard
```

在运行 Windows Server 的服务器上，输入以下命令：

```
cd "C:\Program Files\Amazon\AmazonCloudWatchAgent"
amazon-cloudwatch-agent-config-wizard.exe
```

2. 回答这些问题以自定义您的服务器的配置文件。
3. 如果您要在本地存储配置文件，则配置文件 config.json 将存储在 Linux 服务器上的 /opt/aws/amazon-cloudwatch-agent/bin/ 中以及 Windows 服务器上的 C:\Program Files\Amazon\AmazonCloudWatchAgent 中。然后，您可以将该文件复制到要安装代理的其他服务器上。

如果要使用 Systems Manager 安装和配置该代理，请务必在提示是否将该文件存储在 Systems Manager Parameter Store 时回答 Yes（是）。您也可以选择将该文件存储在 Parameter Store 中，即使未使用 SSM Agent 安装 CloudWatch 代理。要能够将该文件存储在 Parameter Store 中，您必须使用具有足够权限的 IAM 角色。有关更多信息，请参阅[创建 IAM 角色和用户以用于 CloudWatch 代理](#) (p. 669)。

## 手动创建或编辑 CloudWatch 代理配置文件

CloudWatch 代理配置文件是一个 JSON 文件，其中包含三个部分：agent、metrics 和 logs。

- agent 部分包含总体代理配置的字段。如果您使用向导，则不会创建 agent 部分。

- `metrics` 部分指定要收集并发布到 CloudWatch 的自定义指标。如果只将代理用于收集日志，则可以忽略文件中的 `metrics` 部分。
- `logs` 部分指定要将哪些日志文件发布到 CloudWatch Logs。如果服务器运行 Windows Server，这可能包括 Windows 事件日志中的事件。

以下几节介绍了该 JSON 文件的结构和字段。您还可以查看该配置文件的架构定义。在 Linux 服务器上，架构定义位于 [installation-directory/doc/amazon-cloudwatch-agent-schema.json](#) 中；在运行 Windows Server 的服务器上，架构定义位于 [installation-directory/amazon-cloudwatch-agent-schema.json](#) 中。

如果手动创建或编辑 代理配置文件，则可以给它指定任何名称。为了简化故障排除，我们建议您在 Linux 服务器上将其命名为 `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`，在运行 Windows Server 的服务器上将其命名为 `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json`。创建该文件后，您可以将其复制到要安装代理的其他服务器上。

## CloudWatch 代理配置文件：Agent 部分

`agent` 部分包含以下字段。向导不会创建 `agent` 部分。相反，向导忽略该部分，并在该部分的所有字段中使用默认值。

- `metrics_collection_interval` – 可选。指定该配置文件中指定的所有指标的收集频率。您可以为特定类型的指标覆盖该值。

该值是以秒为单位指定的。例如，指定 10 将导致每 10 秒收集一次指标；将其设置为 300，将指定每 5 分钟收集一次指标。

如果您将该值设置为低于 60 秒，则将每个指标作为高精度指标进行收集。有关高精度指标的更多信息，请参阅 [高精度指标 \(p. 147\)](#)。

默认值是 60。

- `region` – 指定在监控 Amazon EC2 实例时用于 CloudWatch 端点的区域。收集的指标将发送到该区域，例如，`us-west-1`。如果省略该字段，则该代理将指标发送到 Amazon EC2 实例所在的区域。

如果监控本地服务器，则不使用该字段，该代理从 Amazon 配置文件的 `AmazonCloudWatchAgent` 配置文件中读取区域。

- `credentials` – 指定一个在向不同 Amazon 账户发送指标和日志时使用的 IAM 角色。如果指定，则此字段包含一个参数 `role_arn`。
  - `role_arn` – 指定一个在向不同 Amazon 账户发送指标和日志时用于身份验证的 IAM 角色的 Amazon Resource Name (ARN)。有关更多信息，请参阅 [向不同账户发送指标和日志 \(p. 746\)](#)。
- `debug` – 可选。指定使用调试日志消息运行 CloudWatch 代理。默认值为 `false`。
- `aws_sdk_log_level` – 可选。仅在 CloudWatch 代理版本 1.247350.0 及更高版本中受支持。

您可以将此字段指定为以让代理为 Amazon SDK 终端节点执行日志记录。此字段的值可以包括以下一个或多个选项。使用 `|` 字符分隔多个选项。

- `LogDebug`
- `LogDebugWithSigning`
- `LogDebugWithHTTPBody`
- `LogDebugRequestRetries`
- `LogDebugWithEventStreamBody`

有关这些选项的更多信息，请参阅 [LogLevelType](#)。

- `logfile` – 指定 CloudWatch 代理将日志消息写入到的位置。如果指定空字符串，则将日志传输到 `stderr`。如果未指定该选项，则默认位置如下所示：

- Linux : /opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log
- Windows Server : c:\ProgramData\Amazon\CloudWatchAgent\Logs\amazon-cloudwatch-agent.log

CloudWatch 代理自动轮换它创建的日志文件。在日志文件大小达到 100 MB 时，将会轮换掉该文件。代理将轮换的日志文件最多保留七天，并最多保留五个轮换掉的备份日志文件。备份日志文件的文件名后附有时间戳。时间戳显示轮换掉文件的日期和时间，例如，amazon-cloudwatch-agent-2018-06-08T21-01-50.247.log.gz。

- omit\_hostname – 可选。默认情况下，主机名将作为代理收集的指标维度进行发布（除非您使用 metrics 部分的 append\_dimensions 字段）。将 omit\_hostname 设置为 true，以禁止将主机名作为维度进行发布（即使在您没有使用 append\_dimensions 的情况下）。默认值为 false。
- run\_as\_user – 可选。指定用于运行 CloudWatch 代理的用户。如果未指定该参数，则使用 root 用户。该选项仅在 Linux 服务器上有效。

如果指定该选项，在启动 CloudWatch 代理之前，用户必须存在。有关更多信息，请参阅[以不同用户身份运行 CloudWatch 代理 \(p. 742\)](#)。

- user\_agent – 可选。指定 user-agent 字符串，该字符串由 CloudWatch 代理在 CloudWatch 后端进行 API 调用时使用。默认值是由代理版本、用于编译代理的 Go 编程语言版本、运行时操作系统和架构、构建时间以及启用的插件组成的字符串。

下面是 agent 部分的示例。

```
"agent": {
 "metrics_collection_interval": 60,
 "region": "us-west-1",
 "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log",
 "debug": false,
 "run_as_user": "cwagent"
}
```

## CloudWatch 代理配置文件：Metrics（指标）部分

在运行 Linux 或 Windows Server 的服务器上，metrics 部分包含以下字段：

- namespace – 可选。用于该代理收集的指标的命名空间。默认值为 CWAgent。最大长度为 255 个字符。以下是示例：

```
{
 "metrics": {
 "namespace": "Development/Product1Metrics",

 },
}
```

- append\_dimensions – 可选。将 Amazon EC2 指标维度添加到该代理收集的所有指标中。这也会导致代理不会将主机名作为维度发布。

以下列表中显示了受 append\_dimensions 支持的键值对。任何其他密钥值对都将被忽略。

- "ImageID": "\${aws:ImageId}" 将实例的 AMI ID 设置为 ImageID 维度的值。
- "InstanceId": "\${aws:InstanceId}" 将实例的实例 ID 设置为 InstanceID 维度的值。
- "InstanceType": "\${aws:InstanceType}" 将实例的实例类型设置为 InstanceType 维度的值。
- "AutoScalingGroupName": "\${aws:AutoScalingGroupName}" 将实例的 Auto Scaling 组名称设置为 AutoScalingGroupName 维度的值。

如果您想将维度附加到具有任意键值对的指标，请在该字段中为该特定类型的指标使用 append\_dimensions 参数。

如果您指定一个值（该值取决于 Amazon EC2 元数据），并且使用代理，则必须确保服务器能够访问 Amazon EC2 的端点。有关这些端点的更多信息，请参阅 Amazon Web Services 一般参考中的 [Amazon Elastic Compute Cloud \(Amazon EC2\)](#)。

- `aggregation_dimensions` – 可选。指定所收集的指标要在其上聚合的维度。例如，如果累积 `AutoScalingGroupName` 维度上的指标，则会汇总每个 Auto Scaling 组的所有实例中的指标，并将这些指标作为一个整体进行查看。

您可以累积一个或多个维度的指标。例如，指定 `[["InstanceId"], ["InstanceType"]]`，`[["InstanceId", "InstanceType"]]` 将单独汇总实例 ID 的指标，单独汇总实例类型的指标以及汇总两个维度组合的指标。

您也可以指定 `[]` 以将所有指标累积到一个集合中，而不考虑任何维度。

- `endpoint_override` – 指定一个 FIPS 终端节点或私有链接，以用作代理在其中发送指标的端点。如果指定此项并设置私有链接，您可以将指标发送到 Amazon VPC 终端节点。有关更多信息，请参阅 [Amazon VPC 是什么？](#)。

`endpoint_override` 的值必须是表示 URL 的字符串。

例如，在配置文件的指标部分中，以下内容将代理设置为在发送指标时使用 VPC 终端节点。

```
{
 "metrics": {
 "endpoint_override": "vpce-XXXXXXXXXXXXXXXXXXXXX.monitoring.us-
east-1.vpce.amazonaws.com",

 },
}
```

- `metrics_collected` – 必需。指定要收集的指标，包括通过 `StatsD` 或 `collectd` 收集的自定义指标。该部分包含几个小节。

`metrics_collected` 部分的内容取决于该配置文件适用于运行 Linux 还是 Windows Server 的服务器。

- `force_flush_interval` – 以秒为单位指定指标在发送到服务器之前保留在内存缓冲区中的最大时间量。无论此处的设置如何，如果缓冲区中的指标大小达到 40 KB 或 20 个不同的指标，指标会立即发送到服务器。

默认值是 60。

- `credentials` – 指定一个在向不同账户发送指标时使用的 IAM 角色。如果指定，则此字段包含一个参数 `role_arn`。
- `role_arn` – 指定一个在向不同账户发送指标时用于身份验证的 IAM 角色的 ARN。有关更多信息，请参阅 [向不同账户发送指标和日志 \(p. 746\)](#)。如果在此指定，该值将覆盖在配置文件的 `agent` 部分指定的 `role_arn`（如果有）。

## Linux 部分

在运行 Linux 的服务器上，配置文件的 `metrics_collected` 部分还可能包含以下字段。

其中的很多字段可能包含 `measurement` 部分，其中列出您要为该资源收集的指标。这些 `measurement` 部分可能指定了完整的指标名称（如 `swap_used`），也可能仅指定将附加到资源类型的指标名称部分。例如，如果在 `diskio` 部分的 `measurement` 部分中指定 `reads`，则会导致收集 `diskio_reads` 指标。

- `collectd` – 可选。指定要使用 `collectd` 协议检索自定义指标。您可以使用 `collectd` 软件将指标发送到 CloudWatch 代理。有关可用于 `collectd` 的配置选项的更多信息，请参阅 [使用 collectd 检索自定义指标 \(p. 718\)](#)。

- `ethtool` – 可选。指定要使用 `ethtool` 插件检索网络指标。此插件可以导入标准 `ethtool` 实用工具收集的指标，以及 Amazon EC2 实例中的网络性能指标。有关可用于 `ethtool` 的配置选项的更多信息，请参阅 [收集网络性能指标 \(p. 707\)](#)。
- `cpu` – 可选。指定要收集的 CPU 指标。该部分仅适用于 Linux 实例。您必须对任何要收集的 CPU 指标至少包含 `resources` 和 `totalcpu` 字段之一。此部分包含以下字段：
  - `resources` – 可选。将此字段的值指定为 `*` 会导致收集每个 CPU 的指标。唯一允许的值为 `*`。
  - `totalcpu` – 可选。指定是否报告所有 CPU 内核的汇总 CPU 指标。默认值为 `true`。
  - `measurement` – 指定一组要收集的 CPU 指标。可能的值为 `time_active`、`time_guest`、`time_guest_nice`、`time_idle`、`time_iowait`、`time_irq`、`time_nice`、`time_system` 和 `usage_user`。如果包含 `cpu`，则该字段是必需的。

默认情况下，`cpu_usage_*` 指标的单位是 `Percent`，`cpu_time_*` 指标没有单位。

在每个指标的条目中，您可以选择指定下面的一个或两个字段：

- `rename` – 为该指标指定不同的名称。
- `unit` – 指定用于该指标的单位，从而覆盖该指标的 `None` 默认单位 `None`。您指定的单位必须是有效的 CloudWatch 指标单位，如 [MetricDatum](#) 的 `Unit` 描述中所示。
- `metrics_collection_interval` – 可选。指定收集 CPU 指标的频率，从而覆盖配置文件的 `agent` 部分中指定的全局 `metrics_collection_interval`。

该值是以秒为单位指定的。例如，指定 10 将导致每 10 秒收集一次指标；将其设置为 300，将指定每 5 分钟收集一次指标。

如果您将该值设置为低于 60 秒，则将每个指标作为高精度指标进行收集。有关高精度指标的更多信息，请参阅 [高精度指标 \(p. 147\)](#)。

- `append_dimensions` – 可选。仅用于 CPU 指标的其他维度。如果指定该字段，除了用于该代理收集的所有类型的指标的全局 `append_dimensions` 字段中指定的维度以外，还会使用该字段中的维度。
- `disk` – 可选。指定要收集的磁盘指标。该部分仅适用于 Linux 实例。该部分最多可以包含两个字段：
  - `resources` – 可选。指定一组磁盘装载点。该字段将 CloudWatch 限制为仅从列出的挂载点中收集指标。您可以指定 `*` 值以从所有装载点中收集指标。默认值为从所有装载点中收集指标。
  - `measurement` – 指定一组要收集的磁盘指标。可能的值为 `free`、`total`、`used`、`used_percent`、`inodes_free`、`inodes_used` 和 `inodes_total`。如果包含 `disk`，则该字段是必需的。

#### Note

`disk` 指标具有一个用于 `Partition` 的维度，这意味着生成的自定义指标的数量取决于与您的实例关联的分区数量。您拥有的磁盘分区数取决于您使用的 AMI 以及您附加到服务器的 Amazon EBS 卷数。

要查看每个 `disk` 指标的默认单位，请参阅 [Linux 和 macOS 实例上的 CloudWatch 代理收集的指标 \(p. 729\)](#)。

在每个指标的条目中，您可以选择指定下面的一个或两个字段：

- `rename` – 为该指标指定不同的名称。
- `unit` – 指定用于该指标的单位，从而覆盖该指标的 `None` 默认单位 `None`。您指定的单位必须是有效的 CloudWatch 指标单位，如 [MetricDatum](#) 的 `Unit` 描述中所示。
- `ignore_file_system_types` – 指定在收集磁盘指标时要排除的文件系统类型。有效值包括 `sysfs`、`devtmpfs` 等等。
- `drop_device` – 将其设置为 `true` 会导致 `Device` 不会作为磁盘指标的维度被包括在内。

对于使用 Nitro 系统的实例来说，防止 `Device` 被用作维度非常有用，因为在这些实例上，当实例重新启动时，每个磁盘装载的设备名称都会发生变化。这可能会导致指标中的数据不一致，并导致基于这些指标的警报变成 `INSUFFICIENT DATA` 状态。



默认为 `false`。

- `metrics_collection_interval` – 可选。指定收集磁盘指标的频率，从而覆盖配置文件的 `agent` 部分中指定的全局 `metrics_collection_interval`。

该值是以秒为单位指定的。

如果您将该值设置为低于 60 秒，则将每个指标作为高精度指标进行收集。有关更多信息，请参阅[高精度指标 \(p. 147\)](#)。

- `append_dimensions` – 可选。仅用于磁盘指标的其他维度。如果指定该字段，除了用于该代理收集的所有类型的指标的 `append_dimensions` 字段中指定的维度以外，还会使用该字段中的维度。
- `diskio` – 可选。指定要收集的磁盘 I/O 指标。该部分仅适用于 Linux 实例。该部分最多可以包含两个字段：
  - `resources` – 可选。如果指定一组设备，CloudWatch 仅从这些设备中收集指标。否则，将收集所有设备的指标。也可以指定 `*` 值以从所有设备中收集指标。
  - `measurement` – 指定一组要收集的 `diskio` 指标。可能的值为 `reads`、`writes`、`read_bytes`、`write_bytes`、`read_time`、`write_time`、`io_time` 和 `iops_in_progress`。如果包含 `diskio`，则该字段是必需的。

在每个指标的条目中，您可以选择指定下面的一个或两个字段：

- `rename` – 为该指标指定不同的名称。
- `unit` – 指定用于该指标的单位，从而覆盖该指标的 `None` 默认单位 `None`。您指定的单位必须是有效的 CloudWatch 指标单位，如 [MetricDatum](#) 的 `Unit` 描述中所示。
- `metrics_collection_interval` – 可选。指定收集磁盘 I/O 指标的频率，从而覆盖配置文件的 `agent` 部分中指定的全局 `metrics_collection_interval`。

该值是以秒为单位指定的。

如果您将该值设置为低于 60 秒，则将每个指标作为高精度指标进行收集。有关高精度指标的更多信息，请参阅[高精度指标 \(p. 147\)](#)。

- `append_dimensions` – 可选。仅用于磁盘 I/O 指标的其他维度。如果指定该字段，除了用于该代理收集的所有类型的指标的 `append_dimensions` 字段中指定的维度以外，还会使用该字段中的维度。
- `swap` – 可选。指定要收集的交换内存指标。该部分仅适用于 Linux 实例。该部分最多可以包含三个字段：
  - `measurement` – 指定一组要收集的交换内存指标。可能的值为 `free`、`used` 和 `used_percent`。如果包含 `swap`，则该字段是必需的。

要查看每个 `swap` 指标的默认单位，请参阅[Linux 和 macOS 实例上的 CloudWatch 代理收集的指标 \(p. 729\)](#)。

在每个指标的条目中，您可以选择指定下面的一个或两个字段：

- `rename` – 为该指标指定不同的名称。
- `unit` – 指定用于该指标的单位，从而覆盖该指标的 `None` 默认单位 `None`。您指定的单位必须是有效的 CloudWatch 指标单位，如 [MetricDatum](#) 的 `Unit` 描述中所示。
- `metrics_collection_interval` – 可选。指定收集交换内存指标的频率，从而覆盖配置文件的 `agent` 部分中指定的全局 `metrics_collection_interval`。

该值是以秒为单位指定的。

如果您将该值设置为低于 60 秒，则将每个指标作为高精度指标进行收集。有关高精度指标的更多信息，请参阅[高精度指标 \(p. 147\)](#)。

- `append_dimensions` – 可选。仅用于交换内存指标的其他维度。如果指定该字段，除了用于该代理收集的所有类型的指标的全局 `append_dimensions` 字段中指定的维度以外，还会使用该字段中的维度。这是作为高精度指标收集的。
- `mem` – 可选。指定要收集的内存指标。该部分仅适用于 Linux 实例。该部分最多可以包含三个字段：

- **measurement** – 指定一组要收集的内存指标。可能的值为 `active`、`available`、`available_percent`、`buffered`、`cached`、`free`、`inactive`、`total`、`used` 和 `used_percent`。如果包含 `mem`，则该字段是必需的。

要查看每个 `mem` 指标的默认单位，请参阅[Linux 和 macOS 实例上的 CloudWatch 代理收集的指标 \(p. 729\)](#)。

在每个指标的条目中，您可以选择指定下面的一个或两个字段：

- **rename** – 为该指标指定不同的名称。
- **unit** – 指定用于该指标的单位，从而覆盖该指标的默认单位 `None`。您指定的单位必须是有效的 CloudWatch 指标单位，如 [MetricDatum](#) 的 `Unit` 描述中所示。
- **metrics\_collection\_interval** – 可选。指定收集内存指标的频率，从而覆盖配置文件的 `agent` 部分中指定的全局 `metrics_collection_interval`。

该值是以秒为单位指定的。

如果您将该值设置为低于 60 秒，则将每个指标作为高精度指标进行收集。有关高精度指标的更多信息，请参阅[高精度指标 \(p. 147\)](#)。

- **append\_dimensions** – 可选。仅用于内存指标的其他维度。如果指定该字段，除了用于该代理收集的所有类型的指标的 `append_dimensions` 字段中指定的维度以外，还会使用该字段中的维度。
- **net** – 可选。指定要收集的每个联网指标。该部分仅适用于 Linux 实例。该部分最多可以包含四个字段：
  - **resources** – 可选。如果指定一组网络接口，CloudWatch 仅从这些接口中收集指标。否则，将收集所有设备的指标。也可以指定 `*` 值以从所有接口中收集指标。
  - **measurement** – 指定一组要收集的联网指标。可能的值为 `bytes_sent`、`bytes_recv`、`drop_in`、`drop_out`、`err_in`、`err_out`、`packets_sent` 和 `packets_recv`。如果包含 `net`，则该字段是必需的。

要查看每个 `net` 指标的默认单位，请参阅[Linux 和 macOS 实例上的 CloudWatch 代理收集的指标 \(p. 729\)](#)。

在每个指标的条目中，您可以选择指定下面的一个或两个字段：

- **rename** – 为该指标指定不同的名称。
- **unit** – 指定用于该指标的单位，从而覆盖该指标的默认单位 `None`。您指定的单位必须是有效的 CloudWatch 指标单位，如 [MetricDatum](#) 的 `Unit` 描述中所示。
- **metrics\_collection\_interval** – 可选。指定收集网络指标的频率，从而覆盖配置文件的 `agent` 部分中指定的全局 `metrics_collection_interval`。

该值是以秒为单位指定的。例如，指定 10 将导致每 10 秒收集一次指标；将其设置为 300，将指定每 5 分钟收集一次指标。

如果您将该值设置为低于 60 秒，则将每个指标作为高精度指标进行收集。有关高精度指标的更多信息，请参阅[高精度指标 \(p. 147\)](#)。

- **append\_dimensions** – 可选。仅用于网络指标的其他维度。如果指定该字段，除了用于该代理收集的所有类型的指标的 `append_dimensions` 字段中指定的维度以外，还会使用该字段中的维度。
- **netstat** – 可选。指定收集 TCP 连接状态和 UDP 连接指标。该部分仅适用于 Linux 实例。该部分最多可以包含三个字段：
  - **measurement** – 指定一组要收集的网络状态指标。可能的值为 `tcp_close`、`tcp_close_wait`、`tcp_closing`、`tcp_established`、`tcp_fin_wait1`、`tcp_fin_wait2`、`t` 和 `udp_socket`。如果包含 `netstat`，则该字段是必需的。

要查看每个 `netstat` 指标的默认单位，请参阅[Linux 和 macOS 实例上的 CloudWatch 代理收集的指标 \(p. 729\)](#)。

在每个指标的条目中，您可以选择指定下面的一个或两个字段：

- **rename** – 为该指标指定不同的名称。 692



- `unit` – 指定用于该指标的单位，从而覆盖该指标的默认单位 `None`。您指定的单位必须是有效的 CloudWatch 指标单位，如 [MetricDatum](#) 的 `Unit` 描述中所示。
- `metrics_collection_interval` – 可选。指定收集网络状态指标的频率，从而覆盖配置文件的 `agent` 部分中指定的全局 `metrics_collection_interval`。

该值是以秒为单位指定的。

如果您将该值设置为低于 60 秒，则将每个指标作为高精度指标进行收集。有关高精度指标的更多信息，请参阅 [高精度指标](#) (p. 147)。

- `append_dimensions` – 可选。仅用于网络状态指标的其他维度。如果指定该字段，除了用于该代理收集的所有类型的指标的 `append_dimensions` 字段中指定的维度以外，还会使用该字段中的维度。
- `processes` – 可选。指定要收集的进程指标。该部分仅适用于 Linux 实例。该部分最多可以包含三个字段：
  - `measurement` – 指定一组要收集的进程指标。可能的值为 `blocked`、`dead`、`idle`、`paging`、`running`、`sleeping`、`stopped`、`total`、`total_threads`、`wait` 和 `zombies`。如果包含 `processes`，则该字段是必需的。

对于所有 `processes` 指标，默认单位为 `Count`。

在每个指标的条目中，您可以选择指定下面的一个或两个字段：

- `rename` – 为该指标指定不同的名称。
- `unit` – 指定用于该指标的单位，从而覆盖该指标的默认单位 `None`。您指定的单位必须是有效的 CloudWatch 指标单位，如 [MetricDatum](#) 的 `Unit` 描述中所示。
- `metrics_collection_interval` – 可选。指定收集进程指标的频率，从而覆盖配置文件的 `agent` 部分中指定的全局 `metrics_collection_interval`。

该值是以秒为单位指定的。例如，指定 10 将导致每 10 秒收集一次指标；将其设置为 300，将指定每 5 分钟收集一次指标。

如果您将该值设置为低于 60 秒，则将每个指标作为高精度指标进行收集。有关更多信息，请参阅 [高精度指标](#) (p. 147)。

- `append_dimensions` – 可选。仅用于进程指标的其他维度。如果指定该字段，除了用于该代理收集的所有类型的指标的 `append_dimensions` 字段中指定的维度以外，还会使用该字段中的维度。
- `nvidia_gpu` – 可选。指定要收集的 NVIDIA GPU 指标。本部分仅适用于配置了 NVIDIA GPU 加速器并安装了 NVIDIA 系统管理界面 ( `nvidia-smi` ) 的主机上的 Linux 实例。

收集的 NVIDIA GPU 指标以字符串 `nvidia_smi_` 作为前缀以与收集的其他加速器类型的指标进行区分。该部分最多可以包含两个字段：

- `measurement` – 指定一组要收集的 NVIDIA GPU 指标。有关此处可能使用的值的列表，请参阅 [收集 NVIDIA GPU 指标](#) (p. 709) 表中的指标列。

在每个指标的条目中，您可以选择指定以下一个或两个字段：

- `rename` – 为该指标指定不同的名称。
- `unit` – 指定用于该指标的单位，从而覆盖该指标的默认单位 `None`。您指定的单位必须是有效的 CloudWatch 指标单位，如 [MetricDatum](#) 的 `Unit` 描述中所示。
- `metrics_collection_interval` – 可选。指定收集 NVIDIA GPU 指标的频率，从而覆盖配置文件的 `agent` 部分中指定的全局 `metrics_collection_interval`。
- `procstat` – 可选。指定您希望从各个流程检索指标。有关可用于 `procstat` 的配置选项的更多信息，请参阅 [使用 procstat 插件收集流程指标](#) (p. 711)。
- `statsd` – 可选。指定要使用 StatsD 协议检索自定义指标。CloudWatch 代理充当协议的守护进程。您可以使用任何标准 StatsD 客户端将指标发送到 CloudWatch 代理。有关可用于 StatsD 的配置选项的更多信息，请参阅 [使用 StatsD 检索自定义指标](#) (p. 717)。
- `ethtool` – 可选。指定您希望将 ethtool 统计数据导入到 CloudWatch。有关更多信息，请参阅 [收集网络性能指标](#) (p. 707)。

下面是一个适用于 Linux 服务器的 `metrics` 部分示例。在此示例中，会收集三个 CPU 指标、三个 `netstat` 指标、三个流程指标和一个磁盘指标，并将代理设置为从 `collectd` 客户端接收其他指标。

```
"metrics": {
 "metrics_collected": {
 "collectd": {},
 "cpu": {
 "resources": [
 "*"
],
 "measurement": [
 {"name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit": "Percent"},
 {"name": "cpu_usage_nice", "unit": "Percent"},
 "cpu_usage_guest"
],
 "totalcpu": false,
 "metrics_collection_interval": 10,
 "append_dimensions": {
 "test": "test1",
 "date": "2017-10-01"
 }
 },
 "netstat": {
 "measurement": [
 "tcp_established",
 "tcp_syn_sent",
 "tcp_close"
],
 "metrics_collection_interval": 60
 },
 "disk": {
 "measurement": [
 "used_percent"
],
 "resources": [
 "*"
],
 "drop_device": true
 },
 "processes": {
 "measurement": [
 "running",
 "sleeping",
 "dead"
]
 }
 },
 "append_dimensions": {
 "ImageId": "${aws:ImageId}",
 "InstanceId": "${aws:InstanceId}",
 "InstanceType": "${aws:InstanceType}",
 "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
 },
 "aggregation_dimensions" : [["AutoScalingGroupName"], ["InstanceId", "InstanceType"],
[]]
}
```

## Windows Server

在 Windows Server 的 `metrics_collected` 部分中，您可以包含每个 Windows 性能对象的小节，例如，`Memory`、`Processor` 和 `LogicalDisk`。有关可用的对象和计数器的信息，请参阅 Microsoft Windows 文档。

在每个对象的小节中，您可以指定要收集的计数器的 `measurement` 数组。需要为在配置文件中指定的每个对象指定 `measurement` 数组。您也可以指定 `resources` 字段以命名从中收集指标的实例。您还可以为 `resources` 指定 `*`，以便为每个实例收集单独的指标。如果省略 `resources`，则将所有实例的数据汇总到一个组中。对于没有实例的对象，请省略 `resources`。

在每个对象部分中，您还可以指定以下可选字段：

- `metrics_collection_interval` – 可选。指定为该对象收集指标的频率，从而覆盖配置文件的 `agent` 部分中指定的全局 `metrics_collection_interval`。

该值是以秒为单位指定的。例如，指定 10 将导致每 10 秒收集一次指标；将其设置为 300，将指定每 5 分钟收集一次指标。

如果您将该值设置为低于 60 秒，则将每个指标作为高精度指标进行收集。有关更多信息，请参阅[高精度指标 \(p. 147\)](#)。

- `append_dimensions` – 可选。指定仅用于该对象的指标的其他维度。如果指定该字段，除了用于该代理收集的所有类型的指标的全局 `append_dimensions` 字段中指定的维度以外，还会使用该字段中的维度。

在每个计数器部分中，您还可以指定以下可选字段：

- `rename` – 指定在 CloudWatch 中用于该指标的不同名称。
- `unit` – 指定用于该指标的单位。您指定的单位必须是有效的 CloudWatch 指标单位，如 [MetricDatum](#) 的 `Unit` 描述中所示。

您可以在 `metrics_collected` 中包括两个其他可部分：

- `statsd` – 让您能够使用 StatsD 协议检索自定义指标。CloudWatch 代理充当协议的守护进程。您可以使用任何标准 StatsD 客户端将指标发送到 CloudWatch 代理。有关更多信息，请参阅[使用 StatsD 检索自定义指标 \(p. 717\)](#)。
- `procstat` – 让您能够从各个流程检索指标。有关更多信息，请参阅[使用 procstat 插件收集流程指标 \(p. 711\)](#)。

下面是一个用于 Windows Server 的 `metrics` 部分的示例。在此示例中，会收集许多 Windows 指标，并且计算机也被设置为从 StatsD 客户端接收其他指标。

```
"metrics": {
 "metrics_collected": {
 "statsd": {},
 "Processor": {
 "measurement": [
 { "name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent" },
 "% Interrupt Time",
 "% User Time",
 "% Processor Time"
],
 "resources": [
 "*"
],
 "append_dimensions": {
 "d1": "win_foo",
 "d2": "win_bar"
 }
 },
 "LogicalDisk": {
 "measurement": [
 { "name": "% Idle Time", "unit": "Percent" },
 { "name": "% Disk Read Time", "rename": "DISK_READ" },
```

```
 "% Disk Write Time"
],
 "resources": [
 "*"
]
},
"Memory": {
 "metrics_collection_interval": 5,
 "measurement": [
 "Available Bytes",
 "Cache Faults/sec",
 "Page Faults/sec",
 "Pages/sec"
],
 "append_dimensions": {
 "d3": "win_bo"
 }
},
"Network Interface": {
 "metrics_collection_interval": 5,
 "measurement": [
 "Bytes Received/sec",
 "Bytes Sent/sec",
 "Packets Received/sec",
 "Packets Sent/sec"
],
 "resources": [
 "*"
],
 "append_dimensions": {
 "d3": "win_bo"
 }
},
"System": {
 "measurement": [
 "Context Switches/sec",
 "System Calls/sec",
 "Processor Queue Length"
],
 "append_dimensions": {
 "d1": "win_foo",
 "d2": "win_bar"
 }
}
},
"append_dimensions": {
 "ImageId": "${aws:ImageId}",
 "InstanceId": "${aws:InstanceId}",
 "InstanceType": "${aws:InstanceType}",
 "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
},
"aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"], ["d1"], []]
}
}
```

## CloudWatch 代理配置文件：Logs（日志）部分

logs 部分包含以下字段：

- **logs\_collected** – 如果包含 logs 部分，则为必需。指定从服务器中收集的日志文件和 Windows 事件日志。它可以包含两个字段：files 和 windows\_events。
- **files** – 指定 CloudWatch 代理收集的常规日志文件。它包含一个字段 (collect\_list)，它进一步定义这些文件。

- `collect_list` – 如果包含 `files`，则为必需。包含一组条目，每个条目指定一个要收集的日志文件。其中的每个条目可以包含以下字段：
  - `file_path` – 指定要上传到 CloudWatch Logs 的日志文件的路径。接受标准 Unix Glob 匹配规则，并添加 `**` 以作为超级星号。例如，指定 `/var/log/**/*.log` 将导致收集 `/var/log` 目录树中的所有 `.log` 文件。有关更多示例，请参阅 [Glob 库](#)。

您也可以使用标准星号作为标准通配符。例如，`/var/log/system.log*` 与 `/var/log` 中的 `system.log_1111`、`system.log_2222` 等文件匹配。

根据文件修改时间，只有最新文件才会推送到 CloudWatch Logs。我们建议您使用通配符指定同一类型的一系列文件（如 `access_log.2018-06-01-01` 和 `access_log.2018-06-01-02`）而不是多个不同类型的文件（如 `access_log_80` 和 `access_log_443`）。要指定多个类型的文件，请向代理配置文件再添加一个日志流条目，让每种日志文件都转到不同的日志流。

- `auto_removal` – 可选。如果是这样，则 CloudWatch 代理将旧日志文件上传到 CloudWatch Logs 后会自动删除它们。代理仅从创建多个文件的日志中删除完整的文件，例如为每个日期创建单独文件的日志。如果日志连续写入单个文件，则不会删除该日志。

如果您已经有轮换或删除日志文件的方法，建议您忽略此字段或者将其设置为 `false`。

如果省略该字段，则使用 `false` 的默认值。

- `log_group_name` – 可选。指定要在 CloudWatch Logs 中使用的日志组名称。您可以在名称中包含 `{instance_id}`、`{hostname}`、`{local_hostname}` 和 `{ip_address}` 作为变量。`{hostname}` 从 EC2 元数据中检索主机名，`{local_hostname}` 使用网络配置文件中的主机名。

如果使用这些变量创建许多不同的日志组，请记住每个区域每个账户存在 1000000 个日志组的限制。

允许的字符包括 `a-z`、`A-Z`、`0-9`、“`_`”（下划线）、“`-`”（连字符）、“`/`”（正斜杠）和“`.`”（句点）。

我们建议您指定此字段以防止混淆。如果您省略该字段，则将最终圆点之前的文件路径作为日志组名称。例如，如果文件路径为 `/tmp/TestLogFile.log.2017-07-11-14`，则日志组名称为 `/tmp/TestLogFile.log`。

- `log_stream_name` – 可选。指定要在 CloudWatch Logs 中使用的日志流名称。您可以在名称中包含 `{instance_id}`、`{hostname}`、`{local_hostname}` 和 `{ip_address}` 作为变量。`{hostname}` 从 EC2 元数据中检索主机名，`{local_hostname}` 使用网络配置文件中的主机名。

如果省略此字段，则使用全局 `logs` 部分中 `log_stream_name` 参数的值。如果也省略了该值，则使用 `{instance_id}` 的默认值。

如果还没有日志流，则会自动创建一个日志流。

- `retention_in_days` – 可选。在指定的日志组中指定保留日志事件的天数。

如果代理目前正在日志组，并且您指定 `-1` 或忽略此字段，则保留将设置为永不过期。

如果此日志组已存在，并且您指定了此字段，则将使用您指定的新保留期。如果您为已存在的日志组忽略此字段，则日志组的保留不会更改。

有效的值为

`-1`、`1`、`3`、`5`、`7`、`14`、`30`、`60`、`90`、`120`、`150`、`180`、`365`、`400`、`545`、`731`、`1827`、`2192`、`2557`、`2922`、`3281` 和 `3653`。

如果配置代理以将多个日志流写入同一个日志组，则在一个位置指定 `retention_in_days` 将为整个日志组设置日志保留。如果您在多个位置为同一日志组指定 `retention_in_days`，则在所有这些值均相等时才会设置保留。但是，如果在多个位置为同一日志组指定不同的 `retention_in_days` 值，则不会设置日志保留，代理将停止并返回错误。

## Note

代理的 IAM 角色或 IAM 用户必须拥有 `logs:PutRetentionPolicy` 才能设置保留策略。有关更多信息，请参阅 [允许 CloudWatch 代理设置日志保留策略 \(p. 664\)](#)。

## Warning

如果您为已存在的日志组设置 `retention_in_days`，则该日志组中在您指定天数之前发布的所有日志都将被删除。例如，将其设置为 3 会导致 3 天前和之前的所有日志被删除。

- `filters` – 可选。可以包含一组条目，其中的每个条目都指定一个正则表达式和筛选条件类型以指定是否发布或删除与筛选条件匹配的日志条目。如果您忽略此字段，则日志文件中的所有日志都会发布到 CloudWatch Logs。如果您包括此字段，则代理将使用您指定的所有筛选条件处理每条日志消息，并且只有通过所有筛选条件的日志事件才会发布到 CloudWatch Logs。未通过所有筛选条件的日志条目仍将保留在主机的日志文件中，但不会发送到 CloudWatch Logs。

筛选条件数组中的每个条目可以包含以下字段：

- `type`— 表示筛选条件类型。有效值为 `include` 和 `exclude`。对于 `include`，日志条目必须与要发布到 CloudWatch Logs 的表达式匹配。对于 `exclude`，与筛选条件匹配的每个日志条目不会发送到 CloudWatch Logs。
- `expression`— 遵循 [RE2 语法](#) 的正则表达式字符串。

## Note

CloudWatch 代理不会检查您提供的任何正则表达式的性能，也不会限制正则表达式评估的运行时间。我们建议您不要编写评估昂贵的表达式。有关可能问题的更多信息，请参阅 [正则表达式拒绝服务 - ReDoS](#)

例如，以下 CloudWatch 代理配置文件的摘录会向 CloudWatch Logs 发布 PUT 和 POST 请求的日志，但不包括来自 Firefox 的日志。

```
"collect_list": [
 {
 "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/test.log",
 "log_group_name": "test.log",
 "log_stream_name": "test.log",
 "filters": [
 {
 "type": "exclude",
 "expression": "Firefox"
 },
 {
 "type": "include",
 "expression": "P(UT|OST)"
 }
]
 },

]
```

## Note

配置文件中筛选条件的顺序对性能至关重要。在上一个示例中，代理在开始评估第二个筛选条件之前会删除与 `Firefox` 匹配的所有日志。为了减少多个筛选条件评估的日志条目，首先将预计排除更多日志的筛选条件放在配置文件中。

- `timezone` – 可选。指定在日志事件中放置时间戳时要使用的时区。有效值为 `UTC` 和 `Local`。默认值为 `Local`。

如果您不指定 `timestamp_format` 的值，此参数将被忽略。

- `timestamp_format` – 可选。使用纯文本和以 % 开头的特殊符号指定时间戳格式。如果省略该字段，则使用当前时间。如果使用该字段，您可以使用以下列表中的符号作为格式的一部分。

如果单个日志条目包含两个与该格式相符的时间戳，则使用第一个时间戳。

此符号列表与较旧的 CloudWatch Logs 代理使用的列表不同。有关这些区别的摘要，请参阅[统一的 CloudWatch 代理和更早的 CloudWatch Logs 代理之间的时间戳差异 \(p. 747\)](#)。

`%y`

使用以零填充的十进制数字形式的没有世纪的年份。例如，19 表示 2019 年。

`%Y`

使用十进制数字形式的具有世纪的年份。例如：2019。

`%b`

使用区域设置的缩写名称形式的月份

`%B`

使用区域设置的完整名称形式的月份

`%m`

使用以零填充的十进制数字形式的月份

`%-m`

十进制数字形式的月份 (不使用零填充)

`%d`

使用以零填充的十进制数字形式的日期

`%-d`

十进制数字形式的日期 (不使用零填充)

`%A`

星期的全称，例如 Monday

`%a`

星期的缩写，例如 Mon

`%H`

使用以零填充的十进制数字形式的小时 (24 小时制)

`%I`

使用以零填充的十进制数字形式的小时 (12 小时制)

`%-I`

十进制数字形式的小时 (12 小时制) (不使用零填充)

`%p`

AM 或 PM

`%M`

使用以零填充的十进制数字形式的分钟

`%-M`

十进制数字形式的分钟 (不使用零填充)



%S

使用以零填充的十进制数字形式的秒

%-S

十进制数字形式的秒 (不使用零填充)

%f

小数秒采用十进制数 ( 1-9 位数 ) , 左侧用零填充。

%Z

时区, 例如 PST

%z

时区, 以本地时区与 UTC 的偏移量表示。例如: -0700。仅支持此格式。例如, -07:00 不是有效格式。

- `multi_line_start_pattern` – 指定用于识别日志消息开头的模式。日志消息由与模式匹配的行以及不与模式匹配的任何后续行组成。

如果您省略此字段, 则会禁用多行模式, 而且以非空格字符开头的任何行都会使上一个日志消息结束 (如果存在), 并开始新的日志消息。

如果包含该字段, 您可以指定 `{timestamp_format}` 以使用与您的时间戳格式相同的正则表达式。否则, 您可以为 CloudWatch Logs 指定不同的正则表达式, 以用于确定多行条目的起始行。

- `encoding` – 指定日志文件的编码, 以便正确读取该文件。如果您指定错误的编码, 则可能导致数据丢失, 因为无法解码的字符将被其他字符替代。

默认值为 `utf-8`。以下是所有可能值:

`ascii, big5, euc-jp, euc-kr, gbk, gb18030, ibm866, iso2022-jp, iso8859-2, iso8859-3, iso8859-4, iso8859-5, iso8859-6, iso8859-7, iso8859-8, iso8859-8-i, iso8859-10, iso8859-13, iso8859-14, iso8859-15, iso8859-16, koi8-r, koi8-u, macintosh, shift_jis, utf-8, utf-16, windows-874, windows-1250, windows-1251, windows-1252, windows-1253, windows-1254, windows-1255, windows-1256, windows-1257, windows-1258, x-mac-cyrillic`

- `windows_events` 部分指定从运行 Windows Server 的服务器中收集的 Windows 事件类型。其中包括以下字段:
  - `collect_list` – 如果包含 `windows_events`, 则为必需。指定要收集的 Windows 事件的类型和级别。要收集的每个日志在该部分中包含一个条目, 其中可以包含以下字段:
  - `event_name` – 指定要记录的 Windows 事件的类型。这等同于 Windows 事件日志通道名称: 例如 `System`、`Security`、`Application` 等。要记录的每种类型的 Windows 事件需要使用该字段。

#### Note

当 CloudWatch 从 Windows 日志通道检索消息时, 它会根据其 `Full Name` 属性查找日志通道。同时, Windows 事件查看器导航窗格将显示日志通道的 `Log Name` 属性。`Full Name` 和 `Log Name` 并不总是匹配。确认通道的 `Full Name`, 在 Windows 事件查看器中右键单击, 然后打开 `Properties` (属性)。

- `event_levels` – 指定要录入的事件的级别。您必须指定要录入的每个级别。可能的值包括 `INFORMATION`、`WARNING`、`ERROR`、`CRITICAL` 和 `VERBOSE`。要记录的每种类型的 Windows 事件需要使用该字段。
- `log_group_name` – 必需。指定要在 CloudWatch Logs 中使用的日志组名称。
- `log_stream_name` – 可选。指定要在 CloudWatch Logs 中使用的日志流名称。您可以在名称中包含 `{instance_id}`、`{hostname}`、`{local_hostname}` 和 `{ip_address}` 作为变



量。{hostname} 从 EC2 元数据中检索主机名，{local\_hostname} 使用网络配置文件中的主机名。

如果省略此字段，则使用全局 logs 部分中 log\_stream\_name 参数的值。如果也省略了该值，则使用 {instance\_id} 的默认值。

如果还没有日志流，则会自动创建一个日志流。

- event\_format – 可选。指定在 CloudWatch Logs 中存储 Windows 事件时要使用的格式。与在 Windows 事件查看器中一样，xml 使用 XML 格式。text 使用原 CloudWatch Logs 代理格式。
- log\_stream\_name – 必需。对于未在 collect\_list 的相应条目内的 log\_stream\_name 参数中定义各日志流名称的任何日志或 Windows 事件，指定要用于它们的默认日志流名称。
- endpoint\_override – 指定一个 FIPS 端点或私有链接，以用作代理在其中发送日志的端点。如果指定此字段并设置私有链接，您可以将日志发送到 Amazon VPC 终端节点。有关更多信息，请参阅 [Amazon VPC 是什么？](#)。

endpoint\_override 的值必须是表示 URL 的字符串。

例如，在配置文件的日志部分中，以下内容将代理设置为在发送日志时使用 VPC 终端节点。

```
{
 "logs": {
 "endpoint_override": "vpce-xxxxxxxxxxxxxxxxxxxxx.logs.us-east-1.vpce.amazonaws.com",

 },
}
```

- force\_flush\_interval – 以秒为单位指定日志在发送到服务器之前保留在内存缓冲区中的最大时间量。无论此字段的设置如何，如果缓冲区中的日志大小达到 1 MB，日志会立即发送到服务器。默认值为 5。
- credentials – 指定一个在向不同 Amazon 账户发送日志时使用的 IAM 角色。如果指定，则此字段包含一个参数 role\_arn。
  - role\_arn – 指定一个在向不同 Amazon 账户发送日志时用于身份验证的 IAM 角色的 ARN。有关更多信息，请参阅 [向不同账户发送指标和日志 \(p. 746\)](#)。如果在此指定，它将覆盖在配置文件的 agent 部分指定的 role\_arn (如果有)。
- metrics\_collected – 指定代理将收集日志中嵌入的指标。目前，metrics\_collected 字段只能包含 emf 字段。
  - emf – 指定代理将收集采用嵌入式指标格式的日志。您可以从这些日志生成指标数据。有关更多信息，请参阅 [采用 CloudWatch 嵌入式指标格式摄取高基数日志以生成指标 \(p. 754\)](#)。

下面是一个 logs 部分示例。

```
"logs":
{
 "logs_collected": {
 "files": {
 "collect_list": [
 {
 "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-cloudwatch-agent.log",
 "log_group_name": "amazon-cloudwatch-agent.log",
 "log_stream_name": "my_log_stream_1",
 "timestamp_format": "%H: %M: %S%y%b%-d"
 },
 {
 "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\test.log",
 "log_group_name": "test.log",
```

```
 "log_stream_name": "my_log_stream_name_2"
 }
],
 "windows_events": {
 "collect_list": [
 {
 "event_name": "System",
 "event_levels": [
 "INFORMATION",
 "ERROR"
],
 "log_group_name": "System",
 "log_stream_name": "System"
 },
 {
 "event_name": "CustomizedName",
 "event_levels": [
 "INFORMATION",
 "ERROR"
],
 "log_group_name": "CustomizedLogGroup",
 "log_stream_name": "CustomizedLogStream"
 }
]
 }
 },
 "log_stream_name": "my_log_stream_name"
}
```

## CloudWatch 代理如何处理稀疏日志文件

稀疏文件是具有空块和真实内容的文件。稀疏文件通过将表示空块的简短信息（而不是构成此块的实际 Null 字节）写入磁盘，以更有效地使用磁盘空间。这使得稀疏文件的实际大小通常比其表现大小小得多。

但是，CloudWatch 代理对稀疏文件的处理方式与对普通文件的处理方式不同。当代理读取稀疏文件时，空块被视为填充 Null 字节的“真实”块。因此，CloudWatch 代理向 CloudWatch 发布的字节数与稀疏文件的表现大小相同。

配置 CloudWatch 代理来发布稀疏文件可能会导致高于预期 CloudWatch 成本，因此我们建议不要这样做。例如，Linux 中的 `/var/logs/lastlog` 通常是一个非常稀疏的文件，我们建议您不要将其发布到 CloudWatch。

## CloudWatch 代理配置文件：完整示例

下面是一个适用于 Linux 服务器的完整 CloudWatch 代理配置文件示例。

要收集的指标的 `measurement` 部分中列出的项目可能指定了完整的指标名称，也可能仅指定将附加到资源类型的指标名称部分。例如，如果在 `diskio` 部分的 `measurement` 部分中指定 `reads` 或 `diskio_reads`，将导致收集 `diskio_reads` 指标。

该示例提供了两种方法以在 `measurement` 部分中指定指标。

```
{
 "agent": {
 "metrics_collection_interval": 10,
 "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log"
 },
 "metrics": {
 "namespace": "MyCustomNamespace",
 "metrics_collected": {
 "cpu": {
 "resources": [
```

```
 "*"
],
 "measurement": [
 { "name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit": "Percent" },
 { "name": "cpu_usage_nice", "unit": "Percent" },
 "cpu_usage_guest"
],
 "totalcpu": false,
 "metrics_collection_interval": 10,
 "append_dimensions": {
 "customized_dimension_key_1": "customized_dimension_value_1",
 "customized_dimension_key_2": "customized_dimension_value_2"
 }
},
"disk": {
 "resources": [
 "/",
 "/tmp"
],
 "measurement": [
 { "name": "free", "rename": "DISK_FREE", "unit": "Gigabytes" },
 "total",
 "used"
],
 "ignore_file_system_types": [
 "sysfs", "devtmpfs"
],
 "metrics_collection_interval": 60,
 "append_dimensions": {
 "customized_dimension_key_3": "customized_dimension_value_3",
 "customized_dimension_key_4": "customized_dimension_value_4"
 }
},
"diskio": {
 "resources": [
 "*"
],
 "measurement": [
 "reads",
 "writes",
 "read_time",
 "write_time",
 "io_time"
],
 "metrics_collection_interval": 60
},
"swap": {
 "measurement": [
 "swap_used",
 "swap_free",
 "swap_used_percent"
]
},
"mem": {
 "measurement": [
 "mem_used",
 "mem_cached",
 "mem_total"
],
 "metrics_collection_interval": 1
},
"net": {
 "resources": [
 "eth0"
],
 "measurement": [
```

```
 "bytes_sent",
 "bytes_recv",
 "drop_in",
 "drop_out"
]
},
"netstat": {
 "measurement": [
 "tcp_established",
 "tcp_syn_sent",
 "tcp_close"
],
 "metrics_collection_interval": 60
},
"processes": {
 "measurement": [
 "running",
 "sleeping",
 "dead"
]
}
},
"append_dimensions": {
 "ImageId": "${aws:ImageId}",
 "InstanceId": "${aws:InstanceId}",
 "InstanceType": "${aws:InstanceType}",
 "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
},
"aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"], ["d1"],
[]],
 "force_flush_interval" : 30
},
"logs": {
 "logs_collected": {
 "files": {
 "collect_list": [
 {
 "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-
agent.log",
 "log_group_name": "amazon-cloudwatch-agent.log",
 "log_stream_name": "amazon-cloudwatch-agent.log",
 "timezone": "UTC"
 },
 {
 "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/test.log",
 "log_group_name": "test.log",
 "log_stream_name": "test.log",
 "timezone": "Local"
 }
]
 }
 },
 "log_stream_name": "my_log_stream_name",
 "force_flush_interval" : 15
}
}
```

下面是一个适用于运行 Windows Server 的服务器的完整 CloudWatch 代理配置文件示例。

```
{
 "agent": {
 "metrics_collection_interval": 60,
 "logfile": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-
cloudwatch-agent.log"
 },
}
```

```
"metrics": {
 "namespace": "MyCustomNamespace",
 "metrics_collected": {
 "Processor": {
 "measurement": [
 {"name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent"},
 "% Interrupt Time",
 "% User Time",
 "% Processor Time"
],
 "resources": [
 "*"
],
 "append_dimensions": {
 "customized_dimension_key_1": "customized_dimension_value_1",
 "customized_dimension_key_2": "customized_dimension_value_2"
 }
 },
 "LogicalDisk": {
 "measurement": [
 {"name": "% Idle Time", "unit": "Percent"},
 {"name": "% Disk Read Time", "rename": "DISK_READ"},
 "% Disk Write Time"
],
 "resources": [
 "*"
]
 },
 "customizedObjectName": {
 "metrics_collection_interval": 60,
 "customizedCounterName": [
 "metric1",
 "metric2"
],
 "resources": [
 "customizedInstances"
]
 },
 "Memory": {
 "metrics_collection_interval": 5,
 "measurement": [
 "Available Bytes",
 "Cache Faults/sec",
 "Page Faults/sec",
 "Pages/sec"
]
 },
 "Network Interface": {
 "metrics_collection_interval": 5,
 "measurement": [
 "Bytes Received/sec",
 "Bytes Sent/sec",
 "Packets Received/sec",
 "Packets Sent/sec"
],
 "resources": [
 "*"
],
 "append_dimensions": {
 "customized_dimension_key_3": "customized_dimension_value_3"
 }
 },
 "System": {
 "measurement": [
 "Context Switches/sec",
 "System Calls/sec",
```

```
 "Processor Queue Length"
]
 },
 "append_dimensions": {
 "ImageId": "${aws:ImageId}",
 "InstanceId": "${aws:InstanceId}",
 "InstanceType": "${aws:InstanceType}",
 "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
 },
 "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"], ["d1"], []]
},
"logs": {
 "logs_collected": {
 "files": {
 "collect_list": [
 {
 "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-
cloudwatch-agent.log",
 "log_group_name": "amazon-cloudwatch-agent.log",
 "timezone": "UTC"
 },
 {
 "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\
\\test.log",
 "log_group_name": "test.log",
 "timezone": "Local"
 }
]
 },
 "windows_events": {
 "collect_list": [
 {
 "event_name": "System",
 "event_levels": [
 "INFORMATION",
 "ERROR"
],
 "log_group_name": "System",
 "log_stream_name": "System",
 "event_format": "xml"
 },
 {
 "event_name": "CustomizedName",
 "event_levels": [
 "WARNING",
 "ERROR"
],
 "log_group_name": "CustomizedLogGroup",
 "log_stream_name": "CustomizedLogStream",
 "event_format": "xml"
 }
]
 }
 },
 "log_stream_name": "example_log_stream_name"
}
```

## 手动保存 CloudWatch 代理配置文件

如果手动创建或编辑 CloudWatch 代理配置文件，则可以给它指定任何名称。为了简化故障排除，我们建议您在 Linux 服务器上将其命名为 `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`，在运行 Windows Server 的服务器上将其命名为 `$Env:ProgramData`

\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json。创建该文件后，您可以将其复制到要运行代理的其他服务器上。

## 将 CloudWatch 代理配置文件上传到 Systems Manager Parameter Store

如果您计划使用 SSM Agent 在服务器上安装 CloudWatch 代理，则在手动编辑 CloudWatch 代理配置文件后，可以将其上传到 Systems Manager Parameter Store。为此，您可以使用 Systems Manager `put-parameter` 命令。

要能够将该文件存储在 Parameter Store 中，您必须使用具有足够权限的 IAM 角色。有关更多信息，请参阅[创建 IAM 角色和用户以用于 CloudWatch 代理 \(p. 669\)](#)。

可以使用以下命令，其中 `parameter name` 是在 Parameter Store 中用于该文件的名称，而 `configuration_file_pathname` 是已编辑的配置文件的完整路径和文件名。

```
aws ssm put-parameter --name "parameter name" --type "String" --value
file://configuration_file_pathname
```

## 收集网络性能指标

在 Linux 上运行的 EC2 实例使用 Elastic Network Adapter (ENA) 发布网络性能指标。版本 1.246396.0 和更高版本的 CloudWatch 代理使您能够将这些网络性能指标导入到 CloudWatch 中。当您将这些网络性能指标导入到 CloudWatch 时，它们将作为 CloudWatch 自定义指标收费。

有关 ENA 驱动程序的更多信息，请参阅在[Linux 实例上启用 Elastic Network Adapter \(ENA\) 增强联网](#)和在[Windows 实例上启用 Elastic Network Adapter \(ENA\) 增强联网](#)。

在 Linux 服务器和 Windows 服务器上，设置网络性能指标集合的方式不同。

下表列出了 ENA 适配器启用的这些网络性能指标。当 CloudWatch 代理从 Linux 实例将这些指标导入 CloudWatch 时，它会在每个指标名称的开头前置 `ethtool_`。

| 指标                                                                                                                      | 描述                                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Linux 服务器上的名称： <b>bw_in_allowance_exceeded</b><br><br>Windows 服务器上的名称： <b>Aggregate inbound BW allowance exceeded</b>   | 因入站聚合带宽超过实例的最大值而排队和/或丢弃的数据包的数量。<br><br>仅当您将此指标列在 CloudWatch 代理配置文件 <code>metrics_collected</code> 部分的 <code>ethtool</code> 子部分时，才会收集该指标。有关更多信息，请参阅 <a href="#">收集网络性能指标 (p. 707)</a><br><br>单位：无 |
| Linux 服务器上的名称： <b>bw_out_allowance_exceeded</b><br><br>Windows 服务器上的名称： <b>Aggregate outbound BW allowance exceeded</b> | 因出站聚合带宽超过实例的最大值而排队和/或丢弃的数据包的数量。<br><br>仅当您将此指标列在 CloudWatch 代理配置文件 <code>metrics_collected</code> 部分的 <code>ethtool</code> 子部分时，才会收集该指标。有关更多信息，请参阅 <a href="#">收集网络性能指标 (p. 707)</a><br><br>单位：无 |
| Linux 服务器上的名称： <b>conntrack_allowance_exceeded</b>                                                                      | 由于连接跟踪超过实例的最大值且无法建立新连接而丢弃的数据包的数量。这可能会导致进出实例的流量丢失数据包。                                                                                                                                             |

| 指标                                                                                                                               | 描述                                                                                                                                                                                                                                                                 |
|----------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Windows 服务器上的名称： <b>Connection tracking allowance exceeded</b>                                                                   | <p>仅当您将此指标列在 CloudWatch 代理配置文件 <code>metrics_collected</code> 部分的 <code>ethtool</code> 子部分时，才会收集该指标。有关更多信息，请参阅 <a href="#">收集网络性能指标 (p. 707)</a></p> <p>单位：无</p>                                                                                                   |
| <p>Linux 服务器上的名称：<b>linklocal_allowance_exceeded</b></p> <p>Windows 服务器上的名称：<b>Link local packet rate allowance exceeded</b></p> | <p>由于到本地代理服务的流量的 PPS 超出网络接口的最大值而丢弃的数据包数量。这会影响流向 DNS 服务、实例元数据服务和 Amazon Time Sync Service 的流量。</p> <p>仅当您将此指标列在 CloudWatch 代理配置文件 <code>metrics_collected</code> 部分的 <code>ethtool</code> 子部分时，才会收集该指标。有关更多信息，请参阅 <a href="#">收集网络性能指标 (p. 707)</a></p> <p>单位：无</p> |
| <p>Linux 服务器上的名称：<b>pps_allowance_exceeded</b></p> <p>Windows 服务器上的名称：<b>PPS allowance exceeded</b></p>                          | <p>因双向 PPS 超过实例的最大值而排队和/或丢弃的数据包的数量。</p> <p>仅当您将此指标列在 CloudWatch 代理配置文件 <code>metrics_collected</code> 部分的 <code>ethtool</code> 子部分时，才会收集该指标。有关更多信息，请参阅 <a href="#">收集网络性能指标 (p. 707)</a></p> <p>单位：无</p>                                                           |

## Linux 设置

在 Linux 服务器上，`ethtool` 插件使您能够将网络性能指标导入到 CloudWatch 中。

`ethtool` 是一个标准的 Linux 实用工具，可以收集有关 Linux 服务器上以太网设备的统计数据。它收集的统计数据取决于网络设备和驱动程序。这些统计数据的示例包括 `tx_packets`、`rx_bytes`、`tx_errors` 和 `align_errors`。当您使用 `ethtool` 插件与 CloudWatch 代理一起使用时，您还可以将这些统计数据以及本部分前面列出的 EC2 网络性能指标导入到 CloudWatch 中。

当 CloudWatch 代理将指标导入 CloudWatch 时，它将为所有导入指标的名称添加一个 `ethtool_` 前缀。所以标准的 `ethtool` 统计数据 `rx_bytes` 在 CloudWatch 中被称为 `ethtool_rx_bytes`，EC2 网络性能指标 `bw_in_allowance_exceeded` 在 CloudWatch 中被称为 `ethtool_bw_in_allowance_exceeded`。

在 Linux 服务器上，如需导入 `ethtool` 指标，请为 CloudWatch 代理配置文件 `metrics_collected` 部分添加 `ethtool` 部分。`ethtool` 部分包含以下子部分：

- `interface_include` – 包括此部分会导致代理仅从本节中列出名称的接口收集指标。如果省略此部分，则从 `interface_exclude` 中未列出的所有以太网接口收集指标。

默认以太网接口为 `eth0`。

- `interface_exclude` – 如果包含此部分，请列出您不希望从中收集指标的以太网接口。

`ethtool` 插件始终忽略环回接口。

- `metrics_include` – 此部分列出了要导入到 CloudWatch 中的指标。它可以包括 `ethtool` 收集的标准统计数据和 Amazon EC2 高精度网络指标。

以下示例显示 CloudWatch 代理配置文件的一部分。此配置收集标准的 `ethtool` 指标 `rx_packets` 和 `tx_packets`，以及仅来自 `eth1` 接口的 Amazon EC2 网络性能指标。



有关 CloudWatch 代理配置文件的更多信息，请参阅 [手动创建或编辑 CloudWatch 代理配置文件 \(p. 686\)](#)。

```
"metrics": {
 "append_dimensions": {
 "InstanceId": "${aws:InstanceId}"
 },
 "metrics_collected": {
 "ethtool": {
 "interface_include": [
 "eth1"
],
 "metrics_include": [
 "rx_packets",
 "tx_packets",
 "bw_in_allowance_exceeded",
 "bw_out_allowance_exceeded",
 "conntrack_allowance_exceeded",
 "linklocal_allowance_exceeded",
 "pps_allowance_exceeded"
]
 }
 }
}
```

## 查看网络性能指标

将网络性能指标导入到 CloudWatch 后，您可以以时间序列图形的形式查看这些指标，并创建可监视这些指标的告警，并在它们超出指定的阈值时通知您。以下程序介绍了如何以时间序列图形的方式查看 ethtool 指标。有关设置告警的更多信息，请参阅 [使用 Amazon CloudWatch 告警 \(p. 151\)](#)。

由于所有这些指标都是聚合计数器，因此您可以使用 CloudWatch 指标数学函数（如 `RATE(METRICS())`）在图形中计算这些指标的比率，或使用它们设置告警。有关指标数学函数的更多信息，请参阅 [使用指标数学 \(p. 116\)](#)。

在 CloudWatch 控制台中查看网络性能指标

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 为该代理收集的指标选择命名空间。默认情况下，该命名空间为 CWAgent，但您可能已经在 CloudWatch 代理配置文件中指定了不同的命名空间。
4. 选择指标维度（例如 Per-Instance Metrics（每个实例的指标））。
5. All metrics 选项卡显示命名空间中该维度的所有指标。您可执行以下操作：
  - a. 要为指标绘制图表，请选中该指标旁的复选框。要选择所有指标，请选中表的标题行中的复选框。
  - b. 要对表进行排序，请使用列标题。
  - c. 要按资源进行筛选，请选择资源 ID，然后选择 Add to search（添加到搜索）。
  - d. 要按指标进行筛选，请选择指标名称，然后选择 Add to search（添加到搜索）。
6. （可选）要将此图表添加到 CloudWatch 控制面板，请选择 Actions（操作），然后选择 Add to dashboard（添加到控制面板）。

## 收集 NVIDIA GPU 指标

您可以使用 CloudWatch 代理从 Linux 服务器收集 NVIDIA GPU 指标。要对此进行设置，请在 CloudWatch 代理配置文件的 `metrics_collected` 部分中添加 `nvidia_gpu` 部分。有关更多信息，请参阅 [Linux 部分 \(p. 689\)](#)。

可以收集以下指标。所有这些指标都在没有 CloudWatch Unit 的情况下收集，但是您可以通过向 CloudWatch 代理配置文件添加参数来为每个指标指定单位。有关更多信息，请参阅[Linux 部分 \(p. 689\)](#)。

| 指标                        | CloudWatch 的指标名称                     | 描述                               |
|---------------------------|--------------------------------------|----------------------------------|
| utilization_gpu           | nvidia_smi_utilization_gpu           | 在过去的采样周期内 GPU 上的一个或多个内核运行的时间百分比。 |
| temperature_gpu           | nvidia_smi_temperature_gpu           | GPU 核心温度（以摄氏度为单位）。               |
| power_draw                | nvidia_smi_power_draw                | 上次测量的整个显卡功耗（以瓦为单位）。              |
| utilization_memory        | nvidia_smi_utilization_memory        | 在过去的样本周期内读取或写入全局（设备）内存的时间百分比。    |
| fan_speed                 | nvidia_smi_fan_speed                 | 设备风扇目前预计以最大风扇速度运行的百分比。           |
| memory_total              | nvidia_smi_memory_total              | 报告的总内存（以 MB 为单位）。                |
| memory_used               | nvidia_smi_memory_used               | 已使用的内存（以 MB 为单位）。                |
| memory_free               | nvidia_smi_memory_free               | 空闲内存（以 MB 为单位）。                  |
| pcie_link_generation      | nvidia_smi_pcie_link_generation      | 当前链接生成。                          |
| pcie_link_width           | nvidia_smi_pcie_link_width           | 当前链接宽度。                          |
| encoder_stats_active      | nvidia_smi_encoder_stats_active      | 当前编码器会话数量。                       |
| encoder_stats_avg_fps     | nvidia_smi_encoder_stats_avg_fps     | 每秒编码帧数的移动平均值。                    |
| encoder_stats_avg_latency | nvidia_smi_encoder_stats_avg_latency | 编码延迟的移动平均值（以微秒为单位）。              |
| clocks_current_graphics   | nvidia_smi_clocks_current_graphics   | 显卡（着色器）时钟的当前频率。                  |
| clocks_current_sm         | nvidia_smi_clocks_current_sm         | 流式多处理器（SM）时钟的当前频率。               |
| clocks_current_memory     | nvidia_smi_clocks_current_memory     | 内存时钟的当前频率。                       |
| clocks_current_video      | nvidia_smi_clocks_current_video      | 视频（编码器加解码器）时钟的当前频率。              |

所有这些指标都使用以下维度收集：

| 维度    | 描述                                         |  |
|-------|--------------------------------------------|--|
| index | 此服务器上 GPU 的唯一标识符。表示设备的 NVIDIA 管理库（NVML）索引。 |  |
| name  | GPU 类型。例如，NVIDIA Tesla A100                |  |
| host  | 服务器主机名。                                    |  |

## 使用 procstat 插件收集流程指标

procstat 插件让您能够从各个流程收集指标。这在 Linux 服务器上以及运行 Windows Server 2012 或更高版本的服务器上受支持。

### 主题

- [配置 CloudWatch 代理用于 procstat \(p. 711\)](#)
- [Procstat 收集的指标 \(p. 713\)](#)
- [查看由 CloudWatch 代理导入的进程指标 \(p. 716\)](#)

## 配置 CloudWatch 代理用于 procstat

要使用 procstat 插件，请在 CloudWatch 代理配置文件的 `metrics_collected` 部分中添加 `procstat` 部分。您可以通过以下三种方式指定要监控的进程。您可以仅使用其中一个方法，但您可以使用该方法来指定一个或多个要监控的进程。

- `pid_file`：按照它们创建的进程标识号 (PID) 文件的名称选择进程。
- `exe`：使用正则表达式匹配规则，选择进程名称与您指定的字符串相匹配的进程。匹配是“包含”匹配，这意味着如果您指定 `agent` 作为要匹配的术语，则具有类似于 `cloudwatchagent` 的名称的进程将匹配该术语。有关更多信息，请参阅[语法](#)。
- `pattern`：通过用于启动进程的命令行选择进程。使用正则表达式匹配规则，选择具有与指定字符串匹配的命令行的所有进程。将检查整个命令行，包括与命令一起使用的参数和选项。

匹配是“包含”匹配，这意味着如果您指定 `-c` 作为要匹配的术语，则具有类似于 `-config` 的参数的进程将匹配该术语。

CloudWatch 代理仅使用其中一个方法，即使您包含多个以上部分。如果您指定多个部分，CloudWatch 代理将使用 `pid_file` 部分（如果存在）。否则，它使用 `exe` 部分。

在 Linux 服务器上，您在 `exe` 或 `pattern` 部分中指定的字符串的计算结果为正则表达式。在运行 Windows Server 的服务器上，这些字符串的计算结果为 WMI 查询。例如，`pattern: "%apache%"` 就是一个示例。有关更多信息，请参阅[LIKE 运算符](#)。

无论使用哪种方法，您都可以包含一个可选 `metrics_collection_interval` 参数，该参数以秒为单位指定收集这些指标的频率。如果省略该参数，则使用 60 秒的默认值。

在以下部分的示例中，`procstat` 部分是唯一包含在代理配置文件的 `metrics_collected` 部分中的部分。实际配置文件也在 `metrics_collected` 中包含其他部分。有关更多信息，请参阅[手动创建或编辑 CloudWatch 代理配置文件 \(p. 686\)](#)。

### 使用 pid\_file 进行配置

以下示例 `procstat` 部分监控创建 PID 文件 `example1.pid` 和 `example2.pid` 的进程。从每个进程收集不同的指标。默认情况下，每 10 秒收集一次从创建 `example2.pid` 的进程中收集的指标，每 60 秒收集一次从 `example1.pid` 进程中收集的指标。

```
{
 "metrics": {
 "metrics_collected": {
 "procstat": [
 {
 "pid_file": "/var/run/example1.pid",
 "measurement": [
 "cpu_usage",
 "memory_rss"
]
 },
 {

```

```
 "pid_file": "/var/run/example2.pid",
 "measurement": [
 "read_bytes",
 "read_count",
 "write_bytes"
],
 "metrics_collection_interval": 10
 }
}
}
```

### 使用 exe 进行配置

以下示例 procstat 部分监控名称与字符串 agent 或 plugin 相匹配的所有进程。从每个进程收集相同的指标。

```
{
 "metrics": {
 "metrics_collected": {
 "procstat": [
 {
 "exe": "agent",
 "measurement": [
 "cpu_time",
 "cpu_time_system",
 "cpu_time_user"
]
 },
 {
 "exe": "plugin",
 "measurement": [
 "cpu_time",
 "cpu_time_system",
 "cpu_time_user"
]
 }
]
 }
 }
}
```

### 使用模式进行配置

以下示例 procstat 部分监控命令行与字符串 config 或 -c 相匹配的所有进程。从每个进程收集相同的指标。

```
{
 "metrics": {
 "metrics_collected": {
 "procstat": [
 {
 "pattern": "config",
 "measurement": [
 "rlimit_memory_data_hard",
 "rlimit_memory_data_soft",
 "rlimit_memory_stack_hard",
 "rlimit_memory_stack_soft"
]
 },
 {
 "pattern": "-c",

```

```
 "measurement": [
 "rlimit_memory_data_hard",
 "rlimit_memory_data_soft",
 "rlimit_memory_stack_hard",
 "rlimit_memory_stack_soft"
]
 }
}
```

## Procstat 收集的指标

下表列出可以使用 procstat 插件收集的指标。

CloudWatch 代理将 procstat 添加到以下指标名称的开头。有不同的语法，具体取决于是从 Linux 服务器还是从运行 Windows Server 的服务器收集指标。例如，从 Linux 中收集时，cpu\_time 指标显示为 procstat\_cpu\_time；从 Windows Server 中收集时，它显示为 procstat cpu\_time。

| 指标名称            | 可用于                        | 描述                                                    |
|-----------------|----------------------------|-------------------------------------------------------|
| cpu_time        | Linux                      | 进程使用 CPU 的时间量。该指标以百分之一秒为单位。<br><br>单位：计数              |
| cpu_time_system | Linux、Windows Server、macOS | 进程处于系统模式的时间量。该指标以百分之一秒为单位。<br><br>类型：浮点值<br><br>单位：计数 |
| cpu_time_user   | Linux、Windows Server、macOS | 进程处于用户模式的时间量。该指标以百分之一秒为单位。<br><br>单位：计数               |
| cpu_usage       | Linux、Windows Server、macOS | 进程在任何容量中处于活动状态的时间百分比。<br><br>单位：百分比                   |
| memory_data     | Linux、macOS                | 进程用于数据的内存量。<br><br>单位：字节                              |
| memory_locked   | Linux、macOS                | 进程已锁定的内存量。<br><br>单位：字节                               |

| 指标名称         | 可用于                        | 描述                                                                                                                                                                           |
|--------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| memory_rss   | Linux、Windows Server、macOS | 进程使用的实际内存量（驻留集）。<br><br>单位：字节                                                                                                                                                |
| memory_stack | Linux、macOS                | 进程使用的堆栈内存量。<br><br>单位：字节                                                                                                                                                     |
| memory_swap  | Linux、macOS                | 进程使用的交换内存量。<br><br>单位：字节                                                                                                                                                     |
| memory_vms   | Linux、Windows Server、macOS | 进程使用的虚拟内存量。<br><br>单位：字节                                                                                                                                                     |
| pid          | Linux、Windows Server、macOS | 进程标识符 (ID)。<br><br>单位：计数                                                                                                                                                     |
| pid_count    | Linux、Windows Server、macOS | 与此进程关联的进程 ID 的数量。<br><br>在 Linux 服务器和 macOS 电脑上，此指标的完整名称为 <code>procstat_lookup_pid_count</code> 。<br>在 Windows 服务器上则为 <code>procstat_lookup_pid_count</code> 。<br><br>单位：计数 |
| read_bytes   | Linux、Windows Server       | 进程已从磁盘读取的字节数。<br><br>单位：字节                                                                                                                                                   |
| write_bytes  | Linux、Windows Server       | 进程已写入到磁盘的字节数。<br><br>单位：字节                                                                                                                                                   |
| read_count   | Linux、Windows Server       | 进程已执行的磁盘读取操作的数目。<br><br>单位：计数                                                                                                                                                |

| 指标名称                         | 可用于                  | 描述                             |
|------------------------------|----------------------|--------------------------------|
| write_count                  | Linux、Windows Server | 进程已执行的磁盘写入操作的数目。<br><br>单位：计数  |
| involuntary_context_switches | Linux                | 进程不自觉上下文切换的次数。<br><br>单位：计数    |
| voluntary_context_switches   | Linux                | 进程自觉上下文切换的次数。<br><br>单位：计数     |
| realtime_priority            | Linux                | 进程的实时优先级的当前使用率。<br><br>单位：计数   |
| nice_priority                | Linux                | 进程的良好状态优先级的当前使用率。<br><br>单位：计数 |
| signals_pending              | Linux                | 等待由进程处理的信号数量。<br><br>单位：计数     |
| rlimit_cpu_time_hard         | Linux                | 进程的硬 CPU 时间资源限制。<br><br>单位：计数  |
| rlimit_cpu_time_soft         | Linux                | 进程的软 CPU 时间资源限制。<br><br>单位：计数  |
| rlimit_file_locks_hard       | Linux                | 进程的硬文件锁资源限制。<br><br>单位：计数      |
| rlimit_file_locks_soft       | Linux                | 进程的软文件锁资源限制。<br><br>单位：计数      |
| rlimit_memory_data_hard      | Linux                | 进程上用于数据的内存的硬资源限制。<br><br>单位：字节 |

| 指标名称                                   | 可用于   | 描述                             |
|----------------------------------------|-------|--------------------------------|
| <code>rlimit_memory_data_soft</code>   | Linux | 进程上用于数据的内存的软资源限制。<br><br>单位：字节 |
| <code>rlimit_memory_locked_hard</code> | Linux | 进程上用于锁定的内存的硬资源限制。<br><br>单位：字节 |
| <code>rlimit_memory_locked_soft</code> | Linux | 进程上用于锁定的内存的软资源限制。<br><br>单位：字节 |
| <code>rlimit_memory_rss_hard</code>    | Linux | 进程上用于物理内存的硬资源限制。<br><br>单位：字节  |
| <code>rlimit_memory_rss_soft</code>    | Linux | 进程上用于物理内存的软资源限制。<br><br>单位：字节  |
| <code>rlimit_memory_stack_hard</code>  | Linux | 进程堆栈的硬资源限制。<br><br>单位：字节       |
| <code>rlimit_memory_stack_soft</code>  | Linux | 进程堆栈的软资源限制。<br><br>单位：字节       |
| <code>rlimit_memory_vms_hard</code>    | Linux | 进程上用于虚拟内存的硬资源限制。<br><br>单位：字节  |

## 查看由 CloudWatch 代理导入的进程指标

将进程指标导入到 CloudWatch 后，您可以以时间序列图形的形式查看这些指标，并创建可监视这些指标的告警，并在它们超出指定的阈值时通知您。以下程序介绍了如何以时间序列图形的方式查看进程指标。有关设置告警的更多信息，请参阅 [使用 Amazon CloudWatch 告警 \(p. 151\)](#)。

在 CloudWatch 控制台中查看进程指标

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。



3. 为该代理收集的指标选择命名空间。默认情况下，该命名空间为 CWAgent，但您可能已经在 CloudWatch 代理配置文件中指定了不同的命名空间。
4. 选择指标维度（例如 Per-Instance Metrics（每个实例的指标））。
5. All metrics 选项卡显示命名空间中该维度的所有指标。您可执行以下操作：
  - a. 要为指标绘制图表，请选中该指标旁的复选框。要选择所有指标，请选中表的标题行中的复选框。
  - b. 要对表进行排序，请使用列标题。
  - c. 要按资源进行筛选，请选择资源 ID，然后选择 Add to search。
  - d. 要按指标进行筛选，请选择指标名称，然后选择 Add to search。
6. （可选）要将此图表添加到 CloudWatch 控制面板，请选择 Actions（操作），然后选择 Add to dashboard（添加到控制面板）。

## 使用 StatsD 检索自定义指标

您可以将 CloudWatch 代理和 StatsD 协议结合使用，以从应用程序或服务中检索附加的自定义指标。StatsD 是一种流行的开源解决方案，可以从各种应用程序中收集指标。StatsD 对于检测您自己的指标特别有用。有关将 CloudWatch 代理和 StatsD 结合使用的示例，请参阅[如何使用 Amazon CloudWatch 代理更好地监控您的自定义应用程序指标](#)。

StatsD 在 Linux 服务器以及运行 Windows Server 的服务器上受支持。CloudWatch 支持以下 StatsD 格式：

```
MetricName:value|type|@sample_rate|#tag1:
value,tag1...
```

- MetricName – 一个没有冒号、竖线、# 字符或 @ 字符的字符串。
- value – 它可以是整数或浮点数。
- type – 指定 c 表示计数器，g 表示计量器，ms 表示计时器，h 表示直方图，s 表示集合。
- sample\_rate – （可选）一个介于 0 和 1 之间的浮点数（含）。仅用于计数器、直方图和计时器指标。默认值为 1（采样时间为 100%）。
- tags – （可选）一个逗号分隔的标签列表。StatsD 标签类似于 CloudWatch 中的维度。对键/值标签使用冒号，例如 env:prod。

您可以使用任何遵循此格式的 StatsD 客户端将指标发送到 CloudWatch 代理。有关一些可用 StatsD 客户端的更多信息，请参阅[GitHub 上的 StatsD 客户端页面](#)。

要收集这些自定义指标，请将 "statsd": {} 行添加到代理配置文件的 metrics\_collected 部分。您可以手动添加此行。如果您使用向导创建配置文件，向导会为您完成此操作。有关更多信息，请参阅[创建 CloudWatch 代理配置文件](#) (p. 682)。

StatsD 默认配置适用于大多数用户。您可以根据需要将三个可选字段添加到代理配置文件的 statsd 部分：

- service\_address – CloudWatch 代理应该侦听的服务地址。格式为 ip:port。如果您忽略了 IP 地址，则该代理将侦听所有可用接口。只有 UDP 格式受支持，因此您不需要指定 UDP 前缀。  
默认值为 :8125。
- metrics\_collection\_interval – StatsD 插件运行和收集指标的频率（秒）。默认值为 10 秒。范围为 1 – 172000。
- metrics\_aggregation\_interval – CloudWatch 将指标聚合为单个数据点的频率（秒）。默认值为 60 秒。

例如，如果 metrics\_collection\_interval 为 10，并且 metrics\_aggregation\_interval 为 60，则 CloudWatch 每 10 秒收集数据。在每分钟之后，来自该分钟的六个数据读数被聚合到单个数据点中，该数据点被发送到 CloudWatch。

范围为 1 – 172000。将 `metrics_aggregation_interval` 设置为 0 会禁用 StatsD 指标的聚合。

下面是代理配置文件的 `statsd` 部分的示例，使用默认端口、自定义集合和聚合间隔。

```
{
 "metrics":{
 "metrics_collected":{
 "statsd":{
 "service_address":":8125",
 "metrics_collection_interval":60,
 "metrics_aggregation_interval":300
 }
 }
 }
}
```

## 查看由 CloudWatch 代理导入的 StatsD 指标

将 StatsD 指标导入到 CloudWatch 后，您可以以时间序列图形的形式查看这些指标，并创建可监视这些指标的告警，并在它们超出指定的阈值时通知您。以下程序介绍了如何以时间序列图形的方式查看 StatsD 指标。有关设置告警的更多信息，请参阅 [使用 Amazon CloudWatch 告警 \(p. 151\)](#)。

在 CloudWatch 控制台中查看 StatsD 指标

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 为该代理收集的指标选择命名空间。默认情况下，该命名空间为 `CWAgent`，但您可能已经在 CloudWatch 代理配置文件中指定了不同的命名空间。
4. 选择指标维度（例如 `Per-Instance Metrics`（每个实例的指标））。
5. `All metrics` 选项卡显示命名空间中该维度的所有指标。您可执行以下操作：
  - a. 要为指标绘制图表，请选中该指标旁的复选框。要选择所有指标，请选中表的标题行中的复选框。
  - b. 要对表进行排序，请使用列标题。
  - c. 要按资源进行筛选，请选择资源 ID，然后选择 `Add to search`。
  - d. 要按指标进行筛选，请选择指标名称，然后选择 `Add to search`。
6. （可选）要将此图表添加到 CloudWatch 控制面板，请选择 `Actions`（操作），然后选择 `Add to dashboard`（添加到控制面板）。

## 使用 collectd 检索自定义指标

您可以使用 CloudWatch 代理以及 `collectd` 协议（该协议仅在 Linux 服务器上受支持）从应用程序或服务中检索其他指标。`collectd` 是一种通用的开源解决方案，其中的插件可以收集各种应用程序的系统统计数据。通过将 CloudWatch 代理可以收集的指标与 `collectd` 中的其他指标结合起来，您可以更好地监控、分析系统和应用程序并进行故障排除。有关 `collectd` 的更多信息，请参阅 [collectd - 系统统计数据收集守护程序](#)。

您可以使用 `collectd` 软件将指标发送到 CloudWatch 代理。对于 `collectd` 指标，CloudWatch 代理充当服务器，而 `collectd` 插件充当客户端。

`collectd` 软件不会自动安装在每个服务器上。在运行 Amazon Linux 2 的服务器上，请按照下列步骤安装 `collectd`

```
sudo amazon-linux-extras install collectd
```

有关在其他系统上安装 `collectd` 的信息，请参阅 [collectd 下载页面](#)。

要收集这些自定义指标，请将 "collectd": {} 行添加到代理配置文件的 metrics\_collected 部分。您可以手动添加此行。如果您使用向导创建配置文件，向导会为您完成此操作。有关更多信息，请参阅[创建 CloudWatch 代理配置文件 \(p. 682\)](#)。

还将提供可选参数。如果您使用的是 collectd 并且未将 /etc/collectd/auth\_file 用作 collectd\_auth\_file，则必须设置其中一些选项。

- service\_address：CloudWatch 代理应该侦听的服务地址。格式为 "udp://ip:port。默认为 udp://127.0.0.1:25826。
- name\_prefix：附加到每个 collectd 指标名称开头的前缀。默认为 collectd\_。最大长度为 255 个字符。
- collectd\_security\_level：设置网络通信的安全级别。默认值为 encrypt (加密)。

encrypt (加密) 指定只接受加密的数据。sign (签名) 指定只接受已签名且加密的数据。none (无) 指定接受所有数据。如果为 collectd\_auth\_file 指定值，则会尽可能解密加密数据。

有关更多信息，请参阅 collectd Wiki 中的[客户端设置](#)和[可能的交互](#)。

- collectd\_auth\_file 设置用户名映射到密码的文件。这些密码用于验证签名和解密已加密的网络数据包。如果给定，则会验证签名的数据并解密加密的数据包。否则，会接受签名的数据而不检查签名，并且加密的数据不能被解密。

默认为 /etc/collectd/auth\_file。

如果 collectd\_security\_level 设置为 none (无)，则这是可选的。如果将 collectd\_security\_level 设置为 encrypt 或 sign (签名)，则必须指定 collectd\_auth\_file。

对于身份验证文件的格式，每行是一个用户名，后跟一个冒号和任意数量的空格，再后跟密码。例如：

```
user1: user1_password
```

```
user2: user2_password
```

- collectd\_typesdb：包含数据集描述的一个或多个文件的列表。即使列表中只有一个条目，列表也必须用括号括起来。列表中的每个条目都必须用双引号括起来。如果有多个条目，请用逗号分隔。Linux 服务器上的默认值为 ["/usr/share/collectd/types.db"]。macOS 电脑上的默认值取决于 collectd 的版本。例如：["/usr/local/Cellar/collectd/5.12.0/share/collectd/types.db"]。

有关更多信息，请参阅<https://collectd.org/documentation/manpages/types.db.5.shtml>。

- metrics\_aggregation\_interval：CloudWatch 将指标聚合为单个数据点的频率（秒）。默认值为 60 秒。范围为 0 至 172,000。将它设置为 0 会禁用 collectd 指标的聚合。

下面是代理配置文件的 collectd 部分的示例。

```
{
 "metrics":{
 "metrics_collected":{
 "collectd":{
 "name_prefix":"My_collectd_metrics_",
 "metrics_aggregation_interval":120
 }
 }
 }
}
```

## 查看由 CloudWatch 代理导入的已收集指标

将 collected 指标导入到 CloudWatch 后，您可以以时间序列图形的形式查看这些指标，并创建可监视这些指标的告警，并在它们超出指定的阈值时通知您。以下程序介绍了如何以时间序列图形的方式查看 collected 指标。有关设置告警的更多信息，请参阅[使用 Amazon CloudWatch 告警 \(p. 151\)](#)。

在 CloudWatch 控制台中查看 collected 指标

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 为该代理收集的指标选择命名空间。默认情况下，该命名空间为 CWAgent，但您可能已经在 CloudWatch 代理配置文件中指定了不同的命名空间。
4. 选择指标维度（例如 Per-Instance Metrics（每个实例的指标））。
5. All metrics 选项卡显示命名空间中该维度的所有指标。您可执行以下操作：
  - a. 要为指标绘制图表，请选中该指标旁的复选框。要选择所有指标，请选中表的标题行中的复选框。
  - b. 要对表进行排序，请使用列标题。
  - c. 要按资源进行筛选，请选择资源 ID，然后选择 Add to search。
  - d. 要按指标进行筛选，请选择指标名称，然后选择 Add to search。
6. （可选）要将此图表添加到 CloudWatch 控制面板，请选择 Actions（操作），然后选择 Add to dashboard（添加到控制面板）。

## 在 Amazon EC2 实例上设置和配置 Prometheus 指标集合

以下部分介绍如何在 EC2 实例上安装具有 Prometheus 监控功能的 CloudWatch 代理，以及如何配置代理以抓取其他目标。它还提供了设置示例工作负载以使用 Prometheus 监控进行测试的教程。

有关 CloudWatch 代理支持的操作系统的信息，请参阅 [使用 CloudWatch 代理从 Amazon EC2 实例和本地部署服务器中收集指标和日志 \(p. 659\)](#)

VPC 安全组要求

如果您使用的是 VPC，则以下要求适用。

- Prometheus 工作负载的安全组的入口规则必须向 CloudWatch 代理打开 Prometheus 端口，以便通过私有 IP 抓取 Prometheus 指标。
- CloudWatch 代理的安全组的出口规则必须允许 CloudWatch 代理通过私有 IP 连接到 Prometheus 工作负载的端口。

主题

- [步骤 1：安装 CloudWatch 代理 \(p. 720\)](#)
- [步骤 2：抓取 Prometheus 源并导入指标 \(p. 720\)](#)
- [示例：为 Prometheus 指标测试设置 Java/JMX 示例工作负载 \(p. 724\)](#)

### 步骤 1：安装 CloudWatch 代理

第一步是在 EC2 实例上安装 CloudWatch 代理。有关说明，请参阅[安装 CloudWatch 代理 \(p. 660\)](#)。

### 步骤 2：抓取 Prometheus 源并导入指标

具有 Prometheus 监控功能的 CloudWatch 代理需要两种配置来抓取 Prometheus 指标。一种是 Prometheus 文档的 `<scrape_config>` 中记录的标准 Prometheus 配置。另一种配置是 CloudWatch 代理配置文件。

#### Prometheus 抓取配置

CloudWatch 代理支持标准的 Prometheus 抓取配置，如 Prometheus 文档中的 `<scrape_config>` 所述。您可以编辑此部分以更新此文件中已有的配置，并添加其他 Prometheus 抓取目标。示例配置文件包含以下全局配置行：

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
```

```
global:
 scrape_interval: 1m
 scrape_timeout: 10s
scrape_configs:
- job_name: MY_JOB
 sample_limit: 10000
 file_sd_configs:
 - files: ["C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\prometheus_sd_1.yaml", "C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\prometheus_sd_2.yaml"]
```

global 部分指定在所有配置上下文中有有效的参数。它们还用作其他配置部分的默认值。它包含以下参数：

- `scrape_interval` – 定义抓取目标的频率。
- `scrape_timeout` – 定义在抓取请求超时之前等待的时间。

scrape\_configs 部分指定了一组目标和参数，用于定义如何抓取它们。它包含以下参数：

- `job_name` – 默认情况下分配给已抓取指标的任务名称。
- `sample_limit` – 将被接受的抓取样本数量的每次抓取限制。
- `file_sd_configs` – 文件服务发现配置的列表。它读取一组包含零个或多个静态配置列表的文件。file\_sd\_configs 部分包含 `files` 参数，该参数定义从中提取目标组的文件的模式。

CloudWatch 代理支持以下服务发现配置类型。

**static\_config** 允许指定目标列表及其公用标签集。它是在抓取配置中指定静态目标的规范方法。

以下是用于从本地主机中抓取 Prometheus 指标的示例静态配置。如果 Prometheus 端口对运行代理的服务器打开，也可以从其他服务器中抓取指标。

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_sd_1.yaml
- targets:
 - 127.0.0.1:9404
 labels:
 key1: value1
 key2: value2
```

该示例包括以下参数：

- `targets` – 由静态配置抓取的目标。
- `labels` – 分配给从目标中抓取的所有指标的标签。

**ec2\_sd\_config** 允许从 Amazon EC2 实例中检索抓取目标。以下为示例 `ec2_sd_config` 从 EC2 实例列表中抓取 Prometheus 指标。这些实例的 Prometheus 端口必须向运行 CloudWatch 代理的服务器打开。运行 CloudWatch 代理的 EC2 实例的 IAM 角色必须包含 `ec2:DescribeInstance` 权限。例如，您可以将托管策略 `AmazonEC2ReadOnlyAccess` 附加到运行 CloudWatch 代理的实例。

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
global:
 scrape_interval: 1m
 scrape_timeout: 10s
scrape_configs:
- job_name: MY_JOB
 sample_limit: 10000
 ec2_sd_configs:
 - region: us-east-1
 port: 9404
 filters:
 - name: instance-id
```

```
values:
 - i-98765432109876543
 - i-12345678901234567
```

该示例包括以下参数：

- **region** – 目标 EC2 实例所在的 Amazon 区域。如果将此项保留空白，则使用实例元数据中的区域。
- **port** – 要从中抓取指标的端口。
- **filters** – 用于筛选实例列表的可选筛选条件。此示例基于 EC2 实例 ID 进行筛选。有关可以筛选的更多条件，请参阅 [DescribeInstances](#)。

## Prometheus 的 CloudWatch 代理配置

CloudWatch 代理配置文件包含 logs 和 metrics\_collected 下的 prometheus 部分。它包括以下参数。

- **cluster\_name** – 指定要在日志事件中添加为标签的集群名称。此字段为可选项。
- **log\_group\_name** – 指定已抓取 Prometheus 指标的日志组名称。
- **prometheus\_config\_path** – 指定 Prometheus 抓取配置文件路径。
- **emf\_processor** – 指定嵌入式指标格式处理器配置。有关嵌入式指标格式的更多信息，请参阅 [采用 CloudWatch 嵌入式指标格式摄取高基数日志以生成指标 \(p. 754\)](#)。

emf\_processor 部分可能包括以下参数：

- **metric\_declaration\_dedup** – 设置为 true，则启用嵌入式指标格式指标的重复数据消除功能。
- **metric\_namespace** – 指定发射的 CloudWatch 指标的指标命名空间。
- **metric\_unit** – 指定指标名称：指标单位映射。有关受支持指标单位的信息，请参阅 [MetricDatum](#)。
- **metric\_declaration** – 是指定要生成的采用嵌入式指标格式的日志数组的部分。默认情况下，CloudWatch 代理从中进行导入的每个 Prometheus 源都有 metric\_declaration 部分。这些部分各包括以下字段：
  - **source\_labels** 指定由 label\_matcher 行检查的标签的值。
  - **label\_matcher** 是一个正则表达式，用于检查 source\_labels 中列出的标签的值。匹配的指标将启用，以包含在发送到 CloudWatch 的嵌入式指标格式中。
  - **metric\_selectors** 是一个正则表达式，用于指定要收集并发送到 CloudWatch 的指标。
  - **dimensions** 是要用作每个选定指标的 CloudWatch 维度的标签列表。

以下是 Prometheus 的 CloudWatch 代理配置示例。

```
{
 "logs":{
 "metrics_collected":{
 "prometheus":{
 "cluster_name":"prometheus-cluster",
 "log_group_name":"Prometheus",
 "prometheus_config_path":"C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\
\\prometheus.yaml",
 "emf_processor":{
 "metric_declaration_dedup":true,
 "metric_namespace":"CWAgent-Prometheus",
 "metric_unit":{
 "jvm_threads_current": "Count",
 "jvm_gc_collection_seconds_sum": "Milliseconds"
 },
 "metric_declaration":[
 {
 "source_labels":[
 "job", "key2"
```

```
],
 "label_matcher": "MY_JOB;^value2",
 "dimensions": [
 [
 "key1", "key2"
],
 [
 "key2"
]
],
 "metric_selectors": [
 "^jvm_threads_current$",
 "^jvm_gc_collection_seconds_sum$"
]
 }
}
}
```

前一个示例配置嵌入式指标格式部分，以便在满足以下条件时作为日志事件发送：

- 标签 job 的值为 MY\_JOB
- 标签 key2 的值为 value2
- Prometheus 指标 jvm\_threads\_current 和 jvm\_gc\_collection\_seconds\_sum 同时包含 job 和 key2 标签。

发送的日志事件包括以下突出显示的部分。

```
{
 "CloudWatchMetrics": [
 {
 "Metrics": [
 {
 "Unit": "Count",
 "Name": "jvm_threads_current"
 },
 {
 "Unit": "Milliseconds",
 "Name": "jvm_gc_collection_seconds_sum"
 }
],
 "Dimensions": [
 [
 "key1",
 "key2"
],
 [
 "key2"
]
],
 "Namespace": "CWAgent-Prometheus"
 }
],
 "ClusterName": "prometheus-cluster",
 "InstanceId": "i-0e45bd06f196096c8",
 "Timestamp": "1607966368109",
 "Version": "0",
 "host": "EC2AMAZ-PDDOIUM",
 "instance": "127.0.0.1:9404",
```



```
"jvm_threads_current": 2,
"jvm_gc_collection_seconds_sum": 0.006000000000000002,
"prom_metric_type": "gauge",
...
}
```

## 示例：为 Prometheus 指标测试设置 Java/JMX 示例工作负载

JMX Exporter 是 Prometheus 的官方导出程序，可以将 JMX MBeans 作为 Prometheus 指标进行抓取和公开。有关详细信息，请参阅 [prometheus/jmx\\_exporter](#)。

CloudWatch 代理可以从 EC2 实例上的 JMX Exporter 的 Java 虚拟机 (JVM)、Hjava 和 Tomcat (Catalina) 中收集预定义的 Prometheus 指标。

### 步骤 1：安装 CloudWatch 代理

第一步是在 EC2 实例上安装 CloudWatch 代理。有关说明，请参阅 [安装 CloudWatch 代理 \(p. 660\)](#)。

### 步骤 2：启动 Java/JMX 工作负载

下一步是启动 Java/JMX 工作负载。

首先，从以下位置下载最新的 JMX Exporter jar 文件：[prometheus/jmx\\_exporter](#)。

为您的示例应用程序使用 jar

以下各部分中的示例命令使用 SampleJavaApplication-1.0-SNAPSHOT.jar 作为 jar 文件。将命令的这些部分替换为应用程序的 jar。

### 准备 JMX Exporter 配置

config.yaml 文件是 JMX Exporter 配置文件。有关更多信息，请参阅 JMX Exporter 文档中的 [配置](#)。

以下是 Java 和 Tomcat 的示例配置。

```

lowercaseOutputName: true
lowercaseOutputLabelNames: true

rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|ProcessCpuLoad|
OpenFileDescriptorCount|AvailableProcessors)'
 name: java_lang_OperatingSystem_$1
 type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
 name: java_lang_threading_$1
 type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
 name: catalina_globalrequestprocessor_$3_total
 labels:
 port: "$2"
 protocol: "$1"
 help: Catalina global $3
 type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=-_!|.:,;]*[-
a-zA-Z0-9+&@#/%?=-_!|.:,;]*), name=([-a-zA-Z0-9+&@#/%?=-_!|.:,;]*), J2EEApplication=none,
J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
 name: catalina_servlet_$3_total
 labels:
 module: "$1"
```



```
 servlet: "$2"
 help: Catalina servlet $3 total
 type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name="(\\w+-\\w+)-(\\d+)"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
 name: catalina_threadpool_$3
 labels:
 port: "$2"
 protocol: "$1"
 help: Catalina threadpool $3
 type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=-_!:.;])*[-a-zA-Z0-9+&@#/%?=-_!:.;]), context=([-a-zA-Z0-9+/%$_-!:.;])*><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
 name: catalina_session_$3_total
 labels:
 context: "$2"
 host: "$1"
 help: Catalina session $3 total
 type: COUNTER

- pattern: ".*"
```

### 使用 Prometheus 导出程序启动 Java 应用程序

启动示例应用程序。这会将 Prometheus 指标发送到端口 9404。请务必将入口点 `com.gubupt.sample.app.App` 替换为您的示例 java 应用程序的正确的信息。

在 Linux 操作系统上，输入以下命令。

```
$ nohup java -javaagent:./jmx_prometheus_javaagent-0.14.0.jar=9404:./config.yaml -cp ./
SampleJavaApplication-1.0-SNAPSHOT.jar com.gubupt.sample.app.App &
```

在 Windows 操作系统上，输入以下命令。

```
PS C:\> java -javaagent:.\jmx_prometheus_javaagent-0.14.0.jar=9404:.\config.yaml -cp .
\SampleJavaApplication-1.0-SNAPSHOT.jar com.gubupt.sample.app.App
```

### 验证 Prometheus 指标发射

验证是否正在发射 Prometheus 指标。

在 Linux 操作系统上，输入以下命令。

```
$ curl localhost:9404
```

在 Windows 操作系统上，输入以下命令。

```
PS C:\> curl http://localhost:9404
```

Linux 上的输出示例：

```
StatusCode : 200
StatusDescription : OK
Content : # HELP jvm_classes_loaded The number of classes that are currently
loaded in the JVM
 # TYPE jvm_classes_loaded gauge
 jvm_classes_loaded 2526.0
 # HELP jvm_classes_loaded_total The total number of class...
```

```
RawContent : HTTP/1.1 200 OK
 Content-Length: 71908
 Content-Type: text/plain; version=0.0.4; charset=utf-8
 Date: Fri, 18 Dec 2020 16:38:10 GMT

 # HELP jvm_classes_loaded The number of classes that are currentl...
Forms : {}
Headers : {[Content-Length, 71908], [Content-Type, text/plain; version=0.0.4;
 charset=utf-8], [Date, Fri, 18
 Dec 2020 16:38:10 GMT]}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : System.__ComObject
RawContentLength : 71908
```

### 步骤 3：配置 CloudWatch 代理以抓取 Prometheus 指标

接下来，在 CloudWatch 代理配置文件中设置 Prometheus 抓取配置。

为 Java/JMX 示例设置 Prometheus 抓取配置

1. 设置 file\_sd\_config 和 static\_config 配置。

在 Linux 操作系统上，输入以下命令。

```
$ cat /opt/aws/amazon-cloudwatch-agent/var/prometheus.yaml
global:
 scrape_interval: 1m
 scrape_timeout: 10s
scrape_configs:
 - job_name: jmx
 sample_limit: 10000
 file_sd_configs:
 - files: ["/opt/aws/amazon-cloudwatch-agent/var/prometheus_file_sd.yaml"]
```

在 Windows 操作系统上，输入以下命令。

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
global:
 scrape_interval: 1m
 scrape_timeout: 10s
scrape_configs:
 - job_name: jmx
 sample_limit: 10000
 file_sd_configs:
 - files: ["C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\
\\prometheus_file_sd.yaml"]
```

2. 设置抓取目标配置。

在 Linux 操作系统上，输入以下命令。

```
$ cat /opt/aws/amazon-cloudwatch-agent/var/prometheus_file_sd.yaml
- targets:
 - 127.0.0.1:9404
 labels:
 application: sample_java_app
 os: linux
```

在 Windows 操作系统上，输入以下命令。

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_file_sd.yaml
- targets:
 - 127.0.0.1:9404
 labels:
 application: sample_java_app
 os: windows
```

3. 通过 `ec2_sd_config` 设置 Prometheus 抓取配置。用正确的 EC2 实例 ID 替代 `your-ec2-instance-id`。

在 Linux 操作系统上，输入以下命令。

```
$ cat .\prometheus.yaml
global:
 scrape_interval: 1m
 scrape_timeout: 10s
scrape_configs:
 - job_name: jmx
 sample_limit: 10000
 ec2_sd_configs:
 - region: us-east-1
 port: 9404
 filters:
 - name: instance-id
 values:
 - your-ec2-instance-id
```

在 Windows 操作系统上，输入以下命令。

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_file_sd.yaml
- targets:
 - 127.0.0.1:9404
 labels:
 application: sample_java_app
 os: windows
```

4. 设置 CloudWatch 代理配置。首先，直接导航到正确的 ID。在 Linux 上，它是 `/opt/aws/amazon-cloudwatch-agent/var/cwagent-config.json`。在 Windows 上，它是 `C:\ProgramData\Amazon\AmazonCloudWatchAgent\cwagent-config.json`。

以下是定义了 Java/JMX Prometheus 指标的示例配置。请务必使用正确的路径替换 `path-to-Prometheus-Scrape-Configuration-file`。

```
{
 "agent": {
 "region": "us-east-1"
 },
 "logs": {
 "metrics_collected": {
 "prometheus": {
 "cluster_name": "my-cluster",
 "log_group_name": "prometheus-test",
 "prometheus_config_path": "path-to-Prometheus-Scrape-Configuration-file",
 "emf_processor": {
 "metric_declaration_dedup": true,
 "metric_namespace": "PrometheusTest",
 "metric_unit": {
 "jvm_threads_current": "Count",
 "jvm_classes_loaded": "Count",
 "java_lang_operatingsystem_freephysicalmemorysize": "Bytes",
 "catalina_manager_activesessions": "Count",
```

```
{
 "jvm_gc_collection_seconds_sum": "Seconds",
 "catalina_globalrequestprocessor_bytesreceived": "Bytes",
 "jvm_memory_bytes_used": "Bytes",
 "jvm_memory_pool_bytes_used": "Bytes"
},
"metric_declaration": [
 {
 "source_labels": ["job"],
 "label_matcher": "^jmx$",
 "dimensions": [["instance"]],
 "metric_selectors": [
 "^jvm_threads_current$",
 "^jvm_classes_loaded$",
 "^java_lang_operatingsystem_freephysicalmemorysize$",
 "^catalina_manager_activesessions$",
 "^jvm_gc_collection_seconds_sum$",
 "^catalina_globalrequestprocessor_bytesreceived$"
]
 },
 {
 "source_labels": ["job"],
 "label_matcher": "^jmx$",
 "dimensions": [["area"]],
 "metric_selectors": [
 "^jvm_memory_bytes_used$"
]
 },
 {
 "source_labels": ["job"],
 "label_matcher": "^jmx$",
 "dimensions": [["pool"]],
 "metric_selectors": [
 "^jvm_memory_pool_bytes_used$"
]
 }
]
},
"force_flush_interval": 5
}
```

5. 通过输入以下命令之一，重新启动 CloudWatch 代理。

在 Linux 操作系统上，输入以下命令。

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/var/cwagent-config.json
```

在 Windows 操作系统上，输入以下命令。

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m ec2 -s -c file:C:\ProgramData\Amazon\AmazonCloudWatchAgent\cwagent-config.json
```

## 查看 Prometheus 指标和日志

您现在可以查看正在收集的 Java/JMX 指标。

查看 Java/JMX 示例工作负载的指标

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在运行集群的区域中，选择左侧导航窗格中的 Metrics ( 指标 )。查找 PrometheusTest 命名空间以查看指标。
3. 要查看 CloudWatch Logs 事件，请在导航窗格中选择 Log groups ( 日志组 )。这些事件位于日志组 prometheus-test 中。

## CloudWatch 代理收集的指标

您可以在服务器上安装 CloudWatch 代理以从服务器中收集指标。您可以在 Amazon EC2 实例和本地部署服务器以及运行 Linux、Windows Server 或 macOS 的电脑上安装该代理。如果在 Amazon EC2 实例上安装该代理，则除了在 Amazon EC2 实例上默认启用的指标以外，它还会收集一些其他指标。

有关在实例上安装 CloudWatch 代理的信息，请参阅 [使用 CloudWatch 代理从 Amazon EC2 实例和本地部署服务器中收集指标和日志](#) (p. 659)。

本节中讨论的所有指标均由 CloudWatch 代理直接收集。

### Windows Server 实例上的 CloudWatch 代理收集的指标

通过在运行 Windows Server 的服务器上安装 CloudWatch 代理，您可以收集与 Windows 性能监视器中的计数器关联的指标。在创建这些计数器的 CloudWatch 指标名称时，将在对象名称和计数器名称之间添加空格。例如，在 CloudWatch 中为 Processor 对象的 % Interrupt Time 计数器指定指标名称 Processor % Interrupt Time。有关 Windows 性能监视器计数器的更多信息，请参阅 Microsoft Windows Server 文档。

CloudWatch 代理收集的指标的默认命名空间为 CWAgent，不过您可以在配置该代理时指定其他命名空间。

### Linux 和 macOS 实例上的 CloudWatch 代理收集的指标

下表列出了您可以使用 Linux 服务器和 macOS 电脑上的 CloudWatch 代理收集的指标。

| 指标                  | 描述                                                                     |
|---------------------|------------------------------------------------------------------------|
| cpu_time_active     | CPU 在任何容量中处于活动状态的时间。该指标以百分之一秒为单位。<br><br>单位：无                          |
| cpu_time_guest      | CPU 为来宾操作系统运行虚拟 CPU 的时间。该指标以百分之一秒为单位。<br><br>单位：无                      |
| cpu_time_guest_nice | CPU 为来宾操作系统运行虚拟 CPU 的时间，它具有较低的优先级，可能会被其他进程中断。该指标以百分之一秒为单位。<br><br>单位：无 |
| cpu_time_idle       | CPU 处于空闲状态的时间。该指标以百分之一秒为单位。                                            |

| 指标                   | 描述                                                           |
|----------------------|--------------------------------------------------------------|
|                      | 单位：无                                                         |
| cpu_time_iowait      | CPU 等待 I/O 操作完成的时间。该指标以百分之一秒为单位。<br>单位：无                     |
| cpu_time_irq         | CPU 处理中断的时间。该指标以百分之一秒为单位。<br>单位：无                            |
| cpu_time_nice        | CPU 处于用户模式的时间，它具有低优先级进程，可能很容易被高优先级进程中断。该指标以百分之一秒为单位。<br>单位：无 |
| cpu_time_softirq     | CPU 处理软件中断的时间。该指标以百分之一秒为单位。<br>单位：无                          |
| cpu_time_steal       | CPU 被盗的时间，这是在虚拟化环境中的其他操作系统上所花的时间。该指标以百分之一秒为单位。<br>单位：无       |
| cpu_time_system      | CPU 处于系统模式的时间。该指标以百分之一秒为单位。<br>单位：无                          |
| cpu_time_user        | CPU 处于用户模式的时间。该指标以百分之一秒为单位。<br>单位：无                          |
| cpu_usage_active     | CPU 在任何容量中处于活动状态的时间百分比。<br>单位：百分比                            |
| cpu_usage_guest      | CPU 为来宾操作系统运行虚拟 CPU 的时间百分比。<br>单位：百分比                        |
| cpu_usage_guest_nice | CPU 为来宾操作系统运行虚拟 CPU 的时间百分比，它具有较低的优先级，可能会被其他进程中断。<br>单位：百分比   |
| cpu_usage_idle       | CPU 空闲时间的百分比。<br>单位：百分比                                      |
| cpu_usage_iowait     | CPU 等待 I/O 操作完成的时间百分比。<br>单位：百分比                             |
| cpu_usage_irq        | CPU 处理中断的时间百分比。<br>单位：百分比                                    |

| 指标                                   | 描述                                                                  |
|--------------------------------------|---------------------------------------------------------------------|
| <code>cpu_usage_nice</code>          | CPU 处于用户模式的时间百分比，它具有低优先级进程，可能很容易被高优先级进程中断。<br>单位：百分比                |
| <code>cpu_usage_softirq</code>       | CPU 处理软件中断的时间百分比。<br>单位：百分比                                         |
| <code>cpu_usage_steal</code>         | CPU 被盗的时间百分比，或者在虚拟化环境中的其他操作系统上所花的时间。<br>单位：百分比                      |
| <code>cpu_usage_system</code>        | CPU 处于系统模式的时间百分比。<br>单位：百分比                                         |
| <code>cpu_usage_user</code>          | CPU 处于用户模式的时间百分比。<br>单位：百分比                                         |
| <code>disk_free</code>               | 磁盘上的可用空间。<br>单位：字节                                                  |
| <code>disk_inodes_free</code>        | 磁盘上的可用索引节点数。<br>单位：计数                                               |
| <code>disk_inodes_total</code>       | 在磁盘上保留的总索引节点数。<br>单位：计数                                             |
| <code>disk_inodes_used</code>        | 在磁盘上使用的索引节点数。<br>单位：计数                                              |
| <code>disk_total</code>              | 磁盘上的总空间，包括已使用的空间和可用空间。<br>单位：字节                                     |
| <code>disk_used</code>               | 磁盘上已使用的空间。<br>单位：字节                                                 |
| <code>disk_used_percent</code>       | 已使用总磁盘空间的百分比。<br>单位：百分比                                             |
| <code>diskio_iops_in_progress</code> | 已发出到设备驱动程序但尚未完成的 I/O 请求数。<br>单位：计数                                  |
| <code>diskio_io_time</code>          | 磁盘已将 I/O 请求排队的时间。<br>单位：毫秒<br><br>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。 |

| 指标                               | 描述                                                                                                                                                                                                      |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| diskio_reads                     | <p>磁盘读取操作数。</p> <p>单位：计数</p> <p>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。</p>                                                                                                                                   |
| diskio_read_bytes                | <p>从磁盘中读取的字节数。</p> <p>单位：字节</p> <p>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。</p>                                                                                                                                |
| diskio_read_time                 | <p>读取请求在磁盘上等待的时间。同时等待的多个读取请求会增加该数字。例如，如果 5 个请求均平均等待 100 毫秒，则报告 500。</p> <p>单位：毫秒</p> <p>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。</p>                                                                           |
| diskio_writes                    | <p>磁盘写入操作数。</p> <p>单位：计数</p> <p>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。</p>                                                                                                                                   |
| diskio_write_bytes               | <p>写入到磁盘的字节数。</p> <p>单位：字节</p> <p>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。</p>                                                                                                                                 |
| diskio_write_time                | <p>写入请求在磁盘上等待的时间。同时等待的多个写入请求会增加该数字。例如，如果 8 个请求均平均等待 1000 毫秒，则报告 8000。</p> <p>单位：毫秒</p> <p>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。</p>                                                                         |
| ethtool_bw_in_allowance_exceeded | <p>因入站聚合带宽超过实例的最大值而排队和/或丢弃的数据包的数量。</p> <p>仅当您将此指标列在 CloudWatch 代理配置文件 <code>metrics_collected</code> 部分的 <code>ethtool</code> 子部分时，才会收集该指标。有关更多信息，请参阅 <a href="#">收集网络性能指标 (p. 707)</a></p> <p>单位：无</p> |



| 指标                                                | 描述                                                                                                                                                                                                                                                                 |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ethtool_bw_out_allowance_exceeded</code>    | <p>因出站聚合带宽超过实例的最大值而排队和/或丢弃的数据包的数量。</p> <p>仅当您将此指标列在 CloudWatch 代理配置文件 <code>metrics_collected</code> 部分的 <code>ethtool</code> 子部分时，才会收集该指标。有关更多信息，请参阅 <a href="#">收集网络性能指标 (p. 707)</a></p> <p>单位：无</p>                                                            |
| <code>ethtool_conntrack_allowance_exceeded</code> | <p>由于连接跟踪超过实例的最大值且无法建立新连接而丢弃的数据包的数量。这可能会导致进出实例的流量丢失数据包。</p> <p>仅当您将此指标列在 CloudWatch 代理配置文件 <code>metrics_collected</code> 部分的 <code>ethtool</code> 子部分时，才会收集该指标。有关更多信息，请参阅 <a href="#">收集网络性能指标 (p. 707)</a></p> <p>单位：无</p>                                       |
| <code>ethtool_linklocal_allowance_exceeded</code> | <p>由于到本地代理服务的流量的 PPS 超出网络接口的最大值而丢弃的数据包数量。这会影响流向 DNS 服务、实例元数据服务和 Amazon Time Sync Service 的流量。</p> <p>仅当您将此指标列在 CloudWatch 代理配置文件 <code>metrics_collected</code> 部分的 <code>ethtool</code> 子部分时，才会收集该指标。有关更多信息，请参阅 <a href="#">收集网络性能指标 (p. 707)</a></p> <p>单位：无</p> |
| <code>ethtool_pps_allowance_exceeded</code>       | <p>因双向 PPS 超过实例的最大值而排队和/或丢弃的数据包的数量。</p> <p>仅当您将此指标列在 CloudWatch 代理配置文件 <code>metrics_collected</code> 部分的 <code>ethtool</code> 子部分时，才会收集该指标。有关更多信息，请参阅 <a href="#">收集网络性能指标 (p. 707)</a>。</p> <p>单位：无</p>                                                          |
| <code>mem_active</code>                           | <p>在上一抽样周期以某种方式使用的内存量。</p> <p>单位：字节</p>                                                                                                                                                                                                                            |
| <code>mem_available</code>                        | <p>可以立即分配给进程的可用内存量。</p> <p>单位：字节</p>                                                                                                                                                                                                                               |
| <code>mem_available_percent</code>                | <p>可以立即分配给进程的可用内存百分比。</p> <p>单位：百分比</p>                                                                                                                                                                                                                            |
| <code>mem_buffered</code>                         | <p>用作缓冲区的内存量。</p> <p>单位：字节</p>                                                                                                                                                                                                                                     |

| 指标                            | 描述                                                              |
|-------------------------------|-----------------------------------------------------------------|
| <code>mem_cached</code>       | 用作文件缓存的内存量。<br>单位：字节                                            |
| <code>mem_free</code>         | 未使用的内存量。<br>单位：字节                                               |
| <code>mem_inactive</code>     | 在上一采样周期未以某种方式使用的内存量<br>单位：字节                                    |
| <code>mem_total</code>        | 内存的总量。<br>单位：字节                                                 |
| <code>mem_used</code>         | 当前使用的内存量。<br>单位：字节                                              |
| <code>mem_used_percent</code> | 当前使用的内存百分比。<br>单位：百分比                                           |
| <code>net_bytes_recv</code>   | 网络接口接收的字节数。<br>单位：字节<br>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。       |
| <code>net_bytes_sent</code>   | 网络接口发送的字节数。<br>单位：字节<br>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。       |
| <code>net_drop_in</code>      | 该网络接口接收并已丢弃的数据包数。<br>单位：计数<br>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。 |
| <code>net_drop_out</code>     | 该网络接口发送并已丢弃的数据包数。<br>单位：计数<br>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。 |
| <code>net_err_in</code>       | 该网络接口检测到的接收错误数。<br>单位：计数<br>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。   |

| 指标                                   | 描述                                                                    |
|--------------------------------------|-----------------------------------------------------------------------|
| <code>net_err_out</code>             | 该网络接口检测到的发送错误数。<br><br>单位：计数<br><br>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。 |
| <code>net_packets_sent</code>        | 该网络接口发送的数据包数。<br><br>单位：计数<br><br>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。   |
| <code>net_packets_recv</code>        | 该网络接口接收的数据包数。<br><br>单位：计数<br><br>应该用于该指标的唯一统计数据是 Sum。请勿使用 Average。   |
| <code>netstat_tcp_close</code>       | 没有状态的 TCP 连接数。<br><br>单位：计数                                           |
| <code>netstat_tcp_close_wait</code>  | 等待来自客户端的终止请求的 TCP 连接数。<br><br>单位：计数                                   |
| <code>netstat_tcp_closing</code>     | 等待来自客户端的包含确认的终止请求的 TCP 连接数。<br><br>单位：计数                              |
| <code>netstat_tcp_established</code> | 已建立的 TCP 连接数。<br><br>单位：计数                                            |
| <code>netstat_tcp_fin_wait1</code>   | 在关闭连接过程中处于 FIN_WAIT1 状态的 TCP 连接数。<br><br>单位：计数                        |
| <code>netstat_tcp_fin_wait2</code>   | 在关闭连接过程中处于 FIN_WAIT2 状态的 TCP 连接数。<br><br>单位：计数                        |
| <code>netstat_tcp_last_ack</code>    | 等待客户端发送连接终止确认消息的 TCP 连接数。这是就在关闭连接之前的最后状态。<br><br>单位：计数                |
| <code>netstat_tcp_listen</code>      | 当前侦听连接请求的 TCP 端口数。<br><br>单位：计数                                       |
| <code>netstat_tcp_none</code>        | 具有不活动客户端的 TCP 连接数。<br><br>单位：计数                                       |

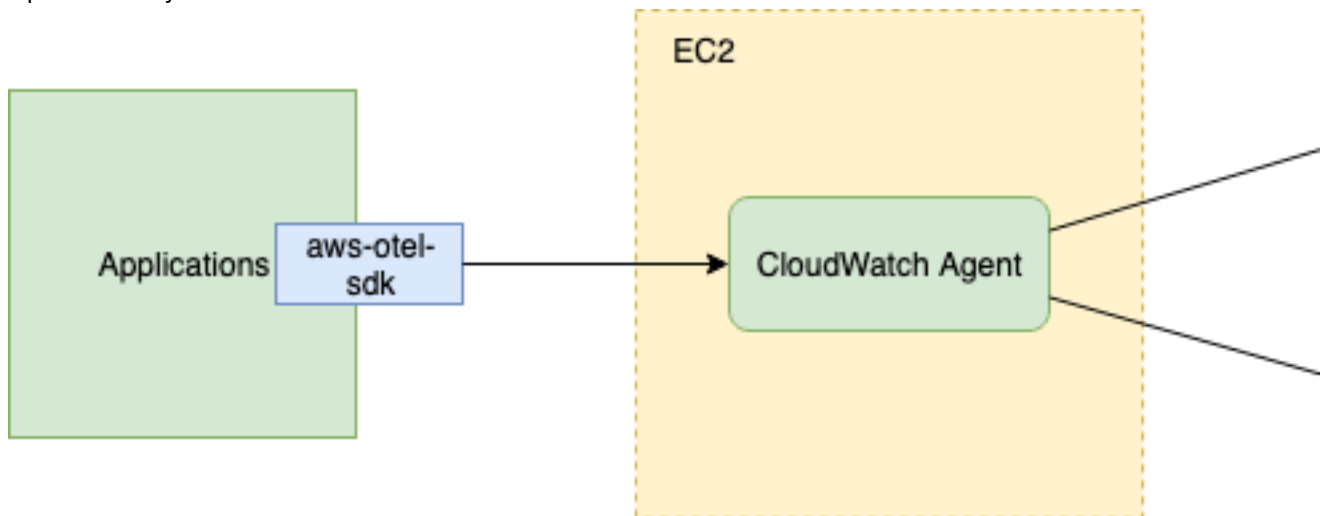
| 指标                                 | 描述                                                                                |
|------------------------------------|-----------------------------------------------------------------------------------|
| <code>netstat_tcp_syn_sent</code>  | 在发送连接请求后等待匹配连接请求的 TCP 连接数。<br>单位：计数                                               |
| <code>netstat_tcp_syn_recv</code>  | 在发送和接收连接请求后等待连接请求确认的 TCP 连接数。<br>单位：计数                                            |
| <code>netstat_tcp_time_wait</code> | 当前等待以确保客户端收到连接终止请求确认的 TCP 连接数。<br>单位：计数                                           |
| <code>netstat_udp_socket</code>    | 当前 UDP 连接数。<br>单位：计数                                                              |
| <code>processes_blocked</code>     | 已阻止的进程数。<br>单位：计数                                                                 |
| <code>processes_dead</code>        | 处于“僵死”状态的进程数，它是由 Linux 上的 <code>x</code> 状态代码表示的。<br>在 macOS 电脑上不会收集此指标。<br>单位：计数 |
| <code>processes_idle</code>        | 处于空闲状态 (睡眠超过 20 秒) 的进程数。仅适用于 FreeBSD 实例。<br>单位：计数                                 |
| <code>processes_paging</code>      | 处于“分页”状态的进程数，它是由 Linux 上的 <code>w</code> 状态代码表示的。<br>在 macOS 电脑上不会收集此指标。<br>单位：计数 |
| <code>processes_running</code>     | 正在运行的进程数，由 <code>R</code> 状态代码表示。<br>单位：计数                                        |
| <code>processes_sleeping</code>    | 睡眠的进程数，由 <code>s</code> 状态代码表示。<br>单位：计数                                          |
| <code>processes_stopped</code>     | 停止的进程数，由 <code>T</code> 状态代码表示。<br>单位：计数                                          |
| <code>processes_total</code>       | 实例上的总进程数。<br>单位：计数                                                                |

| 指标                                   | 描述                                                                                                        |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------|
| <code>processes_total_threads</code> | 组成进程的总线程数。该指标仅适用于 Linux 实例。<br><br>在 macOS 电脑上不会收集此指标。<br><br>单位：计数                                       |
| <code>processes_wait</code>          | 分页的进程数，它是由 FreeBSD 实例上的 w 状态代码表示的。此指标仅在 FreeBSD 实例上可用，在 Linux、Windows Server 或 macOS 实例上不可用。<br><br>单位：计数 |
| <code>processes_zombies</code>       | 僵尸进程数，由 z 状态代码表示。<br><br>单位：计数                                                                            |
| <code>swap_free</code>               | 未使用的交换空间量。<br><br>单位：字节                                                                                   |
| <code>swap_used</code>               | 当前已使用的交换空间量。<br><br>单位：字节                                                                                 |
| <code>swap_used_percent</code>       | 当前已使用的交换空间百分比。<br><br>单位：百分比                                                                              |

## CloudWatch 代理对 OpenTelemetry 的支持

版本 1.247347.3 和更高版本的 CloudWatch 代理具有嵌入式 Amazon OpenTelemetry Collector。这使得 CloudWatch 代理能够与 Amazon OpenTelemetry API 和软件开发工具包集成，并将应用程序遥测数据从 EC2 实例发送到 CloudWatch 和 Amazon X-Ray。此功能适用于希望在不安装或配置多个代理的情况下使用 OpenTelemetry 开始监控的现有 CloudWatch 代理用户。有关 Amazon OpenTelemetry 的更多信息，请参阅 [Amazon Distro for OpenTelemetry](#)。

通过使用 CloudWatch 代理和嵌入式 Amazon OpenTelemetry Collector，您不需要安装单独的 Amazon OpenTelemetry Collector。



具有嵌入式 OpenTelemetry Collector 的 CloudWatch 代理可以从 Amazon OpenTelemetry 软件开发工具包接收指标和追踪，并将其发布到 CloudWatch 和 X-Ray。嵌入在 CloudWatch 代理中的 OpenTelemetry Collector 具有与 Amazon OpenTelemetry Collector 相同的行为，使用它意味着您无需安装单独的 Amazon OpenTelemetry Collector。但是，如果您确实在同一台服务器上安装了两者的，请注意 CloudWatch 代理中的嵌入式 OpenTelemetry Collector 和 Amazon OpenTelemetry Collector 应位于不同的目录中，通过不同的工具进行管理，并作为单独的进程运行。例如，如果您在同一台服务器中配置并运行这两个进程，请确保它们用于侦听的本地端口不会相互冲突。

在 EC2 实例上运行的 CloudWatch 代理支持 CloudWatch 代理中的 OpenTelemetry，但不支持在容器中或本地部署服务器上运行的 CloudWatch 代理。Linux 实例支持 x86-64 和 ARM64 架构，Windows Server 实例支持 x86-64 架构。

## IAM 权限

为了能够发布 OpenTelemetry 指标和追踪，除了 CloudWatchAgentServerPolicy 托管式策略中列出的权限之外，CloudWatch 代理需要额外的 IAM 权限。在您希望使用代理的 OpenTelemetry 支持的服务器上，将以下策略授予给附加到实例的 CloudWatch 代理的 IAM 角色。如果您在文档中使用了默认的建议名称，则此角色名为 CloudWatchAgentServerRole。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "xray:PutTraceSegments",
 "xray:PutTelemetryRecords",
 "xray:GetSamplingRules",
 "xray:GetSamplingTargets",
 "xray:GetSamplingStatisticSummaries",
 "ssm:GetParameters"
],
 "Resource": "*"
 }
]
}
```

## CloudWatch 代理 OpenTelemetry 配置

亚马逊为嵌入式 OpenTelemetry Collector 提供默认配置文件。此配置允许通过默认端口收集 OpenTelemetry 指标和追踪。如果默认配置适用于您，则无需再执行任何配置步骤。要查看此默认配置文件的内容，请参阅 GitHub 上的 [config.yaml](#)。

如果要自定义配置文件，请参阅 [Amazon Distro for OpenTelemetry](#) 文档。如果执行此操作，则在创建自定义配置文件后，可以将其上传到 Parameter Store 中，也可以将其复制到要使用它的每个服务器的文件系统中。有关将其上传到 Parameter Store 的更多信息，请参阅 [将 CloudWatch 代理配置文件上传到 Systems Manager Parameter Store \(p. 707\)](#)。

## 使用命令行管理具有 OpenTelemetry 支持的 CloudWatch 代理

具有嵌入式 OpenTelemetry 支持的 CloudWatch 代理以两个进程运行：CloudWatch 代理进程和 OpenTelemetry Collector 进程。启动 CloudWatch 代理时，每个进程单独决定是否可以成功启动。

您可以运行 `amazon-cloudwatch-agent-ctl` 脚本来管理 OpenTelemetry Collector 进程。要配置和启动嵌入式 OpenTelemetry Collector，请运行以下命令之一：

Linux :

```
sudo /usr/bin/amazon-cloudwatch-agent-ctl -a fetch-config -o configuration-file -s
```

Windows Server :

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -o configuration-file -s
```

在这个命令中，`-a fetch-config` 加载 OpenTelemetry Collector 的配置。配置可能是默认配置，也可以是 Parameter Store 或本地文件中的自定义配置。`-s` 参数将使用此配置重新启动代理。

*configuration-file* 的值可以是 `default` 以使用默认的内置配置。要在 Parameter Store 中使用自定义配置，请指定 `ssm:your-configuration-parameter-name`。如果您将配置存储在本地文件中，请将其指定为 `file:your-configuration-file.yaml`。

例如，以下操作使用 OpenTelemetry Collector 的默认内置配置来配置并启动代理的 OpenTelemetry 进程。

```
sudo /usr/bin/amazon-cloudwatch-agent-ctl -a fetch-config -o default -s
```

### Important

使用 `-o` 选项配置代理的 OpenTelemetry Collector 进程，然后在配置 CloudWatch 代理流程时使用 `-c` 选项。以下命令是一个示例，使用各自的默认内置配置在 Linux 中启动嵌入式 OpenTelemetry Collector 和 CloudWatch 代理：

```
sudo /usr/bin/amazon-cloudwatch-agent-ctl -a fetch-config -o default -c default -s
```

## 使用命令行停止并启动 CloudWatch 代理中的 OpenTelemetry Collector

仅停止 OpenTelemetry Collector 进程

要停止代理中的 OpenTelemetry 进程，但让 CloudWatch 代理进程继续运行，您可以使用命令告诉代理移除 OpenTelemetry 配置，然后重新启动代理，并且不对 OpenTelemetry Collector 进行配置。

在以下命令中，`-a remove-config -o` 指示 CloudWatch 代理删除 OpenTelemetry Collector 的配置文件，`-s` 重新启动 OpenTelemetry Collector，而不使用能够有效地使其停止的任何配置。

Linux :

```
sudo /usr/bin/amazon-cloudwatch-agent-ctl -a remove-config -o configuration-file -s
```

Windows Server :

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a remove-config -o configuration-file -s
```

在此命令中，对于 *configuration-file*，您可以指定 `all` 来删除当前应用的任何配置。例如，以下命令将移除配置并停止 Linux 中已嵌入的 OpenTelemetry Collector：

```
sudo /usr/bin/amazon-cloudwatch-agent-ctl -a remove-config -o all -s
```

停止两个代理进程

您可以使用 `-a stop` 参数停止嵌入式 OpenTelemetry Collector 进程和 CloudWatch 代理进程（如果在运行中）。

Linux：

```
sudo /usr/bin/amazon-cloudwatch-agent-ctl -a stop
```

Windows Server：

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a stop
```

启动两个代理进程

您可以使用 `-a start` 参数启动嵌入式 OpenTelemetry Collector 进程和 CloudWatch 代理进程（如果已配置）。如果从未配置过 OpenTelemetry Collector，则使用此参数将不会启动它。如果您之前从未启动过 CloudWatch 代理进程，则此命令将使用默认配置启动该进程。

Linux：

```
sudo /usr/bin/amazon-cloudwatch-agent-ctl -a start
```

Windows Server：

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a start
```

## 使用 Systems Manager 通过嵌入式 OpenTelemetry Collector 管理 CloudWatch 代理

您还可以使用 Systems Manager 通过 OpenTelemetry Collector 管理 CloudWatch 代理。

使用 Run Command 配置并启动 CloudWatch 代理

1. 在 <https://console.aws.amazon.com/systems-manager/> 处打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

–或者–

如果打开了 Amazon Systems Manager 主页，请向下滚动并选择 Explore Run Command（浏览 Run Command）。

3. 选择 Run command（运行命令）。
4. 在命令文档列表中，选择 AmazonCloudWatch-ManageAgent。
5. 在 Targets（目标）区域中，选择安装了 CloudWatch 代理的实例。
6. 在操作列表中，选择配置。
7. 在 Optional OpenTelemetry Configuration Source（可选的 OpenTelemetry 配置源）列表中，如果希望使用自定义配置，选择 ssm；要使用默认配置，选择 default（默认）。
8. 在 Optional OpenTelemetry Configuration Location（可选的 OpenTelemetry 配置位置）框中，输入您创建并保存到 Systems Manager Parameter Store 的 OpenTelemetry 配置参数的名称，如前面的部分中所述。
9. （可选）如果要同时配置并启动 CloudWatch 代理进程，请执行以下操作：
  - a. 在 Optional Configuration Source（可选的配置源）列表中，如果希望使用自定义配置，选择 ssm；要使用默认配置，选择 default（默认）。



- b. 在 Optional Configuration Location ( 可选的配置位置 ) 框中, 输入您创建并保存到 Systems Manager Parameter Store 的 CloudWatch 代理配置参数的名称。
10. 在完成这些步骤后, 在可选的重新启动列表中选择是以启动该代理。
11. 选择 Run ( 运行 ) 。
12. ( 可选 ) 在 Targets and outputs ( 目标和输出 ) 区域中, 选择实例名称旁边的按钮, 然后选择 View output ( 查看输出 ) 。Systems Manager 将显示已成功启动该代理。

#### 删除配置并停止 OpenTelemetry Collector 进程

1. 在 <https://console.aws.amazon.com/systems-manager/> 处打开 Systems Manager 控制台。
2. 在导航窗格中, 选择 Run Command。

–或者–

如果打开了 Amazon Systems Manager 主页, 请向下滚动并选择 Explore Run Command ( 浏览 Run Command ) 。

3. 选择 Run command (运行命令)。
4. 在命令文档列表中, 选择 AmazonCloudWatch-ManagedAgent。
5. 在 Targets ( 目标 ) 区域中, 选择安装了 CloudWatch 代理的实例。
6. 在 Actions ( 操作 ) 列表中, 选择 Configure (remove) ( 配置 ( 移除 ) ) 。
7. 在 Optional OpenTelemetry ( 可选的配置源 ) 列表中, 选择 ssm。
8. 在完成这些步骤后, 在可选的重新启动列表中选择是以启动该代理。
9. 选择 Run ( 运行 ) 。
10. ( 可选 ) 在 Targets and outputs ( 目标和输出 ) 区域中, 选择实例名称旁边的按钮, 然后选择 View output ( 查看输出 ) 。Systems Manager 将显示已成功启动该代理。

## 生成 OpenTelemetry 的指标和跟踪

Amazon 为应用程序提供了 OpenTelemetry 软件开发工具包和 Java 自动分析代理, 用于生成 OpenTelemetry 指标和跟踪, 并将它们馈送给 OpenTelemetry Collector。有关更多信息, 请参阅下列内容:

- [追踪和指标分析的 Java 软件开发工具包入门](#)
- [使用适用于 OpenTelemetry Go 软件开发工具包的 Amazon Distro 进行追踪](#)
- [Python 软件开发工具包入门](#)
- [追踪和指标分析的 JavaScript 软件开发工具包入门](#)

## CloudWatch 代理的常见场景

以下几节简要说明在使用 CloudWatch 代理时如何完成一些常见的配置和自定义任务。

#### 主题

- [以不同用户身份运行 CloudWatch 代理 \(p. 742\)](#)
- [将自定义维度添加到 CloudWatch 代理收集的指标 \(p. 743\)](#)
- [多个 CloudWatch 代理配置文件 \(p. 743\)](#)
- [汇总或累积 CloudWatch 代理收集的指标 \(p. 745\)](#)
- [使用 CloudWatch 代理收集高精度指标 \(p. 745\)](#)
- [向不同账户发送指标和日志 \(p. 746\)](#)
- [统一的 CloudWatch 代理和更早的 CloudWatch Logs 代理之间的时间戳差异 \(p. 747\)](#)

## 以不同用户身份运行 CloudWatch 代理

在 Linux 服务器上，CloudWatch 默认以根用户身份运行。要让代理以不同用户身份运行，请在 CloudWatch 代理配置文件的 agent 部分中使用 run\_as\_user 参数。该选项仅在 Linux 服务器上可用。

如果已使用 root 用户运行代理并希望更改为使用不同的用户，请使用以下过程之一。

在运行 Linux 的 EC2 实例上以不同用户身份运行 CloudWatch 代理

1. 下载并安装新的 CloudWatch 代理软件包。有关更多信息，请参阅[下载 CloudWatch 代理软件包 \(p. 661\)](#)。
2. 创建新的 Linux 用户，或使用 RPM 或 DEB 文件创建的默认用户（名为 cwagent）。
3. 使用以下方式之一为该用户提供凭证：
  - 如果文件 .aws/credentials 存在于根用户的主目录中，您必须为要用于运行 CloudWatch 代理的用户创建一个凭证文件。该凭证文件是 /home/**username**/.aws/credentials。然后，将 common-config.toml 中的 shared\_credential\_file 参数值设置为凭证文件的路径名。有关更多信息，请参阅[\(可选\) 修改代理的通用配置或区域信息 \(p. 668\)](#)。
  - 如果根用户的主目录中不存在文件 .aws/credentials，您可以执行下列操作之一：
    - 为要用于运行 CloudWatch 代理的用户创建一个凭证文件。该凭证文件是 /home/**username**/.aws/credentials。然后，将 common-config.toml 中的 shared\_credential\_file 参数值设置为凭证文件的路径名。有关更多信息，请参阅[\(可选\) 修改代理的通用配置或区域信息 \(p. 668\)](#)。
    - 将 IAM 角色附加到实例，而不是创建凭证文件。代理将该角色作为凭证提供程序。
4. 在 CloudWatch 代理配置文件中，在 agent 部分中添加以下行：

```
"run_as_user": "username"
```

根据需要，对该配置文件进行其他修改。有关更多信息，请参阅[创建 CloudWatch 代理配置文件 \(p. 682\)](#)

5. 向用户提供必要的权限。用户必须拥有要收集的日志文件的 Read (r) 权限，并且必须对日志文件路径中的每个目录具有 Execute (x) 权限。
6. 使用刚修改的配置文件启动代理。

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:configuration-file-path
```

在运行 Linux 的本地部署服务器上以不同用户身份运行 CloudWatch 代理

1. 下载并安装新的 CloudWatch 代理软件包。有关更多信息，请参阅[下载 CloudWatch 代理软件包 \(p. 661\)](#)。
2. 创建新的 Linux 用户，或使用 RPM 或 DEB 文件创建的默认用户（名为 cwagent）。
3. 将该用户的凭证存储到用户可访问的路径中，例如 /home/**username**/.aws/credentials。
4. 将 common-config.toml 中的 shared\_credential\_file 参数值设置为凭证文件的路径名。有关更多信息，请参阅[\(可选\) 修改代理的通用配置或区域信息 \(p. 668\)](#)。
5. 在 CloudWatch 代理配置文件中，在 agent 部分中添加以下行：

```
"run_as_user": "username"
```

根据需要，对该配置文件进行其他修改。有关更多信息，请参阅[创建 CloudWatch 代理配置文件 \(p. 682\)](#)

6. 向用户提供必要的权限。用户必须拥有要收集的日志文件的 Read (r) 权限，并且必须对日志文件路径中的每个目录具有 Execute (x) 权限。
7. 使用刚修改的配置文件启动代理。

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:configuration-file-path
```

## 将自定义维度添加到 CloudWatch 代理收集的指标

要将自定义维度 (如标签) 添加到该代理收集的指标中，请将 `append_dimensions` 字段添加到代理配置文件中列出这些指标的部分。

例如，以下示例配置文件部分将一个名为 `stackName` 且值为 `Prod` 的自定义维度添加到该代理收集的 `cpu` 和 `disk` 指标中。

```
"cpu":{
 "resources":[
 "*"
],
 "measurement":[
 "cpu_usage_guest",
 "cpu_usage_nice",
 "cpu_usage_idle"
],
 "totalcpu":false,
 "append_dimensions":{
 "stackName":"Prod"
 }
},
"disk":{
 "resources":[
 "/",
 "/tmp"
],
 "measurement":[
 "total",
 "used"
],
 "append_dimensions":{
 "stackName":"Prod"
 }
}
```

切记，每次更改代理配置文件时，您必须重新启动该代理以使更改生效。

## 多个 CloudWatch 代理配置文件

您可以设置 CloudWatch 代理以使用多个配置文件。例如，您可以使用一个常见的配置文件，该文件收集您始终要从基础设施中的所有服务器收集的一组指标和日志。然后，您可以使用其他配置文件，这些配置文件从某些应用程序或在某些情况下收集指标。

要对此进行设置，请首先创建要使用的配置文件。将在同一台服务器上一同使用的任何配置文件必须具有不同的文件名。您可以将配置文件存储在服务器或 Parameter Store 中。

使用 `fetch-config` 选项启动 CloudWatch 代理，并指定第一个配置文件。要将第二个配置文件附加到正在运行的代理，请使用相同的命令，但使用 `append-config` 选项。将收集任一配置文件中列出的所有指标和日志。以下示例 Linux 命令使用配置存储作为文件来阐述此场景。第一行使用 `infrastructure.json` 配置文件启动代理，第二行附加 `app.json` 配置文件。

```
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s
-c file:/tmp/infrastructure.json
```

```
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a append-config -m ec2 -s
-c file:/tmp/app.json
```

以下示例配置文件展示了此功能的一个用法。第一个配置文件用于基础设施中的所有服务器，第二个配置文件仅收集来自特定应用程序的日志，并附加到运行该应用程序的服务器。

infrastructure.json

```
{
 "metrics": {
 "metrics_collected": {
 "cpu": {
 "resources": [
 "*"
],
 "measurement": [
 "usage_active"
],
 "totalcpu": true
 },
 "mem": {
 "measurement": [
 "used_percent"
]
 }
 }
 },
 "logs": {
 "logs_collected": {
 "files": {
 "collect_list": [
 {
 "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-
agent.log",
 "log_group_name": "amazon-cloudwatch-agent.log"
 },
 {
 "file_path": "/var/log/messages",
 "log_group_name": "/var/log/messages"
 }
]
 }
 }
 }
}
```

app.json

```
{
 "logs": {
 "logs_collected": {
 "files": {
 "collect_list": [
 {
 "file_path": "/app/app.log*",
 "log_group_name": "/app/app.log"
 }
]
 }
 }
 }
}
```

```
}
 }
}
}
```

附加到配置的任何配置文件彼此之间以及与初始配置文件之间都必须具有不同的文件名。如果您将 `append-config` 用于与代理已在使用的配置文件具有相同文件名的配置文件，则附加命令将覆盖第一个配置文件中的信息，而不是附加到它。即使这两个具有相同文件名的配置文件位于不同的文件路径中，也是这样。

上述示例显示使用了两个配置文件，但对您可以附加到代理配置的配置文件数量并没有限制。您还可以混合使用位于服务器上的配置文件和位于 Parameter Store 中的配置。

## 汇总或累积 CloudWatch 代理收集的指标

要汇总或累积该代理收集的指标，请将 `aggregation_dimensions` 字段添加到代理配置文件中与该指标对应的部分。

例如，以下配置文件片段累积 `AutoScalingGroupName` 维度上的指标。每个 Auto Scaling 组的所有实例中的指标将进行汇总，可以将这些指标作为一个整体进行查看。

```
"metrics": {
 "cpu": {...}
 "disk": {...}
 "aggregation_dimensions" : ["AutoScalingGroupName"]
}
```

除了累积 Auto Scaling 组名称以外，如果还希望累积各个 `InstanceId` 和 `InstanceType` 维度的组合，请添加以下内容。

```
"metrics": {
 "cpu": {...}
 "disk": {...}
 "aggregation_dimensions" : ["AutoScalingGroupName"], ["InstanceId", "InstanceType"]
}
```

要将指标累积到一个集合中，请使用 `[]`。

```
"metrics": {
 "cpu": {...}
 "disk": {...}
 "aggregation_dimensions" : [[]]
}
```

切记，每次更改代理配置文件时，您必须重新启动该代理以使更改生效。

## 使用 CloudWatch 代理收集高精度指标

`metrics_collection_interval` 字段指定收集的指标的时间间隔 (以秒为单位)。如果为该字段指定小于 60 的值，则将指标作为高精度指标进行收集。

例如，如果所有指标均应为高精度指标并且每 10 秒收集一次，请在 `agent` 部分中指定 10 作为 `metrics_collection_interval` 的值，以用作全局指标收集间隔。

```
"agent": {
 "metrics_collection_interval": 10
```

```
}
```

或者，以下示例将 cpu 指标设置为每秒收集一次，将所有其他指标设置为每分钟收集一次。

```
"agent":{
 "metrics_collection_interval": 60
},
"metrics":{
 "metrics_collected":{
 "cpu":{
 "resources":[
 "*"
],
 "measurement":[
 "cpu_usage_guest"
],
 "totalcpu":false,
 "metrics_collection_interval": 1
 },
 "disk":{
 "resources":[
 "/",
 "/tmp"
],
 "measurement":[
 "total",
 "used"
]
 }
 }
}
```

切记，每次更改代理配置文件时，您必须重新启动该代理以使更改生效。

## 向不同账户发送指标和日志

要让 CloudWatch 代理将指标和/或日志发送到其他账户，请在发送服务器上的代理配置文件中指定 `role_arn` 参数。`role_arn` 值指定在将数据发送到目标账户时代理使用的目标账户中的 IAM 角色。在指标或日志传递到目标账户时，此角色使发送账户能够在目标账户中担任相应的角色。

您还可以在代理配置文件中指定两个单独的 `role_arn` 字符串：一个用于发送指标，另一个用于发送日志。

以下配置文件的 `agent` 部分示例将代理设置为在将指标和日志发送到其他账户时使用 `CrossAccountAgentRole`。

```
{
 "agent": {
 "credentials": {
 "role_arn": "arn:aws:iam::123456789012:role/CrossAccountAgentRole"
 }
 },

}
```

或者，以下示例为发送账户设置不同的角色，以用于发送指标和日志：

```
"metrics": {
 "credentials": {
 "role_arn": "RoleToSendMetrics"
 },
}
```

```
"metrics_collected": {....
```

```
"logs": {
 "credentials": {
 "role_arn": "RoleToSendLogs"
 },

}
```

#### 必需策略

在代理配置文件中指定 `role_arn` 时，还必须确保发送和目标账户的 IAM 角色具有某些策略。发送账户和目标账户中的角色都应具有 `CloudWatchAgentServerPolicy`。有关将该策略分配给角色的更多信息，请参阅 [在 Amazon EC2 实例上创建要与 CloudWatch 代理一起使用的 IAM 角色 \(p. 670\)](#)。

发送账户中的角色还必须包含以下策略。编辑角色时，您可以将此策略添加到 IAM 控制台的 Permissions ( 权限 ) 选项卡。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "sts:AssumeRole"
],
 "Resource": [
 "arn:aws:iam::target-account-ID:role/agent-role-in-target-account"
]
 }
]
}
```

目标账户中的角色必须包含以下策略，以便它识别发送账户使用的 IAM 角色。编辑角色时，您可以将此策略添加到 IAM 控制台的 True relationships ( 信任关系 ) 选项卡。您添加此策略的目标账户中的角色是您在 [创建 IAM 角色和用户以用于 CloudWatch 代理 \(p. 663\)](#) 中创建的角色。此角色是在发送账户使用的策略的 `agent-role-in-target-account` 中指定的角色。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": [
 "arn:aws:iam::sending-account-ID:role/role-in-sender-account"
]
 },
 "Action": "sts:AssumeRole"
 }
]
}
```

## 统一的 CloudWatch 代理和更早的 CloudWatch Logs 代理之间的时间戳差异

与更早的 CloudWatch Logs 代理相比，CloudWatch 代理支持将一组不同的符号集用于时间戳格式。这些差异如下表所示。

| 两种代理均支持的符号                                                  | 仅统一 CloudWatch 代理支持的符号                    | 仅更早的 CloudWatch Logs 代理支持的符号              |
|-------------------------------------------------------------|-------------------------------------------|-------------------------------------------|
| %A、%a、%b、%B、%d、%f、%H、%I、%l、%m、%M、%p、%S、%s、%y、%Y、%Z、%b、%W、%w | %H、%I、%l、%m、%M、%p、%S、%s、%y、%Y、%Z、%b、%W、%w | %H、%I、%l、%m、%M、%p、%S、%s、%y、%Y、%Z、%b、%W、%w |

有关较新的 CloudWatch 代理支持的符号的含义的更多信息，请参阅 Amazon CloudWatch 用户指南中的 [CloudWatch 代理配置文件：日志部分](#)。有关 CloudWatch Logs 代理支持的符号的信息，请参阅 Amazon CloudWatch Logs 用户指南中的 [代理配置文件](#)。

## CloudWatch 代理故障排除

可以使用以下信息帮助解决 CloudWatch 代理问题。

### 主题

- [CloudWatch 代理命令行参数 \(p. 748\)](#)
- [使用 Run Command 安装 CloudWatch 代理失败 \(p. 748\)](#)
- [CloudWatch 代理无法启动 \(p. 749\)](#)
- [验证 CloudWatch 代理是否正在运行 \(p. 749\)](#)
- [CloudWatch 代理未启动，并且错误中提及了 Amazon EC2 区域 \(p. 750\)](#)
- [无法在 Windows Server 上启动 CloudWatch 代理 \(p. 750\)](#)
- [在 Windows Server 上找不到凭证 \(p. 750\)](#)
- [指标存储在何处？ \(p. 750\)](#)
- [CloudWatch 代理需要很长时间才能在容器中运行或记录跃点数限制错误 \(p. 751\)](#)
- [我更新了代理配置，但在 CloudWatch 控制台中看不到新的指标或日志 \(p. 751\)](#)
- [CloudWatch 代理文件和位置 \(p. 751\)](#)
- [CloudWatch 代理生成的日志 \(p. 752\)](#)
- [停止和重新启动 CloudWatch 代理 \(p. 753\)](#)

## CloudWatch 代理命令行参数

要查看 CloudWatch 代理支持的参数的完整列表，请在安装了此代理的电脑上的命令行处输入以下命令：

```
amazon-cloudwatch-agent-ctl -help
```

## 使用 Run Command 安装 CloudWatch 代理失败

要使用 Systems Manager Run Command 安装 CloudWatch 代理，目标服务器上的 SSM Agent 必须为 2.2.93.0 或更高版本。如果 SSM Agent 不是正确的版本，您可能会看到以下错误消息：

```
no latest version found for package AmazonCloudWatchAgent on platform linux
```

```
failed to download installation package reliably
```

有关更新 SSM Agent 版本的信息，请参阅 Amazon Systems Manager 用户指南中的 [安装和配置 SSM Agent](#)。



## CloudWatch 代理无法启动

如果 CloudWatch 代理无法启动，则您的配置可能存在问题。配置信息会记录到 `configuration-validation.log` 文件中。在 Linux 服务器上，该文件位于 `/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log` 中，在运行 Windows Server 的服务器上，该文件位于 `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log` 中。

## 验证 CloudWatch 代理是否正在运行

您可以查询 CloudWatch 代理以确定它是正在运行还是已停止。可使用 Amazon Systems Manager 远程执行此操作。也可以使用命令行，但仅用于检查本地服务器。

使用 Run Command 查询 CloudWatch 代理的状态

1. 在 <https://console.aws.amazon.com/systems-manager/> 处打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

–或者–

如果打开了 Amazon Systems Manager 主页，请向下滚动并选择 Explore Run Command (浏览 Run Command)。

3. 选择 Run command (运行命令)。
4. 在 Command document (命令文档) 列表中，选择 AmazonCloudWatch-ManageAgent 旁边的按钮。
5. 在操作列表中，选择状态。
6. 对于 Optional Configuration Source (可选的配置源)，选择 default (默认值) 并将 Optional Configuration Location (可选的配置位置) 保留为空。
7. 在目标区域，选择要检查的实例。
8. 选择 Run (运行)。

如果该代理正在运行，输出将类似于以下内容。

```
{
 "status": "running",
 "starttime": "2017-12-12T18:41:18",
 "version": "1.73.4"
}
```

如果该代理已停止，"status" 字段将显示 "stopped"。

使用命令行在本地查询 CloudWatch 代理的状态

- 在 Linux 服务器上，输入以下命令：

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a status
```

在运行 Windows Server 的服务器上，以管理员身份在 PowerShell 中输入以下命令：

```
& $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m ec2 -a status
```

## CloudWatch 代理未启动，并且错误中提及了 Amazon EC2 区域

如果代理将不会启动，并且错误消息中提及了 Amazon EC2 区域端点，则您可能已将代理配置为需要访问 Amazon EC2 端点但未授予该访问权限。

例如，如果您在代理配置文件中为 `append_dimensions` 参数指定一个值（取决于 Amazon EC2 元数据），并且使用代理，则必须确保服务器能够访问 Amazon EC2 的端点。有关这些端点的更多信息，请参阅 Amazon Web Services 一般参考中的 [Amazon Elastic Compute Cloud \(Amazon EC2\)](#)。

## 无法在 Windows Server 上启动 CloudWatch 代理

在 Windows Server 上，您可能会看到以下错误：

```
Start-Service : Service 'Amazon CloudWatch Agent (AmazonCloudWatchAgent)' cannot be started
due to the following
error: Cannot start service AmazonCloudWatchAgent on computer '.'.
At C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1:113
char:12
+ $svc | Start-Service
+ ~~~~~
+ CategoryInfo : OpenError:
(System.ServiceProcess.ServiceController:ServiceController) [Start-Service],
ServiceCommandException
+ FullyQualifiedErrorId :
CouldNotStartService,Microsoft.PowerShell.Commands.StartServiceCommand
```

要修复此问题，首先确保该服务器服务正在运行。如果代理在服务器服务未运行时尝试启动，则可能看到此错误。

如果服务器服务已在运行中，可能出现以下问题。在某些 Windows Server 安装中，CloudWatch 代理需要超过 30 秒才能启动。由于默认情况下，Windows Server 的服务启动超时只有 30 秒，因此这会导致代理失败，并出现类似于以下内容的错误：

要解决此问题，请增加服务超时值。有关详细信息，请参阅[服务无法启动](#)，并在 Windows 事件日志中记录事件 7000 和 7011。

## 在 Windows Server 上找不到凭证

在 Windows Server 上，如果凭证所在的位置不是 `$SystemDrive\\Users\\Administrator\\aws`（在 Windows Server 2012 上）或 `"$SystemDrive\\Documents and Settings\\Administrator\\aws`（在 Windows Server 2003 上），您可以在 `common.toml` 中使用 `shared_credential_file` 选项指定自己的凭证路径。

如果您没有凭证文件，则必须创建一个。有关更多信息，请参阅[（可选）修改代理的通用配置或区域信息](#)（p. 668）。

## 指标存储在何处？

如果 CloudWatch 代理已运行，但在 Amazon Web Services Management Console 或 Amazon CLI 中找不到收集的指标，请确认使用的是正确的命名空间。默认情况下，该代理收集的指标的命名空间为 `CWAgent`。您可以使用代理配置文件的 `metrics` 部分中的 `namespace` 字段自定义该命名空间。如果未看到所需的指标，请检查配置文件以确认正在使用该命名空间。

在首次下载 CloudWatch 代理软件包时，代理配置文件为 `amazon-cloudwatch-agent.json`。该文件位于运行配置向导的目录中，或者您可能已将其移到其他目录中。如果使用配置向导，该向导的代理配置文件

输出命名为 `config.json`。有关配置文件（包括 `namespace` 字段）的更多信息，请参阅 [CloudWatch 代理配置文件：Metrics（指标）部分](#) (p. 688)。

## CloudWatch 代理需要很长时间才能在容器中运行或记录跃点数限制错误

当您将在 CloudWatch 代理作为容器服务运行，并希望将 Amazon EC2 指标维度添加到代理收集的所有指标时，您可能在该代理的 v1.247354.0 版中看到以下错误：

```
2022-06-07T03:36:11Z E! [processors.ec2tagger] ec2tagger: Unable to retrieve Instance
Metadata Tags. This plugin must only be used on an EC2 instance.
2022-06-07T03:36:11Z E! [processors.ec2tagger] ec2tagger: Please increase hop limit to 2 by
following this document https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-
instance-metadata-options.html#configuring-IMDS-existing-instances.
2022-06-07T03:36:11Z E! [telegraf] Error running agent: could not initialize processor
ec2tagger: EC2MetadataRequestError: failed to get EC2 instance identity document
caused by: EC2MetadataError: failed to make EC2Metadata request
status code: 401, request id:
caused by:
```

如果代理尝试从容器内的 IMDSv2 获取元数据，而没有适当的跃点数限制，则可能会看到此错误。在 v1.247354.0 之前的代理版本中，您可能在没有看到日志消息的情况下遇到此问题。

要解决此问题，请按照[配置实例元数据选项](#)中的说明将跃点数限制增加到 2。

## 我更新了代理配置，但在 CloudWatch 控制台中看不到新的指标或日志

如果您更新 CloudWatch 代理配置文件，则下次启动代理时需要使用 **fetch-config** 选项。例如，如果您将更新的文件存储在本地计算机上，请输入以下命令：

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -s -m
ec2 -c file:configuration-file-path
```

## CloudWatch 代理文件和位置

下表列出了 CloudWatch 代理安装和使用的文件及其在运行 Linux 或 Windows Server 的服务器上的位置。

| 文件                                                     | Linux 位置                                                                                                                                                           | Windows Server 位置                                                                            |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| 控制代理的启动、停止和重新启动的控制脚本。                                  | <code>/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl</code> 或者 <code>/usr/bin/amazon-cloudwatch-agent-ctl</code>                                 | <code>\$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1</code> |
| 代理写入的日志文件。联系 Amazon Web Services Support 时，您可能需要附上此信息。 | <code>/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log</code> 或者 <code>/var/log/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.log</code> | <code>\$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log</code> |

| 文件                                                                                | Linux 位置                                                                                                                                   | Windows Server 位置                                                                |
|-----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| 代理配置验证文件。                                                                         | /opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log 或者 /var/log/amazon/amazon-cloudwatch-agent/configuration-validation.log | \$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log |
| 在向导创建后立即用于配置代理的 JSON 文件。有关更多信息，请参阅 <a href="#">创建 CloudWatch 代理配置文件</a> (p. 682)。 | /opt/aws/amazon-cloudwatch-agent/bin/config.json                                                                                           | \$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\config.json                      |
| 此 JSON 文件用于配置代理（如果此配置文件已从 Parameter Store 中下载）。                                   | /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json 或者 /etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json      | \$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json      |
| 用于指定代理使用的区域和凭证信息的 TOML 文件（覆盖系统默认值）。                                               | /opt/aws/amazon-cloudwatch-agent/etc/common-config.toml 或者 /etc/amazon/amazon-cloudwatch-agent/common-config.toml                          | \$Env:ProgramData\Amazon\AmazonCloudWatchAgent\common-config.toml                |

## CloudWatch 代理生成的日志

该代理在运行时生成一个日志。该日志包含故障排除信息。该日志是 amazon-cloudwatch-agent.log 文件。在 Linux 服务器上，该文件位于 /opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log 中，在运行 Windows Server 的服务器上，该文件位于 \$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log 中。

您可以将代理配置为在 amazon-cloudwatch-agent.log 文件中记录其他详细信息。在代理配置文件的 agent 部分中，将 debug 字段设置为 true，然后重新配置并重新启动 CloudWatch 代理。要禁用该额外信息的日志记录，请将 debug 字段设置为 false。然后重新配置并重启该代理。有关更多信息，请参阅[手动创建或编辑 CloudWatch 代理配置文件](#) (p. 686)。

在版本 1.247350.0 和更高版本的 CloudWatch 代理中，您可以选择将代理配置文件内 agent 部分中的 aws\_sdk\_log\_level 字段设置为以下一个或多个选项。使用 | 字符分隔多个选项。

- LogDebug
- LogDebugWithSigning
- LogDebugWithHTTPBody
- LogDebugRequestRetries
- LogDebugWithEventStreamBody

有关这些选项的更多信息，请参阅 [LogLevelType](#)。

## 停止和重新启动 CloudWatch 代理

您可以使用 Amazon Systems Manager 或命令行手动停止 CloudWatch 代理。

使用 Run Command 停止 CloudWatch 代理

1. 在 <https://console.aws.amazon.com/systems-manager/> 处打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

–或者–

如果打开了 Amazon Systems Manager 主页，请向下滚动并选择 Explore Run Command (浏览 Run Command)。

3. 选择 Run command (运行命令)。
4. 在命令文档列表中，选择 AmazonCloudWatch-ManageAgent。
5. 在 Targets (目标) 区域中，选择安装了 CloudWatch 代理的实例。
6. 在操作列表中，选择停止。
7. 将 Optional Configuration Source (可选的配置源) 和 Optional Configuration Location (可选的配置位置) 保留空白。
8. 选择 Run (运行)。

使用命令行在本地停止 CloudWatch 代理

- 在 Linux 服务器上，输入以下命令：

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a stop
```

在运行 Windows Server 的服务器上，以管理员身份在 PowerShell 中输入以下命令：

```
& $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m ec2 -a stop
```

要重新启动该代理，请按照[启动 CloudWatch 代理 \(p. 676\)](#)中的说明进行操作。

# 采用 CloudWatch 嵌入式指标格式摄取高基数日志以生成指标

CloudWatch 嵌入式指标格式使您能够以日志的形式摄取复杂的高基数应用程序数据，并通过这些数据生成富有实用价值的指标。您可以将自定义指标与详细的日志事件数据一同嵌入，CloudWatch 会自动摄取自定义指标，以便您可以对其进行可视化和设置告警，从而执行实时事件检测。此外，可以使用 CloudWatch Logs Insights 查询与摄取的指标相关的详细日志事件，以提供对操作事件根本原因的深入洞察。

嵌入式指标格式可帮助您通过临时资源（例如 Lambda 函数和容器）生成可操作的自定义指标。通过使用嵌入式指标格式发送来自这些临时资源的日志，您现在可以轻松创建自定义指标，而无需检测或维护单独的代码，同时获得对日志数据的强大分析功能。

在使用嵌入式指标格式时，您可以使用客户端库生成日志有关更多信息，请参阅 [使用客户端库生成嵌入式指标格式日志 \(p. 754\)](#)。或者，您可以手动构建日志并使用 PutLogEvents API 或 CloudWatch 代理提交该日志。

日志提取和存档以及生成的自定义指标会产生费用。有关更多信息，请参阅 [Amazon CloudWatch 定价](#)。

## Note

配置指标提取时请务必小心，因为它会影响自定义指标的使用和相应的账单。如果您无意中创建了基于高基数维度的指标（例如 `requestId`），则嵌入式指标格式将通过设计创建与每个唯一维度组合对应的自定义指标。有关更多信息，请参阅 [维度](#)。

## 主题

- [使用嵌入式指标格式生成日志 \(p. 754\)](#)
- [在控制台中查看您的指标和日志 \(p. 767\)](#)

## 使用嵌入式指标格式生成日志

您可以使用以下方法生成嵌入式指标格式日志：

- 使用开源客户端库生成和发送日志
- 手动生成日志，然后使用 CloudWatch 代理或 PutLogEvents API 发送日志

如果您使用手动方法之一，则日志必须遵循定义的 JSON 格式。

## 主题

- [使用客户端库生成嵌入式指标格式日志 \(p. 754\)](#)
- [通过 Amazon Distro for OpenTelemetry 使用嵌入式指标格式 \(p. 755\)](#)
- [手动生成嵌入式指标格式日志 \(p. 755\)](#)

## 使用客户端库生成嵌入式指标格式日志

Amazon 提供了开源客户端库，可供您用来创建嵌入式指标格式日志。目前，这些库提供以下列表中的语言版本。计划对其他语言的支持。

库及其使用说明位于 Github 上。请使用以下链接。

- [Node.js](#)
- [Python](#)
- [Java](#)
- [C#](#)

## 通过 Amazon Distro for OpenTelemetry 使用嵌入式指标格式

您可以将嵌入式指标格式用作 OpenTelemetry 项目的一部分。OpenTelemetry 是一项开源计划，通过提供一组规范和 API，消除供应商的跟踪、日志和指标特定格式之间的界限和限制。有关更多信息，请参阅 [OpenTelemetry](#)。

通过 OpenTelemetry 使用嵌入式指标格式需要两个组件：启用 OpenTelemetry 合规的数据源和 Amazon Distro for OpenTelemetry Collector，以与 CloudWatch 嵌入式指标格式日志一起使用。

我们已对由 Amazon 维护的 OpenTelemetry 组件的重新分配进行预先配置，以便尽可能简易地开展引导。有关使用带嵌入式指标格式的 OpenTelemetry 以及其他 Amazon 服务的更多信息，请参阅 [Amazon Distro for OpenTelemetry](#)。

有关语言支持和使用情况的其他信息，请参阅 [Github 上的 Amazon 观察](#)。

## 手动生成嵌入式指标格式日志

要发送手动创建的嵌入式指标格式日志，您可以使用 PutLogEvents API 或 CloudWatch 代理。

如果您使用这些方法之一，则必须遵循嵌入式指标格式规范。

### 主题

- [规范：嵌入式指标格式 \(p. 755\)](#)
- [使用 PutLogEvents API 发送手动创建的嵌入式指标格式日志 \(p. 760\)](#)
- [使用 CloudWatch 代理发送嵌入式指标格式日志 \(p. 762\)](#)

## 规范：嵌入式指标格式

CloudWatch 嵌入式指标格式是一个 JSON 规范，用于指示 CloudWatch Logs 自动提取结构化日志事件中嵌入的指标值。您可以使用 CloudWatch 基于提取的指标值绘制图表并创建告警。

### 嵌入式指标格式规范约定

此格式规范中的关键字“MUST”、“MUST NOT”、“REQUIRED”、“SHALL”、“SHALL NOT”、“SHOULD”、“SHOULD NOT”、“RECOMMENDED”、“MAY”和“OPTIONAL”的解释如[关键字 RFC2119](#)中所述。

此格式规范中的属于“JSON”、“JSON text”、“JSON value”、“member”、“element”、“object”、“array”、“number”、“string”、“boolean”、“true”、“false”和“null”的解释如[JavaScript 对象表示法 RFC8259](#)中所述。

### 嵌入式指标格式规范 PutLogEvents 请求格式

在使用 CloudWatch Logs [PutLogEvents](#) API 发送嵌入式指标格式文档时，客户端必须使用以下日志格式标头：



```
x-amzn-logs-format: json/emf
```

在 Lambda 上，您不需要自己设置此标题。以嵌入式指标格式将 JSON 写到标准输出就足够了。虽然 Lambda 可能会在日志事件前附加元数据（如时间戳和请求 ID），但此元数据后的有效嵌入式指标格式文档被视为有效。

## 嵌入式指标格式文档结构

此部分介绍了嵌入式指标格式文档的结构，该文档由日志格式标头 `x-amzn-logs-format: json/emf` 标识。嵌入式指标格式文档用 [JavaScript 对象表示法 RFC8259](#) 进行定义。

除非另有说明，否则此规范定义的对象不得包含任何其他成员。必须忽略此规范无法识别的成员。此规范中定义的成员是区分大小写的。

嵌入式指标格式受与标准 CloudWatch Logs 事件相同的限制，并且其最大大小限制为 256KB。

使用嵌入式指标格式，您可以按发布在您账户的 `Amazon/Logs` 命名空间中的指标跟踪 EMF 日志的处理。它们可用于从 EMF 跟踪失败的指标生成，以及失败是否由于解析或验证而发生。有关更多详细信息，请参阅 [使用 CloudWatch 指标监控](#)。

## 根节点

LogEvent 消息必须是一个有效的 JSON 对象，并且 LogEvent 消息字符串的开头或结尾没有其他数据。有关 LogEvent 结构的更多信息，请参阅 [InputLogEvent](#)。

嵌入式指标格式文档必须包含根节点上的以下顶级成员。这是一个 [元数据对象 \(p. 756\)](#) 对象。

```
{
 "_aws": {
 "CloudWatchMetrics": [...]
 }
}
```

根节点必须包含由 [MetricDirective 对象 \(p. 757\)](#) 中的引用定义的所有 [目标成员 \(p. 758\)](#) 成员。

根节点可以包含上述要求中未包含的任何其他成员。这些成员的值必须是有效的 JSON 类型。

## 元数据对象

`_aws` 成员可用于表示有关负载的元数据，告知下游服务应如何处理 LogEvent。该值必须是一个对象，并且必须包含以下成员：

- `CloudWatchMetrics` – 一个 [MetricDirective 对象 \(p. 757\)](#) 数组，用于指示 CloudWatch 从 LogEvent 的根节点提取指标。

```
{
 "_aws": {
 "CloudWatchMetrics": [...]
 }
}
```

- `Timestamp` – 一个数字，表示用于从事件中提取的指标的时间戳。值必须表示为自 1970 年 1 月 1 日 00:00:00 UTC 以来的毫秒数。

```
{
 "_aws": {
 "Timestamp": 1559748430481
 }
}
```



```
}
```

### MetricDirective 对象

MetricDirective 对象指示下游服务 LogEvent 包含将提取并发布到 CloudWatch 的指标。MetricDirectives 必须包含以下成员：

- 命名空间 – 一个表示指标的 CloudWatch 命名空间的字符串。
- 维度 – 一个 [DimensionSet 数组 \(p. 757\)](#)。
- 指标 – 一个 [MetricDefinition \(p. 757\)](#) 对象数组。此数组不得包含 100 个以上的 MetricDefinition 对象。

### DimensionSet 数组

DimensionSet 是一个字符串数组，其中包含将应用于文档中的所有指标的维度键。此数组中的值也必须是根节点上的成员 - 称为 [目标成员 \(p. 758\)](#)

DimensionSet 不得包含 30 个以上的维度键。DimensionSet 可以为空。

目标成员必须具有字符串值。此值不得包含 1024 个以上的字符。目标成员定义了将作为指标标识的一部分发布的维度。使用的每个 DimensionSet 都会在 CloudWatch 中创建一个新指标。有关维度的更多信息，请参阅[维度和维度](#)。

```
{
 "_aws": {
 "CloudWatchMetrics": [
 {
 "Dimensions": [["functionVersion"]],
 ...
 }
]
 },
 "functionVersion": "$LATEST"
}
```

#### Note

配置指标提取时请务必小心，因为它会影响自定义指标的使用和相应的账单。如果您无意中创建了基于高基数维度的指标（例如 requestId），则嵌入式指标格式将通过设计创建与每个唯一维度组合对应的自定义指标。有关更多信息，请参阅[维度](#)。

### MetricDefinition 对象

MetricDefinition 是一个必须包含以下成员的对象：

- 名称 – 指标 [目标成员 \(p. 758\)](#) 的字符串 [参考值 \(p. 758\)](#)。指标目标必须是数值或数值数组。

MetricDefinition 对象可以包含以下成员：

- 单位 – 一个可选字符串值，表示相应指标的度量单位。值应是有效的 CloudWatch 指标单位。有关有效单位的信息，请参阅 [MetricDatum](#)。如果未提供值，则假定默认值为 NONE。

```
{
 "_aws": {
 "CloudWatchMetrics": [
 {
 "Metrics": [
```

```
 {
 "Name": "Time",
 "Unit": "Milliseconds"
 },
 ...
]
 },
 "Time": 1
 }
}
```

## 参考值

参考值是引用根节点上的 [目标成员 \(p. 758\)](#) 成员的字符串值。这些引用不应与 [RFC6901](#) 中描述的 JSON 指针混淆。无法嵌套目标值。

## 目标成员

有效目标必须是根节点上的成员，而不能是嵌套对象。例如，“A.a”的参考值必须与以下成员匹配：

```
{ "A.a" }
```

它不得与嵌套成员匹配：

```
{ "A": { "a" } }
```

目标成员的有效值取决于引用它们的对象。指标目标必须是数值或数值数组。数字数组指标目标的成员不得超过 100 个。维度目标必须具有字符串值。

## 嵌入式指标格式示例和 JSON 架构

以下是嵌入式指标格式的有效示例。

```
{
 "_aws": {
 "Timestamp": 1574109732004,
 "CloudWatchMetrics": [
 {
 "Namespace": "lambda-function-metrics",
 "Dimensions": [["functionVersion"]],
 "Metrics": [
 {
 "Name": "time",
 "Unit": "Milliseconds"
 }
]
 }
]
 },
 "functionVersion": "$LATEST",
 "time": 100,
 "requestId": "989ffbf8-9ace-4817-a57c-e4dd734019ee"
}
```

您可以使用以下架构验证嵌入式指标格式文档。

```
{
 "type": "object",
```

```
"title": "Root Node",
"required": [
 "_aws"
],
"properties": {
 "_aws": {
 "$id": "#/properties/_aws",
 "type": "object",
 "title": "Metadata",
 "required": [
 "Timestamp",
 "CloudWatchMetrics"
],
 "properties": {
 "Timestamp": {
 "$id": "#/properties/_aws/properties/Timestamp",
 "type": "integer",
 "title": "The Timestamp Schema",
 "examples": [
 1565375354953
]
 },
 "CloudWatchMetrics": {
 "$id": "#/properties/_aws/properties/CloudWatchMetrics",
 "type": "array",
 "title": "MetricDirectives",
 "items": {
 "$id": "#/properties/_aws/properties/CloudWatchMetrics/items",
 "type": "object",
 "title": "MetricDirective",
 "required": [
 "Namespace",
 "Dimensions",
 "Metrics"
],
 "properties": {
 "Namespace": {
 "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/namespace",
 "type": "string",
 "title": "CloudWatch Metrics Namespace",
 "examples": [
 "MyApp"
],
 "pattern": "^(.*)$",
 "minLength": 1,
 "maxLength": 1024
 },
 "Dimensions": {
 "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Dimensions",
 "type": "array",
 "title": "The Dimensions Schema",
 "minItems": 1,
 "items": {
 "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Dimensions/items",
 "type": "array",
 "title": "DimensionSet",
 "minItems": 0,
 "maxItems": 30,
 "items": {
 "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Dimensions/items/items",
 "type": "string",
 "title": "DimensionReference",
```

```
 "examples": [
 "Operation"
],
 "pattern": "^(.*)$",
 "minLength": 1,
 "maxLength": 250
 }
}

 },
 "Metrics": {
 "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Metrics",
 "type": "array",
 "title": "MetricDefinitions",
 "items": {
 "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Metrics/items",
 "type": "object",
 "title": "MetricDefinition",
 "required": [
 "Name"
],
 "properties": {
 "Name": {
 "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/Name",
 "type": "string",
 "title": "MetricName",
 "examples": [
 "ProcessingLatency"
],
 "pattern": "^(.*)$",
 "minLength": 1,
 "maxLength": 1024
 },
 "Unit": {
 "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/Unit",
 "type": "string",
 "title": "MetricUnit",
 "examples": [
 "Milliseconds"
],
 "pattern": "^(Seconds|Microseconds|
Milliseconds|Bytes|Kilobytes|Megabytes|Gigabytes|Terabytes|Bits|Kilobits|Megabits|Gigabits|
Terabits|Percent|Count|Bytes\\//Second|Kilobytes\\//Second|Megabytes\\//Second|Gigabytes\\//
Second|Terabytes\\//Second|Bits\\//Second|Kilobits\\//Second|Megabits\\//Second|Gigabits\\//
Second|Terabits\\//Second|Count\\//Second|None)$"
 }
 }
 }
 }
}
}
```

## 使用 PutLogEvents API 发送手动创建的嵌入式指标格式日志

您可以使用 CloudWatch Logs PutLogEvents API 将嵌入式指标格式日志发送到 CloudWatch Logs。在调用 PutLogEvents 时，您需要包含以下 HTTP 标头以指示 CloudWatch Logs 应提取指标。

```
x-amzn-logs-format: json/emf
```

以下是使用 Amazon SDK for Java 2.x 的完整示例：

```
package org.example.basicapp;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.InputLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.PutLogEventsRequest;

import java.util.Collections;

public class EmbeddedMetricsExample {
 public static void main(String[] args) {

 final String usage = "To run this example, supply a Region code (eg. us-east-1), log group, and stream name as command line arguments"
 + "Ex: PutLogEvents <region-id> <log-group-name> <stream-name>";

 if (args.length != 3) {
 System.out.println(usage);
 System.exit(1);
 }

 String regionId = args[0];
 String logGroupName = args[1];
 String logStreamName = args[2];

 CloudWatchLogsClient logsClient =
 CloudWatchLogsClient.builder().region(Region.of(regionId)).build();

 // A sequence token is required to put a log event in an existing stream.
 // Look up the stream to find its sequence token.
 String sequenceToken = getNextSequenceToken(logsClient, logGroupName,
 logStreamName);

 // Build a JSON log using the EmbeddedMetricFormat.
 long timestamp = System.currentTimeMillis();
 String message = "{" +
 " \"_aws\": {" +
 " \"Timestamp\": " + timestamp + "," +
 " \"CloudWatchMetrics\": [{" +
 " {" +
 " \"Namespace\": \"MyApp\", " +
 " \"Dimensions\": [\"Operation\"], [\"Operation\",
 \"Cell\"]}, " +
 " \"Metrics\": [{ \"Name\": \"ProcessingLatency\",
 \"Unit\": \"Milliseconds\" }]}] " +
 " }" +
 " }" +
 " }, " +
 " \"Operation\": \"Aggregator\", " +
 " \"Cell\": \"001\", " +
 " \"ProcessingLatency\": 100 " +
 "}";

 InputLogEvent inputLogEvent = InputLogEvent.builder()
 .message(message)
 .timestamp(timestamp)
 .build();
```

```
// Specify the request parameters.
PutLogEventsRequest putLogEventsRequest = PutLogEventsRequest.builder()
 .overrideConfiguration(builder ->
 // provide the log-format header of json/emf
 builder.headers(Collections.singletonMap("x-amzn-logs-
format", Collections.singletonList("json/emf"))))
 .logEvents(Collections.singletonList(inputLogEvent))
 .logGroupName(logGroupName)
 .logStreamName(logStreamName)
 .sequenceToken(sequenceToken)
 .build();

logsClient.putLogEvents(putLogEventsRequest);

System.out.println("Successfully put CloudWatch log event");
}

private static String getNextSequenceToken(CloudWatchLogsClient logsClient, String
logGroupName, String logStreamName) {
 DescribeLogStreamsRequest logStreamRequest =
DescribeLogStreamsRequest.builder()
 .logGroupName(logGroupName)
 .logStreamNamePrefix(logStreamName)
 .build();

 DescribeLogStreamsResponse describeLogStreamsResponse =
logsClient.describeLogStreams(logStreamRequest);

 // Assume that a single stream is returned since a specific stream name was
 // specified in the previous request.
 return
describeLogStreamsResponse.logStreams().get(0).uploadSequenceToken();
}
}
```

#### Note

使用嵌入式指标格式，您可以按发布在您账户的 Amazon/Logs 命名空间中的指标跟踪 EMF 日志的处理。它们可用于从 EMF 跟踪失败的指标生成，以及失败是否由于解析或验证而发生。有关更多信息，请参阅[使用 CloudWatch 指标监控](#)。

## 使用 CloudWatch 代理发送嵌入式指标格式日志

要使用此方法，请先安装要从中发送嵌入式指标格式日志的服务的 CloudWatch 代理，然后才能开始发送事件。

CloudWatch 代理必须是 1.230621.0 版或更高版本。

#### Note

您无需安装 CloudWatch 代理即可通过 Lambda 函数发送日志。  
不会自动处理 Lambda 函数超时。这意味着，如果您的函数在指标刷新前超时，则不会捕获该调用的指标。

### 安装 CloudWatch 代理

为要将嵌入式指标格式日志发送到的每项服务安装 CloudWatch 代理。

#### 在 EC2 上安装 CloudWatch 代理

首先，在实例上安装 CloudWatch 代理。有关更多信息，请参阅[安装 CloudWatch 代理 \(p. 660\)](#)。

安装该代理后，请将它配置为侦听嵌入式指标格式日志的 UDP 或 TCP 端口。以下是侦听默认套接字 tcp:25888 的此配置的示例。有关代理配置的更多信息，请参阅 [手动创建或编辑 CloudWatch 代理配置文件 \(p. 686\)](#)。

```
{
 "logs": {
 "metrics_collected": {
 "emf": { }
 }
 }
}
```

### 在 Amazon ECS 上安装 CloudWatch 代理

在 Amazon ECS 上部署 CloudWatch 代理的最简单方法是将其作为附加项运行，并在与您的应用程序相同的任务定义中定义它。

#### 创建代理配置文件

本地创建 CloudWatch 代理配置文件。在此示例中，相对文件路径将是 amazon-cloudwatch-agent.json。

有关代理配置的更多信息，请参阅 [手动创建或编辑 CloudWatch 代理配置文件 \(p. 686\)](#)。

```
{
 "logs": {
 "metrics_collected": {
 "emf": { }
 }
 }
}
```

#### 将配置推送到 SSM Parameter Store

输入以下命令以将 CloudWatch 代理配置文件推送到 Amazon Systems Manager (SSM) Parameter Store。

```
aws ssm put-parameter \
 --name "cwagentconfig" \
 --type "String" \
 --value "`cat amazon-cloudwatch-agent.json`" \
 --region "${region}"
```

#### 配置任务定义

配置任务定义以使用 CloudWatch 代理并开放 TCP 或 UDP 端口。应使用的示例任务定义取决于您的联网模式。

请注意，webapp 指定了 AWS\_EMF\_AGENT\_ENDPOINT 环境变量。它将由库使用，并且应指向代理正在侦听的终端节点。此外，cwagent 指定 CW\_CONFIG\_CONTENT 作为“valueFrom”参数，该参数指向您在上一步中创建的 SSM 配置。

本节包含一个桥式模式示例和一个主机或 awsvpc 模式示例。有关如何在 Amazon ECS 上配置 CloudWatch 代理的更多示例，请参阅 [GitHub 示例存储库](#)

以下是桥接模式的示例。在启用桥式模式联网后，需要使用 links 参数将代理链接到您的应用程序，并且必须使用容器名称进行寻址。

```
{
 "containerDefinitions": [
 {
 "name": "webapp",
```

```
 "links": ["cwagent"],
 "image": "my-org/web-app:latest",
 "memory": 256,
 "cpu": 256,
 "environment": [{
 "name": "AWS_EMF_AGENT_ENDPOINT",
 "value": "tcp://cwagent:25888"
 }],
 },
 {
 "name": "cwagent",
 "mountPoints": [],
 "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
 "memory": 256,
 "cpu": 256,
 "portMappings": [{
 "protocol": "tcp",
 "containerPort": 25888
 }],
 "environment": [{
 "name": "CW_CONFIG_CONTENT",
 "valueFrom": "cwagentconfig"
 }],
 },
],
}
```

以下是主机模式或 awsvpc 模式的示例。在这些网络模式上运行时，可以通过 localhost 对代理进行寻址。

```
{
 "containerDefinitions": [
 {
 "name": "webapp",
 "image": "my-org/web-app:latest",
 "memory": 256,
 "cpu": 256,
 "environment": [{
 "name": "AWS_EMF_AGENT_ENDPOINT",
 "value": "tcp://127.0.0.1:25888"
 }],
 },
 {
 "name": "cwagent",
 "mountPoints": [],
 "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
 "memory": 256,
 "cpu": 256,
 "portMappings": [{
 "protocol": "tcp",
 "containerPort": 25888
 }],
 "environment": [{
 "name": "CW_CONFIG_CONTENT",
 "valueFrom": "cwagentconfig"
 }],
 },
],
}
```

#### Note

在 awsvpc 模式中，您必须为 VPC 提供公有 IP 地址（仅限 Fargate）、设置 NAT 网关或设置 CloudWatch Logs VPC 终端节点。有关设置 NAT 的更多信息，请参阅 [NAT 网关](#)。有关设置



CloudWatch Logs VPC 终端节点的更多信息，请参阅[将 CloudWatch Logs 与接口 VPC 终端节点结合使用](#)。

以下是如何将公有 IP 地址分配给使用 Fargate 启动类型的任务的示例。

```
aws ecs run-task \
--cluster {{cluster-name}} \
--task-definition cwagent-fargate \
--region {{region}} \
--launch-type FARGATE \
--network-configuration
 "awsvpcConfiguration={subnets=[{{subnetId}}],securityGroups=[{{sgId}}],assignPublicIp=ENABLED}"
```

#### 确保权限

确保执行任务的 IAM 角色具有从 SSM Parameter Store 中进行读取的权限。您可以通过附加 AmazonSSMReadOnlyAccess 策略来添加此权限。为此，请输入以下命令。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonSSMReadOnlyAccess \
--role-name CWAgentECSExecutionRole
```

#### 在 Amazon EKS 上安装 CloudWatch 代理

如果您已在此集群上安装 CloudWatch Container Insights，则可以跳过此过程的某些部分。

#### 权限

如果您尚未安装 Container Insights，请先确保您的 Amazon EKS 节点具有适当的 IAM 权限。它们应已附加 CloudWatchAgentServerPolicy。有关更多信息，请参阅[验证先决条件 \(p. 368\)](#)。

#### 创建 ConfigMap

为代理创建 ConfigMap。ConfigMap 还将告知代理侦听 TCP 或 UDP 端口。使用以下 ConfigMap。

```
cwagent-emf-configmap.yaml
apiVersion: v1
data:
 # Any changes here must not break the JSON format
 cwagentconfig.json: |
 {
 "agent": {
 "omit_hostname": true
 },
 "logs": {
 "metrics_collected": {
 "emf": { }
 }
 }
 }
kind: ConfigMap
metadata:
 name: cwagentemfconfig
 namespace: default
```

如果您已安装 Container Insights，请将以下 "emf": { } 行添加到现有 ConfigMap。

#### 应用 ConfigMap

输入以下命令可应用 ConfigMap。

```
kubectl apply -f cwagent-emf-configmap.yaml
```

## 部署代理

要将 CloudWatch 代理部署为附加项，请将代理添加到 pod 定义中，如下示例所示。

```
apiVersion: v1
kind: Pod
metadata:
 name: myapp
 namespace: default
spec:
 containers:
 # Your container definitions go here
 - name: web-app
 image: my-org/web-app:latest
 # CloudWatch Agent configuration
 - name: cloudwatch-agent
 image: public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest
 imagePullPolicy: Always
 resources:
 limits:
 cpu: 200m
 memory: 100Mi
 requests:
 cpu: 200m
 memory: 100Mi
 volumeMounts:
 - name: cwagentconfig
 mountPath: /etc/cwagentconfig
 ports:
 # this should match the port configured in the ConfigMap
 - protocol: TCP
 hostPort: 25888
 containerPort: 25888
 volumes:
 - name: cwagentconfig
 configMap:
 name: cwagentemfconfig
```

## 使用 CloudWatch 代理发送嵌入式指标格式日志

在安装并运行 CloudWatch 代理后，您可以通过 TCP 或 UDP 发送嵌入式指标格式日志。在通过代理发送日志时有两个要求：

- 日志必须包含一个 LogGroupName 键，该键告知代理要使用的日志组。
- 每个日志事件必须位于单个行上。换句话说，日志事件不能包含换行符 (\n)。

日志事件还必须遵循嵌入式指标格式规范。有关更多信息，请参阅[规范：嵌入式指标格式 \(p. 755\)](#)。

以下是手动从 Linux Bash Shell 发送日志事件的示例。您可以改用所选编程语言提供的 UDP 套接字接口。

```
echo '{"_aws":{"Timestamp":1574109732004,"LogGroupName":"Foo","CloudWatchMetrics":
[{"Namespace":"MyApp","Dimensions":[["Operation"]],"Metrics":
[{"Name":"ProcessingLatency","Unit":"Milliseconds"}]}]}',"Operation":"Aggregator","ProcessingLatency":10
\
> /dev/udp/0.0.0.0/25888
```

### Note

使用嵌入式指标格式，您可以按发布在您账户的 Amazon/Logs 命名空间中的指标跟踪 EMF 日志的处理。它们可用于从 EMF 跟踪失败的指标生成，以及失败是否由于解析或验证而发生。有关更多详细信息，请参阅[使用 CloudWatch 指标监控](#)。

## 在控制台中查看您的指标和日志

在生成用于提取指标的嵌入式指标格式日志后，可以使用 CloudWatch 控制台查看指标。嵌入式指标具有您在生成日志时指定的维度。此外，您使用客户端库生成的嵌入式指标将具有以下默认维度：

- ServiceType
- ServiceName
- LogGroup

查看从嵌入式指标格式日志生成的指标

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 选择在生成嵌入式指标时为其指定的命名空间。如果您使用客户端库生成指标，但未指定命名空间，请选择 `aws-embedded-metrics`。这是使用客户端库生成的嵌入式指标的默认命名空间。
4. 选择指标维度（例如，`ServiceName`）。
5. `All metrics` 选项卡显示命名空间中该维度的所有指标。您可执行以下操作：
  - a. 要对表进行排序，请使用列标题。
  - b. 要为指标绘制图表，请选中该指标旁的复选框。要选择所有指标，请选中表的标题行中的复选框。
  - c. 要按资源进行筛选，请选择资源 ID，然后选择 `Add to search`。
  - d. 要按指标进行筛选，请选择指标名称，然后选择 `Add to search`。

使用 CloudWatch Logs Insights 查询日志

您可以使用 CloudWatch Logs Insights 查询与提取的指标相关的详细日志事件，以提供对操作事件根本原因的深入洞察。从日志中提取指标的好处之一是，您可以稍后按独特指标（指标名称加独特维度集）和指标值筛选日志，以获取生成聚合指标值的事件的上下文。

例如，要获取受影响的请求 ID 或 X-ray 跟踪 ID，您可以在 CloudWatch Logs Insights 中运行以下查询。

```
filter Latency > 1000 and Operation = "Aggregator"
| fields RequestId, TraceId
```

您还可以对高基数键执行查询时间聚合，例如查找受事件影响的客户。以下示例对此进行了说明。

```
filter Latency > 1000 and Operation = "Aggregator"
| stats count() by CustomerId
```

有关更多信息，请参阅[使用 CloudWatch Logs Insights 分析日志数据](#)

# 发布 CloudWatch 指标的 Amazon 服务

以下 Amazon 服务将指标发布到 CloudWatch。有关指标和维度的信息，请参阅指定的文档。

| 服务                                 | 命名空间                   | 文档                                                           |
|------------------------------------|------------------------|--------------------------------------------------------------|
| Amazon Amplify                     | AWS/AmplifyHosting     | <a href="#">监控</a>                                           |
| Amazon API Gateway                 | AWS/ApiGateway         | <a href="#">使用 Amazon CloudWatch 监控 API 执行</a>               |
| Amazon App Runner                  | AWS/AppRunner          | <a href="#">查看报告给 CloudWatch 的 App Runner 服务指标</a>           |
| AppStream 2.0                      | AWS/AppStream          | <a href="#">监控 Amazon AppStream 2.0 资源</a>                   |
| Amazon AppSync                     | AWS/AppSync            | <a href="#">CloudWatch 指标</a>                                |
| Amazon Athena                      | AWS/Athena             | <a href="#">使用 CloudWatch 指标监控 Athena 查询</a>                 |
| Amazon Aurora                      | AWS/RDS                | <a href="#">Amazon Aurora 指标</a>                             |
| Amazon Backup                      | AWS/Backup             | <a href="#">使用 CloudWatch 监控 Amazon 备份指标</a>                 |
| Amazon Billing and Cost Management | AWS/Billing            | <a href="#">使用提醒和通知监控费用</a>                                  |
| Amazon Certificate Manager         | AWS/CertificateManager | <a href="#">支持的 CloudWatch 指标</a>                            |
| ACM Private CA                     | AWS/ACMPrivateCA       | <a href="#">支持的 CloudWatch 指标</a>                            |
| Amazon Chatbot                     | AWS/Chatbot            | <a href="#">使用 Amazon CloudWatch 监控 Amazon Chatbot</a>       |
| Amazon Chime SDK                   | AWS/ChimeSDK           | <a href="#">服务指标</a>                                         |
| Amazon Client VPN                  | AWS/ClientVPN          | <a href="#">使用 Amazon CloudWatch 进行监控</a>                    |
| Amazon CloudFront                  | AWS/CloudFront         | <a href="#">使用 CloudWatch 监控 CloudFront 活动</a>               |
| Amazon CloudHSM                    | AWS/CloudHSM           | <a href="#">获取 CloudWatch 指标</a>                             |
| Amazon CloudSearch                 | AWS/CloudSearch        | <a href="#">使用 Amazon CloudWatch 监控 Amazon CloudSearch 域</a> |
| Amazon CloudTrail                  | AWS/CloudTrail         | <a href="#">支持的 CloudWatch 指标</a>                            |
| CloudWatch 代理                      | CWAgent 或者自定义命名空间      | <a href="#">CloudWatch 代理收集的指标</a>                           |

| 服务                                | 命名空间                         | 文档                                                                               |
|-----------------------------------|------------------------------|----------------------------------------------------------------------------------|
| CloudWatch 指标流                    | AWS/CloudWatch/MetricStreams | 使用 <a href="#">CloudWatch 指标</a> 监控您的指标流                                         |
| CloudWatch RUM                    | AWS/RUM                      | 您可以使用 <a href="#">CloudWatch RUM</a> 收集的 <a href="#">CloudWatch 指标</a>           |
| CloudWatch Synthetics             | CloudWatchSynthetics         | <a href="#">金丝雀发布的 CloudWatch 指标</a>                                             |
| Amazon CloudWatch Logs            | AWS/Logs                     | 使用 <a href="#">CloudWatch 指标</a> 监控使用情况                                          |
| Amazon CodeBuild                  | AWS/CodeBuild                | 监控 <a href="#">Amazon CodeBuild</a>                                              |
| Amazon CodeGuru Profiler          | AWS/CodeGuruProfiler         | 使用 <a href="#">Amazon CloudWatch</a> 监控 <a href="#">Amazon CodeGuru Profiler</a> |
| Amazon Cognito                    | AWS/Cognito                  | 监控 <a href="#">Amazon Cognito</a>                                                |
| Amazon Connect                    | AWS/Connect                  | 使用 <a href="#">Amazon CloudWatch 指标</a> 监控 <a href="#">Amazon Connect</a>        |
| Amazon Data Lifecycle Manager     | AWS/DataLifecycleManager     | 使用 <a href="#">Amazon CloudWatch</a> 监控您的策略                                      |
| Amazon DataSync                   | AWS/DataSync                 | 监控您的任务                                                                           |
| Amazon Database Migration Service | AWS/DMS                      | 监控 <a href="#">Amazon DMS</a> 任务                                                 |
| Amazon Direct Connect             | AWS/DX                       | 使用 <a href="#">Amazon CloudWatch</a> 进行监控                                        |
| Amazon DocumentDB                 | AWS/DocDB                    | <a href="#">Amazon DocumentDB 指标</a>                                             |
| Amazon DynamoDB                   | AWS/DynamoDB                 | <a href="#">DynamoDB 指标与维度</a>                                                   |
| DynamoDB Accelerator (DAX)        | AWS/DAX                      | 查看 <a href="#">DAX 指标和维度</a>                                                     |
| Amazon EC2                        | AWS/EC2                      | 使用 <a href="#">CloudWatch</a> 监控您的实例                                             |
| Amazon EC2 Elastic Graphics       | AWS/ElasticGPUs              | 使用 <a href="#">CloudWatch 指标</a> 监控 <a href="#">Elastic Graphics</a>             |
| Amazon EC2 Spot 实例集               | AWS/EC2Spot                  | <a href="#">Spot 队列的 CloudWatch 指标</a>                                           |
| Amazon EC2 Auto Scaling           | AWS/AutoScaling              | 使用 <a href="#">CloudWatch</a> 监控您的 <a href="#">Auto Scaling</a> 组和实例             |
| Amazon Elastic Beanstalk          | AWS/ElasticBeanstalk         | 发布环境的 <a href="#">Amazon CloudWatch</a> 自定义指标                                    |
| Amazon Elastic Block Store        | AWS/EBS                      | <a href="#">Amazon EBS 的 Amazon CloudWatch 指标</a>                                |

| 服务                               | 命名空间                   | 文档                                                                   |
|----------------------------------|------------------------|----------------------------------------------------------------------|
| Amazon Elastic Container Service | AWS/ECS                | <a href="#">Amazon ECS CloudWatch 指标</a>                             |
| Amazon ECS 集群弹性伸缩                | AWS/ECS/ManagedScaling | <a href="#">Amazon ECS 集群自动扩展</a>                                    |
| Amazon Elastic File System       | AWS/EFS                | <a href="#">使用 CloudWatch 进行监控</a>                                   |
| Amazon Elastic Inference         | AWS/ElasticInference   | <a href="#">使用 CloudWatch 指标监控 Amazon EI</a>                         |
| 弹性负载均衡                           | AWS/ApplicationELB     | <a href="#">Application Load Balancer 的 CloudWatch 指标</a>            |
| 弹性负载均衡                           | AWS/NetworkELB         | <a href="#">Network Load Balancer 的 CloudWatch 指标</a>                |
| 弹性负载均衡                           | AWS/GatewayELB         | <a href="#">网关负载均衡器的 CloudWatch 指标</a>                               |
| 弹性负载均衡                           | AWS/ELB                | <a href="#">经典负载均衡器的 CloudWatch 指标</a>                               |
| Amazon Elastic Transcoder        | AWS/ElasticTranscoder  | <a href="#">使用 Amazon CloudWatch 进行监控</a>                            |
| Amazon ElastiCache for Memcached | AWS/ElastiCache        | <a href="#">使用 CloudWatch 指标监控使用情况</a>                               |
| Amazon ElastiCache for Redis     | AWS/ElastiCache        | <a href="#">使用 CloudWatch 指标监控使用情况</a>                               |
| Amazon OpenSearch Service        | AWS/ES                 | <a href="#">使用 Amazon CloudWatch 监控集群指标</a>                          |
| Amazon EMR                       | AWS/ElasticMapReduce   | <a href="#">使用 CloudWatch 监控指标</a>                                   |
| AWS Elemental MediaConnect       | AWS/MediaConnect       | <a href="#">使用 Amazon CloudWatch 监控 MediaConnect</a>                 |
| AWS Elemental MediaConvert       | AWS/MediaConvert       | <a href="#">使用 CloudWatch 指标查看 AWS Elemental MediaConvert 资源的指标</a>  |
| AWS Elemental MediaLive          | AWS/MediaLive          | <a href="#">使用 Amazon CloudWatch 指标监控活动</a>                          |
| AWS Elemental MediaPackage       | AWS/MediaPackage       | <a href="#">使用 Amazon CloudWatch 指标监控 AWS Elemental MediaPackage</a> |
| AWS Elemental MediaStore         | AWS/MediaStore         | <a href="#">使用 Amazon CloudWatch 指标监控 AWS Elemental MediaStore</a>   |
| AWS Elemental MediaTailor        | AWS/MediaTailor        | <a href="#">使用 Amazon CloudWatch 监控 AWS Elemental MediaTailor</a>    |
| Amazon EventBridge               | AWS/Events             | <a href="#">监控 Amazon EventBridge</a>                                |

| 服务                                      | 命名空间                  | 文档                                                                               |
|-----------------------------------------|-----------------------|----------------------------------------------------------------------------------|
| Amazon FSx for Lustre                   | AWS/FSx               | 监控 <a href="#">Amazon FSx for Lustre</a>                                         |
| Amazon FSx for Windows File Server      | AWS/FSx               | 监控 <a href="#">Amazon FSx for Windows File Server</a>                            |
| Amazon FSx for NetApp ONTAP             | AWS/FSx               | 使用 <a href="#">Amazon CloudWatch</a> 进行监控                                        |
| Amazon FSx for OpenZFS                  | AWS/FSx               | 使用 <a href="#">Amazon CloudWatch</a> 进行监控                                        |
| Amazon GameLift                         | AWS/GameLift          | 使用 <a href="#">CloudWatch</a> 监控 <a href="#">Amazon GameLift</a>                 |
| Amazon Global Accelerator               | AWS/GlobalAccelerator | 使用 <a href="#">Amazon CloudWatch</a> 与 <a href="#">Amazon Global Accelerator</a> |
| Amazon Glue                             | Glue                  | 使用 <a href="#">CloudWatch</a> 指标监控 <a href="#">Amazon Glue</a>                   |
| Amazon Ground Station                   | AWS/GroundStation     | 使用 <a href="#">Amazon CloudWatch</a> 的指标                                         |
| Amazon HealthLake                       | AWS/HealthLake        | 使用 <a href="#">CloudWatch</a> 监控 <a href="#">HealthLake</a>                      |
| Amazon Inspector                        | AWS/Inspector         | 使用 <a href="#">CloudWatch</a> 监控 <a href="#">Amazon Inspector</a>                |
| Amazon Interactive Video Service        | AWS/IVS               | 使用 <a href="#">Amazon CloudWatch</a> 监控 <a href="#">Amazon IVS</a>               |
| Amazon Interactive Video Service Chat   | AWS/IVSChat           | 使用 <a href="#">Amazon CloudWatch</a> 监控 <a href="#">Amazon IVS</a>               |
| Amazon IoT                              | AWS/IoT               | <a href="#">Amazon IoT</a> 指标与维度                                                 |
| Amazon IoT Analytics                    | AWS/IoTAnalytics      | 命名空间、指标和维度                                                                       |
| Amazon IoT SiteWise                     | AWS/IoTSiteWise       | 使用 <a href="#">Amazon CloudWatch</a> 指标监控 <a href="#">Amazon IoT SiteWise</a>    |
| Amazon IoT Things Graph                 | AWS/ThingsGraph       | 指标                                                                               |
| Amazon IoT TwinMaker                    | AWS/IoTTwinMaker      | 使用 <a href="#">Amazon CloudWatch</a> 指标监控 <a href="#">Amazon IoT TwinMaker</a>   |
| Amazon Key Management Service           | AWS/KMS               | 使用 <a href="#">CloudWatch</a> 进行监控                                               |
| Amazon Keyspaces (for Apache Cassandra) | AWS/Cassandra         | <a href="#">Amazon Keyspaces</a> 指标与维度                                           |

| 服务                                        | 命名空间                     | 文档                                                                                                                                                                                |
|-------------------------------------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Amazon Kinesis Data Analytics             | AWS/KinesisAnalytics     | 适用于 SQL 应用程序的 Kinesis Data Analytics : <a href="#">使用 CloudWatch 进行监控</a><br><br>Kinesis Data Analytics for Apache Flink : <a href="#">查看 Amazon Kinesis Data Analytics 指标与维度</a> |
| Amazon Kinesis Data Firehose              | AWS/Firehose             | <a href="#">使用 CloudWatch 指标监控 Kinesis Data Firehose</a>                                                                                                                          |
| Amazon Kinesis Data Streams               | AWS/Kinesis              | <a href="#">使用 Amazon CloudWatch 监控 Amazon Kinesis Data Streams</a>                                                                                                               |
| Amazon Kinesis Video Streams              | AWS/KinesisVideo         | <a href="#">使用 CloudWatch 监控 Kinesis Video Streams 指标</a>                                                                                                                         |
| Amazon Lambda                             | AWS/Lambda               | <a href="#">Amazon Lambda 指标</a>                                                                                                                                                  |
| Amazon Lex                                | AWS/Lex                  | <a href="#">使用 Amazon CloudWatch 监控 Amazon Lex</a>                                                                                                                                |
| Amazon Location Service                   | AWS/Location             | <a href="#">导出到 Amazon CloudWatch 的 Amazon Location Service 指标</a>                                                                                                                |
| Amazon Lookout for Metrics                | AWS/LookoutMetrics       | <a href="#">使用 Amazon CloudWatch 监控 Lookout for Metrics</a>                                                                                                                       |
| Amazon Machine Learning                   | AWS/ML                   | <a href="#">使用 CloudWatch 指标监控 Amazon ML</a>                                                                                                                                      |
| Amazon Managed Streaming for Apache Kafka | AWS/Kafka                | <a href="#">使用 Amazon CloudWatch 监控 Amazon MSK</a>                                                                                                                                |
| Amazon MQ                                 | AWS/AmazonMQ             | <a href="#">使用 Amazon CloudWatch 监控 Amazon MQ 代理</a>                                                                                                                              |
| Amazon Neptune                            | AWS/Neptune              | <a href="#">使用 CloudWatch 监控 Neptune</a>                                                                                                                                          |
| Amazon Network Firewall                   | AWS/NetworkFirewall      | <a href="#">Amazon CloudWatch 中的 Amazon Network Firewall 指标</a>                                                                                                                   |
| Amazon 网络管理器                              | AWS/NetworkManager       | <a href="#">适用于本地资源的 CloudWatch 指标</a>                                                                                                                                            |
| Amazon Nimble Studio                      | AWS/NimbleStudio         | <a href="#">使用 Amazon CloudWatch 监控 Nimble Studio</a>                                                                                                                             |
| Amazon OpsWorks                           | AWS/OpsWorks             | <a href="#">使用 Amazon CloudWatch 监控堆栈</a>                                                                                                                                         |
| Amazon Polly                              | AWS/Polly                | <a href="#">将 CloudWatch 与 Amazon Polly 集成</a>                                                                                                                                    |
| Amazon PrivateLink                        | AWS/PrivateLinkEndpoints | <a href="#">Amazon PrivateLink 的 CloudWatch 指标</a>                                                                                                                                |
| Amazon PrivateLink                        | AWS/PrivateLinkServices  | <a href="#">Amazon PrivateLink 的 CloudWatch 指标</a>                                                                                                                                |
| Amazon QLDB                               | AWS/QLDB                 | <a href="#">Amazon QLDB 指标与维度</a>                                                                                                                                                 |



| 服务                                 | 命名空间                                | 文档                                                                       |
|------------------------------------|-------------------------------------|--------------------------------------------------------------------------|
| Amazon QuickSight                  | AWS/QuickSight                      | 监控 <a href="#">Amazon QuickSight</a> 中的数据                                |
| Amazon Redshift                    | AWS/Redshift                        | <a href="#">Amazon Redshift</a> 性能数据                                     |
| Amazon Relational Database Service | AWS/RDS                             | 使用 <a href="#">Amazon CloudWatch</a> 监控 <a href="#">Amazon RDS</a> 指标    |
| Amazon RoboMaker                   | AWS/RoboMaker                       | 使用 <a href="#">Amazon CloudWatch</a> 监控 <a href="#">Amazon RoboMaker</a> |
| Amazon Route 53                    | AWS/Route53                         | 监控 <a href="#">Amazon Route 53</a>                                       |
| Route 53 应用程序恢复控制器                 | AWS/Route53RecoveryReadiness        | 将 <a href="#">Amazon CloudWatch</a> 与应用程序恢复控制器配合使用                       |
| Amazon SageMaker                   | AWS/SageMaker                       | 使用 <a href="#">CloudWatch</a> 监控 <a href="#">SageMaker</a>               |
| Amazon SageMaker 模型构建管道            | AWS/SageMaker/ModelBuildingPipeline | <a href="#">SageMaker</a> 管道指标                                           |
| Amazon Secrets Manager             | AWS/SecretsManager                  | 使用 <a href="#">Amazon CloudWatch</a> 监控 <a href="#">Secrets Manager</a>  |
| Amazon Service Catalog             | AWS/ServiceCatalog                  | <a href="#">Amazon Service Catalog</a> <a href="#">CloudWatch</a> 指标     |
| Amazon Shield Advanced             | AWS/DDoSProtection                  | 使用 <a href="#">CloudWatch</a> 进行监控                                       |
| Amazon Simple Email Service        | AWS/SES                             | 从 <a href="#">CloudWatch</a> 检索 <a href="#">Amazon SES</a> 事件数据          |
| Amazon Simple Notification Service | AWS/SNS                             | 使用 <a href="#">CloudWatch</a> 监控 <a href="#">Amazon SNS</a>              |
| Amazon Simple Queue Service        | AWS/SQS                             | 使用 <a href="#">CloudWatch</a> 监控 <a href="#">Amazon SQS</a> 队列           |
| Amazon S3                          | AWS/S3                              | 使用 <a href="#">Amazon CloudWatch</a> 监控指标                                |
| S3 Storage Lens                    | AWS/S3/Storage-Lens                 | 在 <a href="#">CloudWatch</a> 中监控 <a href="#">S3 Storage Lens</a> 指标      |
| Amazon Simple Workflow Service     | AWS/SWF                             | 用于 <a href="#">CloudWatch</a> 的 <a href="#">Amazon SWF</a> 指标            |
| Amazon Step Functions              | AWS/States                          | 使用 <a href="#">CloudWatch</a> 监控 <a href="#">Step Functions</a>          |
| Amazon Storage Gateway             | AWS/StorageGateway                  | 监控网关和资源                                                                  |
| Amazon Systems Manager Run Command | AWS/SSM-RunCommand                  | 使用 <a href="#">CloudWatch</a> 监控 <a href="#">Run Command</a> 指标          |

| 服务                       | 命名空间                                                             | 文档                                                            |
|--------------------------|------------------------------------------------------------------|---------------------------------------------------------------|
| Amazon Textract          | AWS/Texttract                                                    | <a href="#">Amazon Textract 的 CloudWatch 指标</a>               |
| Amazon Timestream        | AWS/Timestream                                                   | <a href="#">Timestream 指标与维度</a>                              |
| Amazon Transfer for SFTP | AWS/Transfer                                                     | <a href="#">Amazon SFTP CloudWatch 指标</a>                     |
| Amazon Translate         | AWS/Translate                                                    | <a href="#">Amazon Translate 的 CloudWatch 指标与维度</a>           |
| Amazon Trusted Advisor   | AWS/TrustedAdvisor                                               | <a href="#">使用 CloudWatch 创建 Trusted Advisor 警报</a>           |
| Amazon VPC               | AWS/NATGateway                                                   | <a href="#">使用 CloudWatch 监控 NAT 网关</a>                       |
| Amazon VPC               | AWS/TransitGateway                                               | <a href="#">Transit Gateway 的 CloudWatch 指标</a>               |
| Amazon VPC               | AWS/VPN                                                          | <a href="#">使用 CloudWatch 进行监控</a>                            |
| Amazon VPC IP 地址管理器      | AWS/IPAM                                                         | <a href="#">使用 Amazon CloudWatch 创建告警</a>                     |
| Amazon WAF               | 用于 Amazon WAF 资源的 AWS/WAFV2<br><br>用于 Amazon WAF Classic 资源的 WAF | <a href="#">使用 CloudWatch 进行监控</a>                            |
| Amazon WorkMail          | AWS/WorkMail                                                     | <a href="#">使用 Amazon CloudWatch 监控 Amazon WorkMail</a>       |
| Amazon WorkSpaces        | AWS/WorkSpaces                                                   | <a href="#">使用 CloudWatch 指标监控您的 WorkSpaces</a>               |
| Amazon WorkSpaces Web    | AWS/WorkSpacesWeb                                                | <a href="#">使用 Amazon CloudWatch 监控 Amazon WorkSpaces Web</a> |

# Amazon 使用情况指标

CloudWatch 会收集跟踪某些 Amazon 资源和 API 的使用情况的指标。这些指标发布在 `AWS/Usage` 命名空间中。CloudWatch 中的使用情况指标允许您通过在 CloudWatch 控制台中直观呈现指标、创建自定义控制面板、使用 CloudWatch 异常检测功能检测活动变化以及配置在使用量接近阈值时提示您的告警，主动管理使用情况。

一些 Amazon 服务将这些使用情况指标与 Service Quotas 集成。对于这些服务，您可以使用 CloudWatch 管理您的账户对 Service Quotas 的使用情况。有关更多信息，请参阅[可视化 Service Quotas 并设置告警 \(p. 775\)](#)。

## 主题

- [可视化 Service Quotas 并设置告警 \(p. 775\)](#)
- [Amazon API 使用情况指标 \(p. 776\)](#)
- [CloudWatch 使用情况指标 \(p. 781\)](#)

## 可视化 Service Quotas 并设置告警

对于一些 Amazon 服务，您可以使用使用情况指标在 CloudWatch 图表和控制面板上直观呈现当前服务使用情况。您可以使用 CloudWatch 指标数学函数在图表上显示这些资源的服务配额。还可以配置警报，以在用量接近服务配额时向您发出警报。有关 Service Quotas 的更多信息，请参阅 Service Quotas 用户指南中的[什么是 Service Quotas](#)。

目前，以下服务将其使用情况指标与 Service Quotas 集成：

- [Amazon CloudHSM](#)
- [Amazon Chime 开发工具包](#)
- [Amazon CloudWatch](#)
- [Amazon CloudWatch Logs](#)
- [Amazon DynamoDB](#)
- [Amazon EC2](#)
- [Amazon Elastic Container Registry](#)
- [Amazon Fargate](#)
- [Amazon Fault Injection Simulator](#)
- [Amazon Interactive Video Service](#)
- [Amazon Key Management Service](#)
- [Amazon Kinesis Data Firehose](#)
- [Amazon Location Service](#)
- [Amazon Robomaker](#)

## 可视化服务配额并选择性地设置警报

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 在 All metrics (所有指标) 选项卡上，选择 Usage (使用情况)，然后选择 By Amazon Resource (按亚马逊云科技资源)。

这将显示服务配额用量指标的列表。

4. 选中其中一个指标旁边的复选框。

该图表显示您该 Amazon 资源的当前用量。

5. 要将服务配额添加到图表，请执行以下操作：
- a. 选择 Graphed metrics ( 绘制的指标 ) 选项卡。
  - b. 选择数学表达式、从空表达式开始。在新行中，在 Details ( 详细信息 ) 下，输入 **SERVICE\_QUOTA(m1)**。
- 这将向图表中添加一个新行，并显示指标中表示的资源的服务配额。
6. 要以配额百分比的形式查看您的当前用量，请添加新的表达式或更改当前 SERVICE\_QUOTA 表达式。使用的新表达式是 **m1/SERVICE\_QUOTA(m1)\*100**。
7. (可选) 要设置一个警报，以便在接近服务配额时向您发送通知，请执行以下操作：
- a. 在 **m1/SERVICE\_QUOTA(m1)\*100** 行上的 Action ( 操作 ) 下，选择告警图标。该图标看起来像一个铃铛。
- 这将显示警报创建页面。
- b. 在条件下，确保阈值类型为静态，并将当 Expression1 为设置为大于。在多于下，输入 **80**。这将创建一个警报，当用量超过配额的 80% 时，该警报将进入 ALARM 状态。
  - c. 选择 Next ( 下一步 )。
  - d. 在下一页上，选择 Amazon SNS 主题或创建一个新主题，然后选择 Next ( 下一步 )。当告警进入 ALARM 状态时，会向此主题发送通知。
  - e. 在下一页上，输入警报的名称和描述，然后选择下一步。
  - f. 选择创建警报。

## Amazon API 使用情况指标

大多数支持 Amazon CloudTrail 日志记录的 API 还会向 CloudWatch 报告使用情况指标。CloudWatch 中的 API 使用情况指标允许您通过在 CloudWatch 控制台中直观呈现指标、创建自定义控制面板、使用 CloudWatch 异常检测功能检测活动变化以及配置在使用量接近阈值时提示您的告警，主动管理 API 使用情况。

下表列出了向 CloudWatch 报告 API 使用情况指标的服务，以及 Service 维度用于查看该服务的使用情况指标的值。

| 服务                                                    | Service 维度的值                  |
|-------------------------------------------------------|-------------------------------|
| Amazon Identity and Access Management Access Analyzer | Access Analyzer               |
| Amazon Account Management                             | Account Management            |
| Alexa for Business                                    | A4B                           |
| Amazon API Gateway                                    | API Gateway                   |
| Amazon App Mesh                                       | App Mesh                      |
| Amazon AppConfig                                      | AWS AppConfig                 |
| Amazon AppFlow                                        | AppFlow                       |
| Application Discovery Service                         | Application Discovery Service |
| Amazon AppStream                                      | AppStream                     |

| 服务                                     | Service 维度的值                    |
|----------------------------------------|---------------------------------|
| AppStream 2.0 Image Builder            | Image Builder                   |
| Amazon Athena                          | Athena                          |
| Amazon Audit Manager                   | Audit Manager                   |
| Amazon Backup                          | Backup                          |
| Amazon Batch                           | Batch                           |
| Amazon Braket                          | Braket                          |
| Amazon Budgets                         | Budgets                         |
| Amazon Certificate Manager             | Certificate Manager             |
| Amazon Chime 开发工具包                     | ChimeSDK                        |
| Amazon Cloud Directory                 | Cloud Directory                 |
| Amazon Cloud Map                       | Cloud Map                       |
| Amazon CloudFormation                  | CloudFormation                  |
| Amazon CloudHSM                        | CloudHSM                        |
| Amazon CloudSearch                     | CloudSearch                     |
| Amazon CloudShell                      | CloudShell                      |
| Amazon CloudTrail                      | CloudTrail                      |
| Amazon CloudWatch                      | CloudWatch                      |
| Amazon CloudWatch Logs                 | Logs                            |
| Amazon CloudWatch Application Insights | CloudWatch Application Insights |
| Amazon CodeBuild                       | CodeBuild                       |
| Amazon CodeCommit                      | CodeCommit                      |
| Amazon CodeGuru Profiler               | CodeGuru Profiler               |
| Amazon CodePipeline                    | CodePipeline                    |
| AWS CodeStar                           | CodeStar                        |
| AWS CodeStar 通知                        | CodeStar Notifications          |
| AWS CodeStar 连接                        | CodeStar Connections            |
| Amazon Cognito 身份池                     | Cognito Identity Pools          |
| Amazon Cognito Sync                    | Cognito Sync                    |
| Amazon Comprehend                      | Comprehend                      |
| Amazon Comprehend Medical              | Comprehend Medical              |
| Amazon Compute Optimizer               | ComputeOptimizer                |

| 服务                                | Service 维度的值                         |
|-----------------------------------|--------------------------------------|
| Amazon Connect                    | Connect                              |
| Amazon Connect Customer Profiles  | Customer Profiles                    |
| Amazon 成本和使用情况报告                  | Cost and Usage Report                |
| Amazon Cost Explorer              | Cost Explorer                        |
| Amazon Data Exchange              | Data Exchange                        |
| Amazon Data Lifecycle Manager     | Data Lifecycle Manager               |
| Amazon Database Migration Service | Database Migration Service           |
| Amazon DataSync                   | DataSync                             |
| Amazon DeepLens                   | AWS DeepLens                         |
| Amazon Detective                  | Detective                            |
| Device Advisor                    | Device Advisor                       |
| Amazon Direct Connect             | Direct Connect                       |
| Amazon Directory Service          | Directory Service                    |
| DynamoDB Accelerator              | DynamoDBAccelerator                  |
| Amazon EC2                        | EC2                                  |
| Amazon Elastic Container Registry | ECR Public                           |
| Amazon Elastic Container Service  | ECS                                  |
| Amazon Elastic File System        | EFS                                  |
| Amazon Elastic Kubernetes Service | EKS                                  |
| Amazon Elastic Beanstalk          | Elastic Beanstalk                    |
| Amazon Elastic Inference          | Elastic Inference                    |
| Elastic Load Balancing            | Elastic Load Balancing               |
| Amazon EMR                        | EMR Containers                       |
| Amazon Firewall Manager           | Firewall Manager                     |
| Amazon FSx                        | FSx                                  |
| Amazon GameLift                   | GameLift                             |
| Amazon Glue DataBrew              | DataBrew                             |
| Amazon Managed Grafana            | Grafana                              |
| Amazon IoT Greengrass             | Greengrass                           |
| Amazon Ground Station             | Ground Station                       |
| Amazon Health API 和通知             | Amazon Health APIs And Notifications |

| 服务                                      | Service 维度的值                  |
|-----------------------------------------|-------------------------------|
| Amazon Interactive Video Service        | IVS                           |
| Amazon IoT Core                         | IoT                           |
| Amazon IoT 一键                           | IoT 1-Click                   |
| Amazon IoT Events                       | IoT Events                    |
| Amazon IoT RoboRunner                   | IoT RoboRunner                |
| Amazon IoT SiteWise                     | IoT Sitewise                  |
| Amazon IoT Wireless                     | IoT Wireless                  |
| Amazon Kendra                           | Kendra                        |
| Amazon Keyspaces (for Apache Cassandra) | Keyspaces                     |
| Amazon Kinesis Data Analytics           | Kinesis Analytics             |
| Amazon Kinesis Data Firehose            | Firehose                      |
| Kinesis Video Streams                   | Kinesis Video Streams         |
| Amazon Key Management Service           | KMS                           |
| Amazon Lambda                           | Lambda                        |
| Amazon Launch Wizard                    | Launch Wizard                 |
| Amazon Lex                              | Amazon Lex                    |
| Amazon Lightsail                        | Lightsail                     |
| Amazon Location Service                 | Location                      |
| Amazon Lookout for Vision               | Lookout for Vision            |
| Amazon Machine Learning                 | Amazon Machine Learning       |
| Amazon Macie                            | Macie                         |
| Amazon Managed Services                 | AWS Managed Services          |
| Amazon Marketplace Commerce Analytics   | Marketplace Analytics Service |
| AWS Elemental MediaConnect              | MediaConnect                  |
| AWS Elemental MediaConvert              | MediaConvert                  |
| AWS Elemental MediaLive                 | MediaLive                     |
| AWS Elemental MediaStore                | Mediastore                    |
| AWS Elemental MediaTailor               | MediaTailor                   |
| Amazon Mobile Hub                       | Mobile Hub                    |
| Amazon Network Firewall                 | Network Firewall              |
| Amazon OpsWorks                         | OpsWorks                      |

| 服务                                           | Service 维度的值                    |
|----------------------------------------------|---------------------------------|
| 用于配置管理的 Amazon OpsWorks                      | OPsWorks CM                     |
| Amazon Outposts                              | Outposts                        |
| Amazon Organizations                         | Organizations                   |
| Amazon RDS Performance Insights              | Performance Insights            |
| Amazon Pinpoint                              | Pinpoint                        |
| Amazon Private Certificate Authority         | Private Certificate Authority   |
| Amazon Managed Service for Prometheus        | Prometheus                      |
| Amazon Proton                                | Proton                          |
| Amazon Quantum Ledger Database (Amazon QLDB) | QLDB                            |
| Amazon RDS                                   | RDS                             |
| Amazon Redshift                              | Redshift Data API               |
| Amazon Rekognition                           | Rekognition                     |
| Amazon Resource Access Manager               | Resource Access Manager         |
| Amazon Resource Groups                       | Resource Groups                 |
| Amazon Resource Groups Tagging API           | Resource Groups Tagging API     |
| Amazon RoboMaker                             | RoboMaker                       |
| Amazon Route 53 Domains                      | Route 53 Domains                |
| Amazon Route 53 Resolver                     | Route 53 Resolver               |
| Amazon S3                                    | S3                              |
| Simple Storage Service ( Amazon S3 ) Glacier | Amazon S3 Glacier               |
| Amazon SageMaker                             | Sagemaker                       |
| Savings Plans                                | Savings Plans                   |
| Amazon Secrets Manager                       | Secrets Manager                 |
| Amazon Security Hub                          | Security Hub                    |
| Amazon Server Migration Service              | Amazon Server Migration Service |
| Amazon Service Catalog AppRegistry           | Service Catalog AppRegistry     |
| Service Quotas                               | Service Quotas                  |
| Amazon Shield                                | Shield                          |
| Amazon Signer                                | Signer                          |
| Amazon Simple Notification Service           | SNS                             |



| 服务                             | Service 维度的值         |
|--------------------------------|----------------------|
| Amazon Simple Email Service    | SES                  |
| Amazon Simple Queue Service    | SQS                  |
| Identity Store                 | Identity Store       |
| Storage Gateway                | Storage Gateway      |
| Amazon Web Services Support    | Support              |
| Amazon Simple Workflow Service | SWF                  |
| Amazon Textract                | Textract             |
| Amazon IoT Things Graph        | ThingsGraph          |
| Amazon Timestream              | Timestream           |
| Amazon Transcribe              | Transcribe           |
| Amazon Translate               | Translate            |
| Amazon Transcribe 串流转录         | Transcribe Streaming |
| Amazon Transfer Family         | Transfer             |
| Amazon WAF                     | WAF                  |
| Amazon WorkDocs                | Amazon WorkDocs      |
| Amazon WorkLink                | WorkLink             |
| Amazon WorkMail                | Amazon WorkMail      |
| Amazon WorkSpaces              | Workspaces           |
| Amazon X-Ray                   | X-Ray                |

某些服务还会报告其他 API 的使用情况指标。若要查看 API 是否向 CloudWatch 报告使用情况指标，请使用 CloudWatch 控制台查看该服务在 AWS/Usage 命名空间中报告的指标。

查看向 CloudWatch 报告使用情况指标的服务 API 列表

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，请选择指标。
3. 在 All metrics ( 所有指标 ) 选项卡上，选择 Usage ( 使用情况 )，然后选择 By Amazon Resource ( 按亚马逊云科技资源 )。
4. 在指标列表旁边的搜索框中，输入服务的名称。指标将根据您输入的服务筛选出来。

## CloudWatch 使用情况指标

CloudWatch 会收集跟踪某些 Amazon 资源的使用情况的指标。这些指标与 Amazon 服务配额对应。跟踪这些指标可帮助您主动管理配额。有关更多信息，请参阅 [可视化 Service Quotas 并设置告警 \(p. 775\)](#)。

服务配额使用指标位于 AWS/Usage 命名空间中，并且每分钟收集一次。

目前，CloudWatch 在此命名空间中发布的唯一指标名称是 `CallCount`。此指标与 `Resource`、`Service` 和 `Type` 维度一同发布。`Resource` 维度指定要跟踪的 API 操作的名称。例如，具有 `"Service": "CloudWatch"`、`"Type": "API"` 和 `"Resource": "PutMetricData"` 维度的 `CallCount` 指标指示了 CloudWatch `PutMetricData` API 操作在您的账户中被调用的次数。

`CallCount` 指标没有指定的单位。该指标中最实用的统计数据是 `SUM`，它表示以 1 分钟为间隔的总操作数。

指标

| 指标                     | 描述                |
|------------------------|-------------------|
| <code>CallCount</code> | 在您的账户中执行的指定操作的数量。 |

维度

| 维度                    | 描述                                                                                                                                                                                                               |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Service</code>  | 包含该资源的 Amazon 服务的名称。对于 CloudWatch 使用情况指标，此维度的值为 <code>CloudWatch</code> 。                                                                                                                                        |
| <code>Class</code>    | 所跟踪的资源的类。CloudWatch API 使用情况指标使用此维度（值为 <code>None</code> ）。                                                                                                                                                      |
| <code>Type</code>     | 所跟踪的资源的类型。目前，当 <code>Service</code> 维度为 <code>CloudWatch</code> 时， <code>Type</code> 的唯一有效值为 <code>API</code> 。                                                                                                  |
| <code>Resource</code> | API 操作的名称。包括下列有效值：<br><code>DeleteAlarms</code><br><code>DeleteDashboards</code> 、 <code>DescribeAlarmHistory</code> 、 <code>DescribeAlarms</code> 、 <code>GetDashboard</code> 、<br>和 <code>PutMetricData</code> |

# CloudWatch 教程

下面的方案说明如何使用 Amazon CloudWatch。在第一个方案中，您将使用 CloudWatch 控制台创建账单告警，以便跟踪您的 Amazon 使用率并在超过特定消费阈值时通知您。在第二个更高级的方案中，您使用 Amazon Command Line Interface (Amazon CLI) 为名为 GetStarted 的假想应用程序发布单个指标。

## 场景

- [监控您的预估费用 \(p. 783\)](#)
- [发布指标 \(p. 785\)](#)

## 方案：使用 CloudWatch 监控您的估算费用

在此方案中，将创建一个 Amazon CloudWatch 告警来监控估计费用。在为您的 Amazon 账户启用估计费用监控时，将每天计算几次估计费用并作为指标数据发送到 CloudWatch。

账单指标数据存储在\*\*美国东部（弗吉尼亚北部）区域中，并且反映全球费用。此数据包括您所用的每种 Amazon 服务的估计费用以及 Amazon 估计费用总额。

可以选择在收费超过特定阈值时通过电子邮件接收警报。这些告警由 CloudWatch 触发，消息使用 Amazon Simple Notification Service (Amazon SNS) 进行发送。

### Note

有关分析已计费的 CloudWatch 费用的信息，请参阅 [CloudWatch 账单和成本 \(p. 7\)](#)。

## 任务

- [步骤 1：启用账单提醒 \(p. 783\)](#)
- [步骤 2：创建账单告警 \(p. 784\)](#)
- [步骤 3：检查告警状态 \(p. 784\)](#)
- [步骤 4：编辑账单告警 \(p. 785\)](#)
- [步骤 5：删除账单告警 \(p. 785\)](#)

## 步骤 1：启用账单提醒

在为估算费用创建告警之前，您必须启用账单提醒，以便监控 Amazon 估算费用并使用账单指标数据创建告警。启用账单提醒后，您将无法禁用数据收集，但是可以删除任何已创建的账单警报。

首次启用账单提醒后，大约需要 15 分钟时间，您就可以查看账单数据和设置账单告警。

## 要求

- 您必须使用账户根用户凭证或作为被授予权限的 IAM 用户登录，才能查看账单信息。
- 对于整合账单账户，每个关联账户的账单数据可以在付款账户登录后找到。您可以查看每个关联账户以及整合账户的估计费用总和，和各项服务的估计费用。
- 在整合账单账户中，仅当付款人账户启用 Receive Billing Alerts（接收账单提醒）首选项时，才会捕获成员关联账户的指标。如果您更改了您的管理账户/付款人账户，则必须新的管理账户/付款人账户中启用账单提醒。
- 该账户不能属于 Amazon 合作伙伴网络 (APN)，因为对于 APN 账户，账单指标不会发布到 CloudWatch。有关更多信息，请参阅 [Amazon 合作伙伴网络](#)。

### 启用对估计费用的监控

1. 访问 <https://console.aws.amazon.com/billing/>，打开 Amazon Billing 控制台。
2. 在导航窗格中，选择 Preferences。
3. 选择 Receive Billing Alerts。
4. 选择 Save preferences (保存首选项)。

## 步骤 2：创建账单告警

### Important

创建账单告警之前，您必须将区域设置为美国东部（弗吉尼亚州北部）。账单指标数据存储在該区域中，并表示全球费用。您还必须为您的账户启用账单提醒；或者，如果您使用的是整合账单，则必须在管理账户/付款人账户中启用账单提醒。有关更多信息，请参阅[步骤 1：启用账单提醒](#)。

在该过程中，您可以创建一个告警，以便在 Amazon 的估算费用超出定义的阈值时发送通知。

### 使用 CloudWatch 控制台创建告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 在导航窗格中，选择 Alarms (告警)，然后选择 All alarms (所有告警)。
3. 选择 Create alarm (创建警报)。
4. 选择选择指标。在 Browse (浏览) 中，选择 Billing (账单)，然后选择 Total Estimated Charge (估计费用总额)。

### Note

如果您没看到账单/估计费用总额指标，则启用账单提醒，并将您的区域更改为美国东部（弗吉尼亚州北部）。有关更多信息，请参阅[步骤 1：启用账单提醒](#)。

5. 选中 EstimatedCharges (估算费用) 指标的复选框，然后选择 Select metric (选择指标)。
6. 对于 Threshold type (阈值类型)，选择 Static (静态)。
7. 对于 Whenever EstimatedCharges is ... (当估算费用...)，选择 Greater (大)。
8. 对于 than ... (于...)，定义触发告警的阈值（例如 200 USD）。

### Note

定义阈值后，预览图显示您当月的预估费用。

9. 选择 Next (下一步)。
10. 在 Notification (通知) 下，选择当您的告警处于 ALARM 状态时要通知的 Amazon SNS 主题。您可以选择现有的 Amazon SNS 主题，创建一个新 Amazon SNS 主题，或使用主题 ARN 通知其他账户。如果您希望您的告警为相同告警状态或不同告警状态发送多个通知，请选择 Add notification (添加通知)。
11. 选择 Next (下一步)。
12. 在 Name and description (名称和描述) 下，为您的告警输入名称。
  - (可选) 输入告警的描述。
13. 在 Preview and create (预览并创建) 下，确保您的配置正确，然后选择 Create alarm (创建警告)。

## 步骤 3：检查告警状态

现在，请检查您刚创建的账单警报的状态。

### 检查警报状态

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 如果需要，可将区域更改为美国东部（弗吉尼亚北部）。账单指标数据存储在此区域中，并且反映全球费用。
3. 在导航窗格中，选择 Alarms（告警）。
4. 选中警报旁边的复选框。在确认订阅之前，它显示为“等待确认”。在确认订阅后，请刷新控制台以显示更新后的状态。

## 步骤 4：编辑账单告警

例如，您可能想要将每个月在 Amazon 上花费的金额从 200 美元增加至 400 美元。您可以编辑现有账单警报并增加在触发警报前必须超过的金额。

### 编辑账单警报

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 如果需要，可将区域更改为美国东部（弗吉尼亚北部）。账单指标数据存储在此区域中，并且反映全球费用。
3. 在导航窗格中，选择 Alarms（告警）。
4. 选中警报旁的复选框，然后依次选择 Actions（操作）和 Modify（修改）。
5. 对于 Whenever my total Amazon charges for the month exceed（每当该月我的亚马逊云科技费用总额超过时），请指定必须超过才会触发告警并发送电子邮件通知的新金额。
6. 选择 Save Changes。

## 步骤 5：删除账单告警

如果您不再需要账单警报，可将其删除。

### 删除账单告警

1. 通过以下网址打开 CloudWatch 控制台：<https://console.aws.amazon.com/cloudwatch/>。
2. 如果需要，可将区域更改为美国东部（弗吉尼亚北部）。账单指标数据存储在此区域中，并且反映全球费用。
3. 在导航窗格中，选择 Alarms（告警）。
4. 选中告警旁的复选框，然后依次选择 Actions（操作）和 Delete（删除）。
5. 当系统提示进行确认时，选择 Yes, Delete（是，删除）。

## 方案：将指标发布到 CloudWatch

在此方案中，使用 Amazon Command Line Interface (Amazon CLI) 为名为 GetStarted 的假想应用程序发布单个指标。如果您尚未安装和配置 Amazon CLI，请参阅 Amazon Command Line Interface 用户指南中的[开始设置 Amazon Command Line Interface](#)。

### 任务

- [步骤 1：定义数据配置 \(p. 786\)](#)
- [步骤 2：将指标添加到 CloudWatch \(p. 786\)](#)
- [步骤 3：获取 CloudWatch 中的统计数据 \(p. 787\)](#)

- [步骤 4：使用控制台查看图表 \(p. 787\)](#)

## 步骤 1：定义数据配置

在此场景中，您将发布跟踪该应用程序的请求延迟的数据点。为您的指标和命名空间选择您能够理解的名称。对本例而言，可以将指标命名为 RequestLatency 并将所有的数据点发布到 GetStarted 命名空间。

您将会发布共表示三小时延迟数据的几个数据点。原始数据由分布在三小时中的 15 个请求延迟读数构成。每个读数均以毫秒为单位：

- 第一小时：87、51、125、235
- 第二小时：121、113、189、65、89
- 第三小时：100、47、133、98、100、328

您可以将数据作为单一数据点或数据点聚合发布到 CloudWatch，聚合集称为统计数据集。您可以在低达一分钟的时间内将指标整合到粒度级。您可以将聚合数据点作为含四个预定义键（Sum、Minimum、Maximum 和 SampleCount）的统计数据集发布到 CloudWatch。

您将会把来自第一小时的数据点作为单一数据点发布。对于第二小时和第三小时的数据，您将会整合数据点并发布各小时的统计数据集。项值如下表所示。

| 小时 | 原始数据                       | 总计  | 最低 | 最高  | 样本数 |
|----|----------------------------|-----|----|-----|-----|
| 1  | 87                         |     |    |     |     |
| 1  | 51                         |     |    |     |     |
| 1  | 125                        |     |    |     |     |
| 1  | 235                        |     |    |     |     |
| 2  | 121, 113, 189, 65, 89      | 577 | 65 | 189 | 5   |
| 3  | 100, 47, 133, 98, 100, 328 | 806 | 47 | 328 | 6   |

## 步骤 2：将指标添加到 CloudWatch

定义数据配置后，您可以随时添加数据。

将数据点发布到 CloudWatch

1. 在命令提示符下，运行以下 `put-metric-data` 命令以添加第一个小时的数据。将示例时间戳替换为通用协调时间 (UTC) 中过去两小时的时间戳。

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
--timestamp 2016-10-14T20:30:00Z --value 87 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
--timestamp 2016-10-14T20:30:00Z --value 51 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
--timestamp 2016-10-14T20:30:00Z --value 125 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
--timestamp 2016-10-14T20:30:00Z --value 235 --unit Milliseconds
```

2. 添加第二个小时的数据，并使用比第一个小时晚一小时的时间戳。

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
--timestamp 2016-10-14T21:30:00Z --statistic-values
Sum=577,Minimum=65,Maximum=189,SampleCount=5 --unit Milliseconds
```

3. 添加第三个小时的数据，并忽略默认为当前时间的时间戳。

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
--statistic-values Sum=806,Minimum=47,Maximum=328,SampleCount=6 --unit Milliseconds
```

## 步骤 3：获取 CloudWatch 中的统计数据

您现在已向 CloudWatch 发布了指标，可以使用 [get-metric-statistics](#) 命令检索基于这些指标的统计数据（如下所示）。确保指定足够久的过去 `--start-time` 和 `--end-time` 以涵盖已发布的最早时间戳。

```
aws cloudwatch get-metric-statistics --namespace GetStarted --metric-name RequestLatency --
statistics Average \
--start-time 2016-10-14T00:00:00Z --end-time 2016-10-15T00:00:00Z --period 60
```

下面是示例输出：

```
{
 "Datapoints": [],
 "Label": "Request:Latency"
}
```

## 步骤 4：使用控制台查看图表

将指标发布到 CloudWatch 后，您可以使用 CloudWatch 控制台查看统计图表。

如需在控制台中查看统计数据图表

1. 访问 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在 Navigation 窗格中，选择 Metrics。
3. 在全部指标选项卡的搜索框中，键入 RequestLatency 并按 Enter。
4. 选中 RequestLatency 指标的复选框。上方窗格中会显示一个指标数据图表。

有关更多信息，请参阅[绘制指标的图表](#) (p. 104)。

# 将 CloudWatch 与 Amazon SDK 结合使用

Amazon 软件开发工具包 (SDK) 适用于许多常用编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地了解其首选语言构建应用程序。

| 软件开发工具包文档                                     |
|-----------------------------------------------|
| <a href="#">Amazon SDK for Java</a>           |
| <a href="#">Amazon SDK for JavaScript</a>     |
| <a href="#">Amazon SDK for Kotlin</a>         |
| <a href="#">Amazon SDK for .NET</a>           |
| <a href="#">Amazon SDK for PHP</a>            |
| <a href="#">Amazon SDK for Python (Boto3)</a> |
| <a href="#">Amazon SDK for Ruby</a>           |
| <a href="#">Amazon SDK for Rust</a>           |
| <a href="#">Amazon SDK for Swift</a>          |

有关特定于 CloudWatch 的示例，请参阅 [使用 Amazon SDK 的 CloudWatch 代码示例 \(p. 789\)](#)。



# 使用 Amazon SDK 的 CloudWatch 代码示例

以下代码示例展示如何将 CloudWatch 与 Amazon 软件开发工具包 (SDK) 结合使用。

这些示例可分为以下几类：

## 操作

展示如何调用具体服务函数的代码节选。

## 场景

展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 代码示例

- [使用 Amazon SDK 的 CloudWatch 操作 \(p. 789\)](#)
  - [使用 Amazon SDK 创建 CloudWatch 告警，用于监控指标 \(p. 790\)](#)
  - [使用 Amazon SDK 删除 CloudWatch 告警 \(p. 795\)](#)
  - [使用 Amazon SDK 描述 CloudWatch 告警的历史记录 \(p. 800\)](#)
  - [使用 Amazon SDK 为指标描述 CloudWatch 告警 \(p. 801\)](#)
  - [使用 Amazon SDK 禁用 CloudWatch 告警操作 \(p. 805\)](#)
  - [使用 Amazon SDK 启用 CloudWatch 告警操作 \(p. 810\)](#)
  - [获取 CloudWatch 控制面板的详细信息 \(p. 816\)](#)
  - [使用 Amazon SDK 获取 CloudWatch 指标统计数据 \(p. 817\)](#)
  - [列出 CloudWatch 控制面板 \(p. 818\)](#)
  - [使用 Amazon SDK 列出 CloudWatch 指标 \(p. 819\)](#)
  - [使用 Amazon SDK 将一组数据放入 CloudWatch 指标 \(p. 825\)](#)
  - [使用 Amazon SDK 将数据放入 CloudWatch 指标 \(p. 826\)](#)
- [使用 Amazon SDK 的 CloudWatch 场景 \(p. 830\)](#)
  - [使用 Amazon SDK 管理 CloudWatch 指标和告警 \(p. 830\)](#)

## 使用 Amazon SDK 的 CloudWatch 操作

以下代码示例演示了如何使用 Amazon SDK 来执行单个 CloudWatch 操作。这些代码节选调用了 CloudWatch API，但不旨在孤立运行。每个示例都包含一个指向 GitHub 的链接，其中包含了有关如何在上下文中设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 [Amazon CloudWatch API 参考](#)。

## 示例

- [使用 Amazon SDK 创建 CloudWatch 告警，用于监控指标 \(p. 790\)](#)
- [使用 Amazon SDK 删除 CloudWatch 告警 \(p. 795\)](#)
- [使用 Amazon SDK 描述 CloudWatch 告警的历史记录 \(p. 800\)](#)
- [使用 Amazon SDK 为指标描述 CloudWatch 告警 \(p. 801\)](#)

- [使用 Amazon SDK 禁用 CloudWatch 告警操作 \(p. 805\)](#)
- [使用 Amazon SDK 启用 CloudWatch 告警操作 \(p. 810\)](#)
- [获取 CloudWatch 控制面板的详细信息 \(p. 816\)](#)
- [使用 Amazon SDK 获取 CloudWatch 指标统计数据 \(p. 817\)](#)
- [列出 CloudWatch 控制面板 \(p. 818\)](#)
- [使用 Amazon SDK 列出 CloudWatch 指标 \(p. 819\)](#)
- [使用 Amazon SDK 将一组数据放入 CloudWatch 指标 \(p. 825\)](#)
- [使用 Amazon SDK 将数据放入 CloudWatch 指标 \(p. 826\)](#)

## 使用 Amazon SDK 创建 CloudWatch 告警，用于监控指标

以下代码示例展示了如何创建监控指标的 Amazon CloudWatch 告警。

C++

适用于 C++ 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

创建告警以观看指标。

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
 Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);

request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
 std::cout << "Failed to create CloudWatch alarm:" <<
 outcome.GetError().GetMessage() << std::endl;
```

```
 }
 else
 {
 std::cout << "Successfully created CloudWatch alarm " << alarm_name
 << std::endl;
 }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for C++ API 参考中的 [PutMetricAlarm](#)。

## Java

### SDK for Java 2.x

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
instanceId) {

 try {
 Dimension dimension = Dimension.builder()
 .name("InstanceId")
 .value(instanceId).build();

 PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
 .alarmName(alarmName)
 .comparisonOperator(ComparisonOperator.GREATER_THAN_THRESHOLD)
 .evaluationPeriods(1)
 .metricName("CPUUtilization")
 .namespace("AWS/EC2")
 .period(60)
 .statistic(Statistic.AVERAGE)
 .threshold(70.0)
 .actionsEnabled(false)
 .alarmDescription("Alarm when server CPU utilization exceeds 70%")
 .unit(StandardUnit.SECONDS)
 .dimensions(dimension)
 .build();

 cw.putMetricAlarm(request);
 System.out.printf("Successfully created alarm with name %s",
alarmName);

 } catch (CloudWatchException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 [PutMetricAlarm](#)。

## JavaScript

### SDK for JavaScript V3

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

在单独的模块中创建客户端并将其导出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

导入软件开发工具包和客户端模块，然后调用 API。

```
// Import required AWS SDK clients and commands for Node.js
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "../libs/cloudWatchClient.js";

// Set the parameters
export const params = {
 AlarmName: "Web_Server_CPU_Utilization",
 ComparisonOperator: "GreaterThanOrEqualToThreshold",
 EvaluationPeriods: 1,
 MetricName: "CPUUtilization",
 Namespace: "AWS/EC2",
 Period: 60,
 Statistic: "Average",
 Threshold: 70.0,
 ActionsEnabled: false,
 AlarmDescription: "Alarm when server CPU exceeds 70%",
 Dimensions: [
 {
 Name: "InstanceId",
 Value: "INSTANCE_ID",
 },
],
 Unit: "Percent",
};

export const run = async () => {
 try {
 const data = await cwClient.send(new PutMetricAlarmCommand(params));
 console.log("Success", data);
 return data;
 } catch (err) {
 console.log("Error", err);
 }
};

// Uncomment this line to run execution within this file.
// run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 Amazon SDK for JavaScript API 参考中的 [PutMetricAlarm](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
```

```
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
 AlarmName: 'Web_Server_CPU_Utilization',
 ComparisonOperator: 'GreaterThanThreshold',
 EvaluationPeriods: 1,
 MetricName: 'CPUUtilization',
 Namespace: 'AWS/EC2',
 Period: 60,
 Statistic: 'Average',
 Threshold: 70.0,
 ActionsEnabled: false,
 AlarmDescription: 'Alarm when server CPU exceeds 70%',
 Dimensions: [
 {
 Name: 'InstanceId',
 Value: 'INSTANCE_ID'
 },
],
 Unit: 'Percent'
};

cw.putMetricAlarm(params, function(err, data) {
 if (err) {
 console.log("Error", err);
 } else {
 console.log("Success", data);
 }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 Amazon SDK for JavaScript API 参考中的 [PutMetricAlarm](#)。

## Kotlin

### SDK for Kotlin

#### Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun putMetricAlarm(alarmNameVal: String, instanceIdVal: String) {

 val dimensionOb = Dimension {
 name = "InstanceId"
 value = instanceIdVal
 }

 val request = PutMetricAlarmRequest {
 alarmName = alarmNameVal
 comparisonOperator = ComparisonOperator.GreaterThanThreshold
 evaluationPeriods = 1
 metricName = "CPUUtilization"
 namespace = "AWS/EC2"
 }
```

```
 period = 60
 statistic = Statistic.fromValue("Average")
 threshold = 70.0
 actionsEnabled = false
 alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
utilization exceeds 70%"
 unit = StandardUnit.fromValue("Seconds")
 dimensions = listOf(dimensionObj)
 }

 CloudWatchClient { region = "us-east-1" }.use { cwClient ->
 cwClient.putMetricAlarm(request)
 println("Successfully created an alarm with name $alarmNameVal")
 }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Kotlin API 参考中的 [PutMetricAlarm](#)。

## Python

适用于 Python (Boto3) 的 SDK

### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
class CloudWatchWrapper:
 """Encapsulates Amazon CloudWatch functions."""
 def __init__(self, cloudwatch_resource):
 """
 :param cloudwatch_resource: A Boto3 CloudWatch resource.
 """
 self.cloudwatch_resource = cloudwatch_resource

 def create_metric_alarm(
 self, metric_namespace, metric_name, alarm_name, stat_type, period,
 eval_periods, threshold, comparison_op):
 """
 Creates an alarm that watches a metric.

 :param metric_namespace: The namespace of the metric.
 :param metric_name: The name of the metric.
 :param alarm_name: The name of the alarm.
 :param stat_type: The type of statistic the alarm watches.
 :param period: The period in which metric data are grouped to calculate
 statistics.
 :param eval_periods: The number of periods that the metric must be over the
 alarm threshold before the alarm is set into an
 alarmed
 state.
 :param threshold: The threshold value to compare against the metric
 statistic.
 :param comparison_op: The comparison operation used to compare the
 threshold
 against the metric.
 :return: The newly created alarm.
 """
 try:
 metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
 alarm = metric.put_alarm(
 AlarmName=alarm_name,
 Statistic=stat_type,
```

```
 Period=period,
 EvaluationPeriods=eval_periods,
 Threshold=threshold,
 ComparisonOperator=comparison_op)
 logger.info(
 "Added alarm %s to track metric %s.%s.", alarm_name,
metric_namespace,
 metric_name)
 except ClientError:
 logger.exception(
 "Couldn't add alarm %s to metric %s.%s", alarm_name,
metric_namespace,
 metric_name)
 raise
 else:
 return alarm
```

- 有关 API 详细信息，请参阅 Amazon 适用于 Python (Boto3) 的 SDK API 参考中的 [PutMetricAlarm](#)。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 使用 Amazon SDK 删除 CloudWatch 告警

以下代码示例展示如何删除 Amazon CloudWatch 告警。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to delete Amazon CloudWatch alarms. The example
/// was created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteAlarms
{
 public static async Task Main()
 {
 IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();

 var alarmNames = CreateAlarmNameList();
 await DeleteAlarmsAsyncExample(cwClient, alarmNames);
 }

 /// <summary>
 /// Delete the alarms whose names are listed in the alarmNames parameter.
 /// </summary>
 /// <param name="client">The initialized Amazon CloudWatch client.</param>
 /// <param name="alarmNames">A list of names for the alarms to be
```

```
/// deleted.</param>
public static async Task DeleteAlarmsAsyncExample(IAmazonCloudWatch client,
List<string> alarmNames)
{
 var request = new DeleteAlarmsRequest
 {
 AlarmNames = alarmNames,
 };

 try
 {
 var response = await client.DeleteAlarmsAsync(request);

 if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
 {
 Console.WriteLine("Alarms successfully deleted:");
 alarmNames
 .ForEach(name => Console.WriteLine($"{name}"));
 }
 }
 catch (ResourceNotFoundException ex)
 {
 Console.WriteLine($"Error: {ex.Message}");
 }
}

/// <summary>
/// Defines and returns the list of alarm names to delete.
/// </summary>
/// <returns>A list of alarm names.</returns>
public static List<string> CreateAlarmNameList()
{
 // The list of alarm names passed to DeleteAlarmsAsync
 // can contain up to 100 alarm names.
 var theList = new List<string>
 {
 "ALARM_NAME_1",
 "ALARM_NAME_2",
 };

 return theList;
}
}
```

- 有关 API 详细信息，请参阅 [Amazon SDK for .NET API 参考](#) 中的 DeleteAlarms。

## C++

### SDK for C++

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```



删除告警。

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DeleteAlarmsRequest request;
request.AddAlarmNames(alarm_name);

auto outcome = cw.DeleteAlarms(request);
if (!outcome.IsSuccess())
{
 std::cout << "Failed to delete CloudWatch alarm:" <<
 outcome.GetError().GetMessage() << std::endl;
}
else
{
 std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
 << std::endl;
}
```

- 有关 API 详细信息，请参阅 [Amazon SDK for C++ API 参考](#) 中的 DeleteAlarms。

## Java

### SDK for Java 2.x

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
 try {
 DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
 .alarmNames(alarmName)
 .build();

 cw.deleteAlarms(request);
 System.out.printf("Successfully deleted alarm %s", alarmName);

 } catch (CloudWatchException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
```

- 有关 API 详细信息，请参阅 [Amazon SDK for Java 2.x API 参考](#) 中的 DeleteAlarms。

## JavaScript

### SDK for JavaScript V3

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

在单独的模块中创建客户端并将其导出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
```

```
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

导入软件开发工具包和客户端模块，然后调用 API。

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "../libs/cloudWatchClient.js";

// Set the parameters
export const params = { AlarmNames: "ALARM_NAME" }; // e.g.,
"Web_Server_CPU_Utilization"

export const run = async () => {
 try {
 const data = await cwClient.send(new DeleteAlarmsCommand(params));
 console.log("Success, alarm deleted; requestID:", data);
 return data;
 } catch (err) {
 console.log("Error", err);
 }
};

// Uncomment this line to run execution within this file.
// run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 [Amazon SDK for JavaScript API 参考](#)中的 DeleteAlarms。

## SDK for JavaScript V2

### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

导入软件开发工具包和客户端模块，然后调用 API。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
 AlarmNames: ['Web_Server_CPU_Utilization']
};

cw.deleteAlarms(params, function(err, data) {
 if (err) {
 console.log("Error", err);
 } else {
 console.log("Success", data);
 }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 [Amazon SDK for JavaScript API 参考](#) 中的 `DeleteAlarms`。

## Kotlin

### SDK for Kotlin

#### Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun deleteCWAlarm(alarmNameVal: String) {

 val request = DeleteAlarmsRequest {
 alarmNames = listOf(alarmNameVal)
 }

 CloudWatchClient { region = "us-east-1" }.use { cwClient ->
 cwClient.deleteAlarms(request)
 println("Successfully deleted alarm $alarmNameVal")
 }
}
```

- 有关 API 详细信息，请参阅 [Amazon SDK for Kotlin API 参考](#) 中的 `DeleteAlarms`。

## Python

### 适用于 Python (Boto3) 的 SDK

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
class CloudWatchWrapper:
 """Encapsulates Amazon CloudWatch functions."""
 def __init__(self, cloudwatch_resource):
 """
 :param cloudwatch_resource: A Boto3 CloudWatch resource.
 """
 self.cloudwatch_resource = cloudwatch_resource

 def delete_metric_alarms(self, metric_namespace, metric_name):
 """
 Deletes all of the alarms that are currently watching the specified metric.

 :param metric_namespace: The namespace of the metric.
 :param metric_name: The name of the metric.
 """
 try:
 metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
 metric.alarms.delete()
 logger.info(
 "Deleted alarms for metric %s.%s.", metric_namespace, metric_name)
 except ClientError:
 logger.exception(
 "Couldn't delete alarms for metric %s.%s.", metric_namespace,
```

```
metric_name)
raise
```

- 有关 API 详细信息，请参阅 Amazon 适用于 Python (Boto3) 的 SDK API 参考中的 [DeleteAlarms](#)。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 使用 Amazon SDK 描述 CloudWatch 告警的历史记录

以下代码示例展示如何描述 Amazon CloudWatch 告警的历史记录。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

获取 CloudWatch 告警列表，然后检索每个告警的历史记录。

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example retrieves a list of Amazon CloudWatch alarms and, for
/// each one, displays its history. The example was created using the
/// AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class DescribeAlarmHistories
{
 /// <summary>
 /// Retrieves a list of alarms and then passes each name to the
 /// DescribeAlarmHistoriesAsync method to retrieve its history.
 /// </summary>
 public static async Task Main()
 {
 IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
 var response = await cwClient.DescribeAlarmsAsync();

 foreach (var alarm in response.MetricAlarms)
 {
 await DescribeAlarmHistoriesAsync(cwClient, alarm.AlarmName);
 }
 }

 /// <summary>
 /// Retrieves the CloudWatch alarm history for the alarm name passed
 /// to the method.
 /// </summary>
 /// <param name="client">An initialized CloudWatch client object.</param>
 /// <param name="alarmName">The CloudWatch alarm for which to retrieve
 /// history information.</param>
 public static async Task DescribeAlarmHistoriesAsync(IAmazonCloudWatch
client, string alarmName)
 {
 var request = new DescribeAlarmHistoryRequest
 {
```

```
 AlarmName = alarmName,
 EndDateUtc = DateTime.Today,
 HistoryItemType = HistoryItemType.Action,
 MaxRecords = 1,
 StartDateUtc = DateTime.Today.Subtract(TimeSpan.FromDays(30)),
 };

 var response = new DescribeAlarmHistoryResponse();

 do
 {
 response = await client.DescribeAlarmHistoryAsync(request);

 foreach (var item in response.AlarmHistoryItems)
 {
 Console.WriteLine(item.AlarmName);
 Console.WriteLine(item.HistorySummary);
 Console.WriteLine();
 }

 request.NextToken = response.NextToken;
 }
 while (!string.IsNullOrEmpty(response.NextToken));
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for .NET API 参考中的 [DescribeAlarmHistory](#)。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 使用 Amazon SDK 为指标描述 CloudWatch 告警

以下代码示例显示如何描述指标的 Amazon CloudWatch 告警。

C++

适用于 C++ 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

描述告警。

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DescribeAlarmsRequest request;
request.SetMaxRecords(1);

bool done = false;
```

```
bool header = false;
while (!done)
{
 auto outcome = cw.DescribeAlarms(request);
 if (!outcome.IsSuccess())
 {
 std::cout << "Failed to describe CloudWatch alarms:" <<
 outcome.GetError().GetMessage() << std::endl;
 break;
 }

 if (!header)
 {
 std::cout << std::left <<
 std::setw(32) << "Name" <<
 std::setw(64) << "Arn" <<
 std::setw(64) << "Description" <<
 std::setw(20) << "LastUpdated" <<
 std::endl;
 header = true;
 }

 const auto &alarms = outcome.GetResult().GetMetricAlarms();
 for (const auto &alarm : alarms)
 {
 std::cout << std::left <<
 std::setw(32) << alarm.GetAlarmName() <<
 std::setw(64) << alarm.GetAlarmArn() <<
 std::setw(64) << alarm.GetAlarmDescription() <<
 std::setw(20) <<
 alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
 SIMPLE_DATE_FORMAT_STR) <<
 std::endl;
 }

 const auto &next_token = outcome.GetResult().GetNextToken();
 request.SetNextToken(next_token);
 done = next_token.empty();
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for C++ API 参考中的 [DescribeAlarmsForMetric](#)。

## Java

### SDK for Java 2.x

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void descWAlarms(CloudWatchClient cw) {
 try {

 boolean done = false;
 String newToken = null;

 while(!done) {
 DescribeAlarmsResponse response;
 if (newToken == null) {
 DescribeAlarmsRequest request =
 DescribeAlarmsRequest.builder().build();
 response = cw.describeAlarms(request);
```

```
 } else {
 DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
 .nextToken(newToken)
 .build();
 response = cw.describeAlarms(request);
 }

 for(MetricAlarm alarm : response.metricAlarms()) {
 System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
 }

 if(response.nextToken() == null) {
 done = true;
 } else {
 newToken = response.nextToken();
 }
 }

 } catch (CloudWatchException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
 System.out.printf("Done");
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 [DescribeAlarmsForMetric](#)。

## JavaScript

### SDK for JavaScript V3

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

在单独的模块中创建客户端并将其导出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

导入软件开发工具包和客户端模块，然后调用 API。

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "../libs/cloudWatchClient.js";

// Set the parameters
export const params = { StateValue: "INSUFFICIENT_DATA" };

export const run = async () => {
 try {
 const data = await cwClient.send(new DescribeAlarmsCommand(params));
 console.log("Success", data);
 data.MetricAlarms.forEach(function (item) {
 console.log(item.AlarmName);
 });
 }
}
```

```
});
return data;
} catch (err) {
 console.log("Error", err);
}
};
// Uncomment this line to run execution within this file.
// run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 Amazon SDK for JavaScript API 参考中的 [DescribeAlarmsForMetric](#)。

#### SDK for JavaScript V2

##### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

cw.describeAlarms({StateValue: 'INSUFFICIENT_DATA'}, function(err, data) {
 if (err) {
 console.log("Error", err);
 } else {
 // List the names of all current alarms in the console
 data.MetricAlarms.forEach(function (item, index, array) {
 console.log(item.AlarmName);
 });
 }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 Amazon SDK for JavaScript API 参考中的 [DescribeAlarmsForMetric](#)。

#### Kotlin

##### SDK for Kotlin

##### Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

##### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun descCWAlarms() {

 CloudWatchClient { region = "us-east-1" }.use { cwClient ->
 val response = cwClient.describeAlarms(DescribeAlarmsRequest {})
 response.metricAlarms?.forEach { alarm ->
 println("Retrieved alarm ${alarm.alarmName}")
 }
 }
}
```



```
}
}
}
```

- 有关 API 详细信息，请参阅 [Amazon SDK for Kotlin API 参考](#) 中的 DescribeAlarmsForMetric。

## Python

适用于 Python (Boto3) 的 SDK

### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
class CloudWatchWrapper:
 """Encapsulates Amazon CloudWatch functions."""
 def __init__(self, cloudwatch_resource):
 """
 :param cloudwatch_resource: A Boto3 CloudWatch resource.
 """
 self.cloudwatch_resource = cloudwatch_resource

 def get_metric_alarms(self, metric_namespace, metric_name):
 """
 Gets the alarms that are currently watching the specified metric.

 :param metric_namespace: The namespace of the metric.
 :param metric_name: The name of the metric.
 :returns: An iterator that yields the alarms.
 """
 metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
 alarm_iter = metric.alarms.all()
 logger.info("Got alarms for metric %s.%s.", metric_namespace, metric_name)
 return alarm_iter
```

- 有关 API 详细信息，请参阅 [Amazon 适用于 Python \(Boto3\) 的 SDK API 参考](#) 中的 DescribeAlarmsForMetric。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 使用 Amazon SDK 禁用 CloudWatch 告警操作

以下代码示例展示如何禁用 Amazon CloudWatch 告警操作。

## .NET

Amazon SDK for .NET

### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
```

```
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to disable the Amazon CloudWatch actions for
/// one or more CloudWatch alarms. The example was created using the
/// AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DisableAlarmActions
{
 public static async Task Main()
 {
 IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
 var alarmNames = new List<string>
 {
 "ALARM_NAME",
 "ALARM_NAME_2",
 };

 var success = await DisableAlarmsActionsAsync(cwClient, alarmNames);

 if (success)
 {
 Console.WriteLine("Alarm action(s) successfully disabled.");
 }
 else
 {
 Console.WriteLine("Alarm action(s) were not disabled.")
 }
 }

 /// <summary>
 /// Disable the actions for the list of CloudWatch alarm names passed
 /// in the alarmNames parameter.
 /// </summary>
 /// <param name="client">An initialized CloudWatch client object.</param>
 /// <param name="alarmNames">The list of CloudWatch alarms to disable.</
param>
 /// <returns>A Boolean value indicating the success of the call.</returns>
 public static async Task<bool> DisableAlarmsActionsAsync(
 IAmazonCloudWatch client,
 List<string> alarmNames)
 {
 var request = new DisableAlarmActionsRequest
 {
 AlarmNames = alarmNames,
 };

 var response = await client.DisableAlarmActionsAsync(request);

 return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
 }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for .NET API 参考中的 [DisableAlarmActions](#)。

## C++

适用于 C++ 的 SDK

### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

禁用告警操作。

```
 Aws::CloudWatch::CloudWatchClient cw;

 Aws::CloudWatch::Model::DisableAlarmActionsRequest
 disableAlarmActionsRequest;
 disableAlarmActionsRequest.AddAlarmNames(alarm_name);

 auto disableAlarmActionsOutcome =
 cw.DisableAlarmActions(disableAlarmActionsRequest);
 if (!disableAlarmActionsOutcome.IsSuccess())
 {
 std::cout << "Failed to disable actions for alarm " << alarm_name <<
 ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
 std::endl;
 }
 else
 {
 std::cout << "Successfully disabled actions for alarm " <<
 alarm_name << std::endl;
 }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for C++ API 参考中的 [DisableAlarmActions](#)。

## Java

### SDK for Java 2.x

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void disableActions(CloudWatchClient cw, String alarmName) {
 try {
 DisableAlarmActionsRequest request =
 DisableAlarmActionsRequest.builder()
 .alarmNames(alarmName)
 .build();

 cw.disableAlarmActions(request);
 System.out.printf("Successfully disabled actions on alarm %s",
 alarmName);
 } catch (CloudWatchException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 [DisableAlarmActions](#)。

## JavaScript

### SDK for JavaScript V3

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

在单独的模块中创建客户端并将其导出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

导入软件开发工具包和客户端模块，然后调用 API。

```
// Import required AWS SDK clients and commands for Node.js
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "../libs/cloudWatchClient.js";

// Set the parameters
export const params = { AlarmNames: "ALARM_NAME" }; // e.g.,
 "Web_Server_CPU_Utilization"

export const run = async () => {
 try {
 const data = await cwClient.send(new DisableAlarmActionsCommand(params));
 console.log("Success, alarm disabled:", data);
 return data;
 } catch (err) {
 console.log("Error", err);
 }
};
// Uncomment this line to run execution within this file.
// run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 Amazon SDK for JavaScript API 参考中的 [DisableAlarmActions](#)。

### SDK for JavaScript V2

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

导入软件开发工具包和客户端模块，然后调用 API。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});
```

```
// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

cw.disableAlarmActions({AlarmNames: ['Web_Server_CPU_Utilization']}, function(err,
data) {
 if (err) {
 console.log("Error", err);
 } else {
 console.log("Success", data);
 }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 Amazon SDK for JavaScript API 参考中的 [DisableAlarmActions](#)。

## Kotlin

### SDK for Kotlin

#### Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun disableActions(alarmName: String) {

 val request = DisableAlarmActionsRequest {
 alarmNames = listOf(alarmName)
 }
 CloudWatchClient { region = "us-east-1" }.use { cwClient ->
 cwClient.disableAlarmActions(request)
 println("Successfully disabled actions on alarm $alarmName")
 }
}
```

- 有关 API 详细信息，请参阅 [Amazon SDK for Kotlin API 参考](#) 中的 [DisableAlarmActions](#)。

## Python

### 适用于 Python (Boto3) 的 SDK

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
class CloudWatchWrapper:
 """Encapsulates Amazon CloudWatch functions."""
 def __init__(self, cloudwatch_resource):
 """
 :param cloudwatch_resource: A Boto3 CloudWatch resource.
 """
 self.cloudwatch_resource = cloudwatch_resource
```

```
def enable_alarm_actions(self, alarm_name, enable):
 """
 Enables or disables actions on the specified alarm. Alarm actions can be
 used to send notifications or automate responses when an alarm enters a
 particular state.

 :param alarm_name: The name of the alarm.
 :param enable: When True, actions are enabled for the alarm. Otherwise,
they
 disabled.
 """
 try:
 alarm = self.cloudwatch_resource.Alarm(alarm_name)
 if enable:
 alarm.enable_actions()
 else:
 alarm.disable_actions()
 logger.info(
 "%s actions for alarm %s.", "Enabled" if enable else "Disabled",
 alarm_name)
 except ClientError:
 logger.exception(
 "Couldn't %s actions alarm %s.", "enable" if enable else "disable",
 alarm_name)
 raise
```

- 有关 API 详细信息，请参阅 [Amazon 适用于 Python \(Boto3\) 的 SDK API 参考](#) 中的 `DisableAlarmActions`。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 使用 Amazon SDK 启用 CloudWatch 告警操作

以下代码示例展示如何启用 Amazon CloudWatch 告警操作。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to enable the Amazon CloudWatch actions for
/// one or more CloudWatch alarms. The example was created using the
/// AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class EnableAlarmActions
{
 public static async Task Main()
 {
```

```
IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
var alarmNames = new List<string>
{
 "ALARM_NAME",
 "ALARM_NAME_2",
};

var success = await EnableAlarmActionsAsync(cwClient, alarmNames);

if (success)
{
 Console.WriteLine("Alarm action(s) successfully enabled.");
}
else
{
 Console.WriteLine("Alarm action(s) were not enabled.")
}

}

/// <summary>
/// Enable the actions for the list of CloudWatch alarm names passed
/// in the alarmNames parameter.
/// </summary>
/// <param name="client">An initialized CloudWatch client object.</param>
/// <param name="alarmNames">The list of CloudWatch alarms to enable.</
param>
/// <returns>A Boolean value indicating the success of the call.</returns>
public static async Task<bool> EnableAlarmActionsAsync(IAmazonCloudWatch
client, List<string> alarmNames)
{
 var request = new EnableAlarmActionsRequest
 {
 AlarmNames = alarmNames,
 };

 var response = await client.EnableAlarmActionsAsync(request);

 return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for .NET API 参考中的 [EnableAlarmActions](#)。

## C++

### SDK for C++

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

启用告警操作。

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
 Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
request.AddAlarmActions(actionArn);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);
request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
 std::cout << "Failed to create CloudWatch alarm:" <<
 outcome.GetError().GetMessage() << std::endl;
 return;
}

Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
enable_request.AddAlarmNames(alarm_name);

auto enable_outcome = cw.EnableAlarmActions(enable_request);
if (!enable_outcome.IsSuccess())
{
 std::cout << "Failed to enable alarm actions:" <<
 enable_outcome.GetError().GetMessage() << std::endl;
 return;
}

std::cout << "Successfully created alarm " << alarm_name <<
 " and enabled actions on it." << std::endl;
```

- 有关 API 详细信息，请参阅 Amazon SDK for C++ API 参考中的 [EnableAlarmActions](#)。

## Java

### SDK for Java 2.x

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void enableActions(CloudWatchClient cw, String alarm) {
 try {
 EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
 .alarmNames(alarm)
 .build();

 cw.enableAlarmActions(request);
 }
```



```
 System.out.printf("Successfully enabled actions on alarm %s", alarm);
 } catch (CloudWatchException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 [EnableAlarmActions](#)。

## JavaScript

### SDK for JavaScript V3

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

在单独的模块中创建客户端并将其导出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

导入软件开发工具包和客户端模块，然后调用 API。

```
// Import required AWS SDK clients and commands for Node.js
import {
 PutMetricAlarmCommand,
 EnableAlarmActionsCommand,
} from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
 AlarmName: "ALARM_NAME", //ALARM_NAME
 ComparisonOperator: "GreaterThanOrEqualToThreshold",
 EvaluationPeriods: 1,
 MetricName: "CPUUtilization",
 Namespace: "AWS/EC2",
 Period: 60,
 Statistic: "Average",
 Threshold: 70.0,
 ActionsEnabled: true,
 AlarmActions: ["ACTION_ARN"], //e.g., "arn:aws:automate:us-east-1:ec2:stop"
 AlarmDescription: "Alarm when server CPU exceeds 70%",
 Dimensions: [
 {
 Name: "InstanceId",
 Value: "INSTANCE_ID",
 },
],
 Unit: "Percent",
};
```

```
export const run = async () => {
 try {
 const data = await cwClient.send(new PutMetricAlarmCommand(params));
 console.log("Alarm action added; RequestID:", data);
 return data;
 const paramsEnableAlarmAction = {
 AlarmNames: [params.AlarmName],
 };
 try {
 const data = await cwClient.send(
 new EnableAlarmActionsCommand(paramsEnableAlarmAction)
);
 console.log("Alarm action enabled; RequestID:", data.$metadata.requestId);
 } catch (err) {
 console.log("Error", err);
 return data;
 }
 } catch (err) {
 console.log("Error", err);
 }
};
// Uncomment this line to run execution within this file.
// run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 Amazon SDK for JavaScript API 参考中的 [EnableAlarmActions](#)。

#### SDK for JavaScript V2

##### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

导入软件开发工具包和客户端模块，然后调用 API。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
 AlarmName: 'Web_Server_CPU_Utilization',
 ComparisonOperator: 'GreaterThanThreshold',
 EvaluationPeriods: 1,
 MetricName: 'CPUUtilization',
 Namespace: 'AWS/EC2',
 Period: 60,
 Statistic: 'Average',
 Threshold: 70.0,
 ActionsEnabled: true,
 AlarmActions: ['ACTION_ARN'],
 AlarmDescription: 'Alarm when server CPU exceeds 70%',
 Dimensions: [
 {
 Name: 'InstanceId',
 Value: 'INSTANCE_ID'
 },
],
 Unit: 'Percent'
};
```

```
cw.putMetricAlarm(params, function(err, data) {
 if (err) {
 console.log("Error", err);
 } else {
 console.log("Alarm action added", data);
 var paramsEnableAlarmAction = {
 AlarmNames: [params.AlarmName]
 };
 cw.enableAlarmActions(paramsEnableAlarmAction, function(err, data) {
 if (err) {
 console.log("Error", err);
 } else {
 console.log("Alarm action enabled", data);
 }
 });
 }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 Amazon SDK for JavaScript API 参考中的 [EnableAlarmActions](#)。

## Kotlin

### SDK for Kotlin

#### Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun enableActions(alarm: String) {

 val request = EnableAlarmActionsRequest {
 alarmNames = listOf(alarm)
 }

 CloudWatchClient { region = "us-east-1" }.use { cwClient ->
 cwClient.enableAlarmActions(request)
 println("Successfully enabled actions on alarm $alarm")
 }
}
```

- 有关 API 详细信息，请参阅 [Amazon SDK for Kotlin API 参考](#) 中的 [EnableAlarmActions](#)。

## Python

### 适用于 Python (Boto3) 的 SDK

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
class CloudWatchWrapper:
```

```
"""Encapsulates Amazon CloudWatch functions."""
def __init__(self, cloudwatch_resource):
 """
 :param cloudwatch_resource: A Boto3 CloudWatch resource.
 """
 self.cloudwatch_resource = cloudwatch_resource

def enable_alarm_actions(self, alarm_name, enable):
 """
 Enables or disables actions on the specified alarm. Alarm actions can be
 used to send notifications or automate responses when an alarm enters a
 particular state.

 :param alarm_name: The name of the alarm.
 :param enable: When True, actions are enabled for the alarm. Otherwise,
they
 disabled.
 """
 try:
 alarm = self.cloudwatch_resource.Alarm(alarm_name)
 if enable:
 alarm.enable_actions()
 else:
 alarm.disable_actions()
 logger.info(
 "%s actions for alarm %s.", "Enabled" if enable else "Disabled",
 alarm_name)
 except ClientError:
 logger.exception(
 "Couldn't %s actions alarm %s.", "enable" if enable else "disable",
 alarm_name)
 raise
```

- 有关 API 详细信息，请参阅 [Amazon 适用于 Python \(Boto3\) 的 SDK API 参考](#) 中的 EnableAlarmActions。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 获取 CloudWatch 控制面板的详细信息

以下代码示例展示如何获取 Amazon CloudWatch 控制面板详细信息。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to retrieve the details of an Amazon CloudWatch
/// dashboard. The return value from the call to GetDashboard is a json
/// object representing the widgets in the dashboard. The example was
```

```
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class GetDashboard
{
 public static async Task Main()
 {
 IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
 string dashboardName = "CloudWatch-Default";

 var body = await GetDashboardAsync(cwClient, dashboardName);

 Console.WriteLine(body);
 }

 /// <summary>
 /// Get the json that represents the dashboard.
 /// </summary>
 /// <param name="client">An initialized CloudWatch client.</param>
 /// <param name="dashboardName">The name of the dashboard.</param>
 /// <returns>The string containing the json value describing the
 /// contents and layout of the CloudWatch dashboard.</returns>
 public static async Task<string> GetDashboardAsync(IAmazonCloudWatch
client, string dashboardName)
 {
 var request = new GetDashboardRequest
 {
 DashboardName = dashboardName,
 };

 var response = await client.GetDashboardAsync(request);

 return response.DashboardBody;
 }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for .NET API 参考中的 [GetDashboard](#)。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 使用 Amazon SDK 获取 CloudWatch 指标统计数据

以下代码示例展示如何获取 Amazon CloudWatch 指标统计数据。

### Python

适用于 Python (Boto3) 的 SDK

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
class CloudWatchWrapper:
 """Encapsulates Amazon CloudWatch functions."""
 def __init__(self, cloudwatch_resource):
 """
 :param cloudwatch_resource: A Boto3 CloudWatch resource.
 """
```

```
self.cloudwatch_resource = cloudwatch_resource

def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
 """
 Gets statistics for a metric within a specified time span. Metrics are
 grouped
 into the specified period.

 :param namespace: The namespace of the metric.
 :param name: The name of the metric.
 :param start: The UTC start time of the time span to retrieve.
 :param end: The UTC end time of the time span to retrieve.
 :param period: The period, in seconds, in which to group metrics. The
period
 must match the granularity of the metric, which depends on
 the metric's age. For example, metrics that are older than
 three hours have a one-minute granularity, so the period
must
 be at least 60 and must be a multiple of 60.
 :param stat_types: The type of statistics to retrieve, such as average
value
 or maximum value.
 :return: The retrieved statistics for the metric.
 """
 try:
 metric = self.cloudwatch_resource.Metric(namespace, name)
 stats = metric.get_statistics(
 StartTime=start, EndTime=end, Period=period, Statistics=stat_types)
 logger.info(
 "Got %s statistics for %s.", len(stats['Datapoints']),
stats['Label'])
 except ClientError:
 logger.exception("Couldn't get statistics for %s.%s.", namespace, name)
 raise
 else:
 return stats
```

- 有关 API 详细信息，请参阅 Amazon 适用于 Python (Boto3) 的 SDK API 参考中的 [GetMetricStatistics](#)。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 列出 CloudWatch 控制面板

以下代码示例展示如何列出 Amazon CloudWatch 控制面板。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
```

```
using Amazon.CloudWatch.Model;

/// <summary>
/// Shows how to retrieve a list of Amazon CloudWatch dashboards. This
/// example was written using AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class ListDashboards
{
 public static async Task Main()
 {
 IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
 var dashboards = await ListDashboardsAsync(cwClient);

 DisplayDashboardList(dashboards);
 }

 /// <summary>
 /// Get the list of available dashboards.
 /// </summary>
 /// <param name="client">The initialized CloudWatch client used to
 /// retrieve a list of defined dashboards.</param>
 /// <returns>A list of DashboardEntry objects.</returns>
 public static async Task<List<DashboardEntry>>
ListDashboardsAsync(IAmazonCloudWatch client)
 {
 var response = await client.ListDashboardsAsync(new
ListDashboardsRequest());
 return response.DashboardEntries;
 }

 /// <summary>
 /// Displays the name of each CloudWatch Dashboard in the list passed
 /// to the method.
 /// </summary>
 /// <param name="dashboards">A list of DashboardEntry objects.</param>
 public static void DisplayDashboardList(List<DashboardEntry> dashboards)
 {
 if (dashboards.Count > 0)
 {
 Console.WriteLine("The following dashboards are defined:");
 foreach (var dashboard in dashboards)
 {
 Console.WriteLine($"Name: {dashboard.DashboardName} Last
modified: {dashboard.LastModified}");
 }
 }
 else
 {
 Console.WriteLine("No dashboards found.");
 }
 }
}
```

- 有关 API 详细信息，请参阅 [Amazon SDK for .NET API 参考](#) 中的 ListDashboards。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 使用 Amazon SDK 列出 CloudWatch 指标

以下代码示例展示如何列出 Amazon CloudWatch 指标。

## .NET

### Amazon SDK for .NET

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example demonstrates how to list metrics for Amazon CloudWatch.
/// The example was created using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class ListMetrics
{
 public static async Task Main()
 {
 IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();

 var filter = new DimensionFilter
 {
 Name = "InstanceType",
 Value = "t1.micro",
 };
 string metricName = "CPUUtilization";
 string namespaceName = "AWS/EC2";

 await ListMetricsAsync(cwClient, filter, metricName, namespaceName);
 }

 /// <summary>
 /// Retrieve CloudWatch metrics using the supplied filter, metrics name,
 /// and namespace.
 /// </summary>
 /// <param name="client">An initialized CloudWatch client.</param>
 /// <param name="filter">The filter to apply in retrieving metrics.</param>
 /// <param name="metricName">The metric name for which to retrieve
 /// information.</param>
 /// <param name="nameSpaceName">The name of the namespace from which
 /// to retrieve metric information.</param>
 public static async Task ListMetricsAsync(
 IAmazonCloudWatch client,
 DimensionFilter filter,
 string metricName,
 string nameSpaceName)
 {
 var request = new ListMetricsRequest
 {
 Dimensions = new List<DimensionFilter>() { filter },
 MetricName = metricName,
 Namespace = nameSpaceName,
 };

 var response = new ListMetricsResponse();
 do
 {
 response = await client.ListMetricsAsync(request);
 }
 }
}
```



```
 if (response.Metrics.Count > 0)
 {
 foreach (var metric in response.Metrics)
 {
 Console.WriteLine(metric.MetricName +
 " (" + metric.Namespace + ")");

 foreach (var dimension in metric.Dimensions)
 {
 Console.WriteLine(" " + dimension.Name + ": " +
 dimension.Value);
 }
 }
 }
 else
 {
 Console.WriteLine("No metrics found.");
 }

 request.NextToken = response.NextToken;
 }
 while (!string.IsNullOrEmpty(response.NextToken));
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for .NET API 参考中的 [ListMetrics](#)。

## C++

### SDK for C++

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

列出指标。

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::ListMetricsRequest request;

if (argc > 1)
{
 request.SetMetricName(argv[1]);
}

if (argc > 2)
{
 request.SetNamespace(argv[2]);
}
```

```
bool done = false;
bool header = false;
while (!done)
{
 auto outcome = cw.ListMetrics(request);
 if (!outcome.IsSuccess())
 {
 std::cout << "Failed to list CloudWatch metrics:" <<
 outcome.GetError().GetMessage() << std::endl;
 break;
 }

 if (!header)
 {
 std::cout << std::left << std::setw(48) << "MetricName" <<
 std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
 std::endl;
 header = true;
 }

 const auto &metrics = outcome.GetResult().GetMetrics();
 for (const auto &metric : metrics)
 {
 std::cout << std::left << std::setw(48) <<
 metric.GetMetricName() << std::setw(32) <<
 metric.GetNamespace();
 const auto &dimensions = metric.GetDimensions();
 for (auto iter = dimensions.cbegin();
 iter != dimensions.cend(); ++iter)
 {
 const auto &dimkv = *iter;
 std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
 if (iter + 1 != dimensions.cend())
 {
 std::cout << ", ";
 }
 }
 std::cout << std::endl;
 }

 const auto &next_token = outcome.GetResult().GetNextToken();
 request.SetNextToken(next_token);
 done = next_token.empty();
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for C++ API 参考中的 [ListMetrics](#)。

## Java

### SDK for Java 2.x

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void listMets(CloudWatchClient cw, String namespace) {

 boolean done = false;
 String nextToken = null;

 try {
 while(!done) {
```

```
ListMetricsResponse response;
if (nextToken == null) {
 ListMetricsRequest request = ListMetricsRequest.builder()
 .namespace(namespace)
 .build();

 response = cw.listMetrics(request);
} else {
 ListMetricsRequest request = ListMetricsRequest.builder()
 .namespace(namespace)
 .nextToken(nextToken)
 .build();

 response = cw.listMetrics(request);
}

for (Metric metric : response.metrics()) {
 System.out.printf("Retrieved metric %s", metric.metricName());
 System.out.println();
}

if(response.nextToken() == null) {
 done = true;
} else {
 nextToken = response.nextToken();
}
}

} catch (CloudWatchException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
}
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 [ListMetrics](#)。

## JavaScript

### SDK for JavaScript V3

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

在单独的模块中创建客户端并将其导出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

导入软件开发工具包和客户端模块，然后调用 API。

```
// Import required AWS SDK clients and commands for Node.js
import { ListMetricsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "../libs/cloudWatchClient.js";
```

```
// Set the parameters
export const params = {
 Dimensions: [
 {
 Name: "LogGroupName" /* required */,
 },
],
 MetricName: "IncomingLogEvents",
 Namespace: "AWS/Logs",
};

export const run = async () => {
 try {
 const data = await cwClient.send(new ListMetricsCommand(params));
 console.log("Success. Metrics:", JSON.stringify(data.Metrics));
 return data;
 } catch (err) {
 console.log("Error", err);
 }
};

// Uncomment this line to run execution within this file.
// run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 Amazon SDK for JavaScript API 参考中的 [ListMetrics](#)。

#### SDK for JavaScript V2

##### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
 Dimensions: [
 {
 Name: 'LogGroupName', /* required */
 },
],
 MetricName: 'IncomingLogEvents',
 Namespace: 'AWS/Logs'
};

cw.listMetrics(params, function(err, data) {
 if (err) {
 console.log("Error", err);
 } else {
 console.log("Metrics", JSON.stringify(data.Metrics));
 }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 Amazon SDK for JavaScript API 参考中的 [ListMetrics](#)。

## Python

适用于 Python (Boto3) 的 SDK

### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
class CloudWatchWrapper:
 """Encapsulates Amazon CloudWatch functions."""
 def __init__(self, cloudwatch_resource):
 """
 :param cloudwatch_resource: A Boto3 CloudWatch resource.
 """
 self.cloudwatch_resource = cloudwatch_resource

 def list_metrics(self, namespace, name, recent=False):
 """
 Gets the metrics within a namespace that have the specified name.
 If the metric has no dimensions, a single metric is returned.
 Otherwise, metrics for all dimensions are returned.

 :param namespace: The namespace of the metric.
 :param name: The name of the metric.
 :param recent: When True, only metrics that have been active in the last
 three hours are returned.
 :return: An iterator that yields the retrieved metrics.
 """
 try:
 kwargs = {'Namespace': namespace, 'MetricName': name}
 if recent:
 kwargs['RecentlyActive'] = 'PT3H' # List past 3 hours only
 metric_iter = self.cloudwatch_resource.metrics.filter(**kwargs)
 logger.info("Got metrics for %s.%s.", namespace, name)
 except ClientError:
 logger.exception("Couldn't get metrics for %s.%s.", namespace, name)
 raise
 else:
 return metric_iter
```

- 有关 API 详细信息，请参阅 Amazon 适用于 Python (Boto3) 的 SDK API 参考中的 [ListMetrics](#)。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 使用 Amazon SDK 将一组数据放入 CloudWatch 指标

以下代码示例展示如何将一组数据放入 Amazon CloudWatch 指标。

## Python

适用于 Python (Boto3) 的 SDK

### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
class CloudWatchWrapper:
 """Encapsulates Amazon CloudWatch functions."""
```

```
def __init__(self, cloudwatch_resource):
 """
 :param cloudwatch_resource: A Boto3 CloudWatch resource.
 """
 self.cloudwatch_resource = cloudwatch_resource

def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
 """
 Sends a set of data to CloudWatch for a metric. All of the data in the set
 have the same timestamp and unit.

 :param namespace: The namespace of the metric.
 :param name: The name of the metric.
 :param timestamp: The UTC timestamp for the metric.
 :param unit: The unit of the metric.
 :param data_set: The set of data to send. This set is a dictionary that
 contains a list of values and a list of corresponding
 counts.
 The value and count lists must be the same length.
 """
 try:
 metric = self.cloudwatch_resource.Metric(namespace, name)
 metric.put_data(
 Namespace=namespace,
 MetricData=[{
 'MetricName': name,
 'Timestamp': timestamp,
 'Values': data_set['values'],
 'Counts': data_set['counts'],
 'Unit': unit}])
 logger.info("Put data set for metric %s.%s.", namespace, name)
 except ClientError:
 logger.exception("Couldn't put data set for metric %s.%s.", namespace,
 name)
 raise
```

- 有关 API 详细信息，请参阅 Amazon 适用于 Python (Boto3) 的 SDK API 参考中的 [PutMetricData](#)。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 使用 Amazon SDK 将数据放入 CloudWatch 指标

以下代码示例显示如何将数据放入 Amazon CloudWatch 指标。

C++

适用于 C++ 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

将数据放入指标。

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("UNIQUE_PAGES");
dimension.SetValue("URLS");

Aws::CloudWatch::Model::MetricDatum datum;
datum.SetMetricName("PAGES_VISITED");
datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
datum.SetValue(data_point);
datum.AddDimensions(dimension);

Aws::CloudWatch::Model::PutMetricDataRequest request;
request.SetNamespace("SITE/TRAFFIC");
request.AddMetricData(datum);

auto outcome = cw.PutMetricData(request);
if (!outcome.IsSuccess())
{
 std::cout << "Failed to put sample metric data:" <<
 outcome.GetError().GetMessage() << std::endl;
}
else
{
 std::cout << "Successfully put sample metric data" << std::endl;
}
```

- 有关 API 详细信息，请参阅 [Amazon SDK for C++ API 参考](#) 中的 PutMetricData。

## Java

### SDK for Java 2.x

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void putMetData(CloudWatchClient cw, Double dataPoint) {

 try {
 Dimension dimension = Dimension.builder()
 .name("UNIQUE_PAGES")
 .value("URLS")
 .build();

 // Set an Instant object.
 String time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
 Instant instant = Instant.parse(time);

 MetricDatum datum = MetricDatum.builder()
 .metricName("PAGES_VISITED")
 .unit(StandardUnit.NONE)
 .value(dataPoint)
 .timestamp(instant)
 .dimensions(dimension).build();

 PutMetricDataRequest request = PutMetricDataRequest.builder()
 .namespace("SITE/TRAFFIC")
```

```
 .metricData(datum).build();

 cw.putMetricData(request);

 } catch (CloudWatchException e) {
 System.err.println(e.awsErrorDetails().errorMessage());
 System.exit(1);
 }
 System.out.printf("Successfully put data point %f", dataPoint);
}
```

- 有关 API 详细信息，请参阅 [Amazon SDK for Java 2.x API 参考](#) 中的 PutMetricData。

## JavaScript

### SDK for JavaScript V3

#### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

在单独的模块中创建客户端并将其导出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

导入软件开发工具包和客户端模块，然后调用 API。

```
// Import required AWS SDK clients and commands for Node.js
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "../libs/cloudWatchClient.js";

// Set the parameters
export const params = {
 MetricData: [
 {
 MetricName: "PAGES_VISITED",
 Dimensions: [
 {
 Name: "UNIQUE_PAGES",
 Value: "URLS",
 },
],
 Unit: "None",
 Value: 1.0,
 },
],
 Namespace: "SITE/TRAFFIC",
};

export const run = async () => {
 try {
 const data = await cwClient.send(new PutMetricDataCommand(params));
 console.log("Success", data.$metadata.requestId);
 return data;
 }
}
```



```
 } catch (err) {
 console.log("Error", err);
 }
 };
// Uncomment this line to run execution within this file.
// run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 [Amazon SDK for JavaScript API 参考](#) 中的 PutMetricData。

## SDK for JavaScript V2

### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

// Create parameters JSON for putMetricData
var params = {
 MetricData: [
 {
 MetricName: 'PAGES_VISITED',
 Dimensions: [
 {
 Name: 'UNIQUE_PAGES',
 Value: 'URLS'
 }
],
 Unit: 'None',
 Value: 1.0
 }
],
 Namespace: 'SITE/TRAFFIC'
};

cw.putMetricData(params, function(err, data) {
 if (err) {
 console.log("Error", err);
 } else {
 console.log("Success", JSON.stringify(data));
 }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅 [Amazon SDK for JavaScript API 参考](#) 中的 PutMetricData。

## Python

### 适用于 Python (Boto3) 的 SDK

### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
class CloudWatchWrapper:
 """Encapsulates Amazon CloudWatch functions."""
 def __init__(self, cloudwatch_resource):
 """
 :param cloudwatch_resource: A Boto3 CloudWatch resource.
 """
 self.cloudwatch_resource = cloudwatch_resource

 def put_metric_data(self, namespace, name, value, unit):
 """
 Sends a single data value to CloudWatch for a metric. This metric is given
 a timestamp of the current UTC time.

 :param namespace: The namespace of the metric.
 :param name: The name of the metric.
 :param value: The value of the metric.
 :param unit: The unit of the metric.
 """
 try:
 metric = self.cloudwatch_resource.Metric(namespace, name)
 metric.put_data(
 Namespace=namespace,
 MetricData=[{
 'MetricName': name,
 'Value': value,
 'Unit': unit
 }]
)
 logger.info("Put data for metric %s.%s", namespace, name)
 except ClientError:
 logger.exception("Couldn't put data for metric %s.%s", namespace, name)
 raise
```

- 有关 API 详细信息，请参阅 Amazon 适用于 Python (Boto3) 的 SDK API 参考中的 [PutMetricData](#)。

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

## 使用 Amazon SDK 的 CloudWatch 场景

以下代码示例向您展示如何通过 Amazon SDK 实施 CloudWatch 中的常见场景。这些场景向您展示了如何通过 CloudWatch 中调用多个函数来完成特定任务。每个场景都包含一个指向 GitHub 的链接，其中包含了有关如何设置和运行代码的说明。

示例

- [使用 Amazon SDK 管理 CloudWatch 指标和告警 \(p. 830\)](#)

## 使用 Amazon SDK 管理 CloudWatch 指标和告警

以下代码示例显示了如何：

- 创建监控指标的告警。
- 将数据放入 CloudWatch 指标并触发告警。
- 从告警中获取数据。

- 删除告警。

## Python

适用于 Python (Boto3) 的 SDK

### Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建一个包装 CloudWatch 操作的类。

```
from datetime import datetime, timedelta
import logging
from pprint import pprint
import random
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class CloudWatchWrapper:
 """Encapsulates Amazon CloudWatch functions."""
 def __init__(self, cloudwatch_resource):
 """
 :param cloudwatch_resource: A Boto3 CloudWatch resource.
 """
 self.cloudwatch_resource = cloudwatch_resource

 def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
 """
 Sends a set of data to CloudWatch for a metric. All of the data in the set
 have the same timestamp and unit.

 :param namespace: The namespace of the metric.
 :param name: The name of the metric.
 :param timestamp: The UTC timestamp for the metric.
 :param unit: The unit of the metric.
 :param data_set: The set of data to send. This set is a dictionary that
 contains a list of values and a list of corresponding
 counts.
 The value and count lists must be the same length.
 """
 try:
 metric = self.cloudwatch_resource.Metric(namespace, name)
 metric.put_data(
 Namespace=namespace,
 MetricData=[{
 'MetricName': name,
 'Timestamp': timestamp,
 'Values': data_set['values'],
 'Counts': data_set['counts'],
 'Unit': unit}]
)
 logger.info("Put data set for metric %s.%s.", namespace, name)
 except ClientError:
 logger.exception("Couldn't put data set for metric %s.%s.", namespace,
 name)
 raise

 def create_metric_alarm(
 self, metric_namespace, metric_name, alarm_name, stat_type, period,
 eval_periods, threshold, comparison_op):
```

```

"""
Creates an alarm that watches a metric.

:param metric_namespace: The namespace of the metric.
:param metric_name: The name of the metric.
:param alarm_name: The name of the alarm.
:param stat_type: The type of statistic the alarm watches.
:param period: The period in which metric data are grouped to calculate
 statistics.
:param eval_periods: The number of periods that the metric must be over the
 alarm threshold before the alarm is set into an
alarmed
 state.
:param threshold: The threshold value to compare against the metric
statistic.
:param comparison_op: The comparison operation used to compare the
threshold
 against the metric.
:return: The newly created alarm.
"""
try:
 metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
 alarm = metric.put_alarm(
 AlarmName=alarm_name,
 Statistic=stat_type,
 Period=period,
 EvaluationPeriods=eval_periods,
 Threshold=threshold,
 ComparisonOperator=comparison_op)
 logger.info(
 "Added alarm %s to track metric %s.%s.", alarm_name,
metric_namespace,
 metric_name)
 except ClientError:
 logger.exception(
 "Couldn't add alarm %s to metric %s.%s", alarm_name,
metric_namespace,
 metric_name)
 raise
 else:
 return alarm

def put_metric_data(self, namespace, name, value, unit):
 """
 Sends a single data value to CloudWatch for a metric. This metric is given
 a timestamp of the current UTC time.

 :param namespace: The namespace of the metric.
 :param name: The name of the metric.
 :param value: The value of the metric.
 :param unit: The unit of the metric.
 """
 try:
 metric = self.cloudwatch_resource.Metric(namespace, name)
 metric.put_data(
 Namespace=namespace,
 MetricData=[{
 'MetricName': name,
 'Value': value,
 'Unit': unit
 }]
)
 logger.info("Put data for metric %s.%s", namespace, name)
 except ClientError:
 logger.exception("Couldn't put data for metric %s.%s", namespace, name)
 raise

```

```

def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
 """
 Gets statistics for a metric within a specified time span. Metrics are
grouped
into the specified period.

:param namespace: The namespace of the metric.
:param name: The name of the metric.
:param start: The UTC start time of the time span to retrieve.
:param end: The UTC end time of the time span to retrieve.
:param period: The period, in seconds, in which to group metrics. The
period
must match the granularity of the metric, which depends on
the metric's age. For example, metrics that are older than
three hours have a one-minute granularity, so the period
must
be at least 60 and must be a multiple of 60.
:param stat_types: The type of statistics to retrieve, such as average
value
or maximum value.
:return: The retrieved statistics for the metric.
 """
 try:
 metric = self.cloudwatch_resource.Metric(namespace, name)
 stats = metric.get_statistics(
 StartTime=start, EndTime=end, Period=period, Statistics=stat_types)
 logger.info(
 "Got %s statistics for %s.", len(stats['Datapoints']),
stats['Label'])
 except ClientError:
 logger.exception("Couldn't get statistics for %s.%s.", namespace, name)
 raise
 else:
 return stats

def get_metric_alarms(self, metric_namespace, metric_name):
 """
 Gets the alarms that are currently watching the specified metric.

:param metric_namespace: The namespace of the metric.
:param metric_name: The name of the metric.
:returns: An iterator that yields the alarms.
 """
 metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
 alarm_iter = metric.alarms.all()
 logger.info("Got alarms for metric %s.%s.", metric_namespace, metric_name)
 return alarm_iter

def delete_metric_alarms(self, metric_namespace, metric_name):
 """
 Deletes all of the alarms that are currently watching the specified metric.

:param metric_namespace: The namespace of the metric.
:param metric_name: The name of the metric.
 """
 try:
 metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
 metric.alarms.delete()
 logger.info(
 "Deleted alarms for metric %s.%s.", metric_namespace, metric_name)
 except ClientError:
 logger.exception(
 "Couldn't delete alarms for metric %s.%s.", metric_namespace,
metric_name)

```

```
raise
```

使用包装类将数据放入指标中，触发监控指标的告警，然后从告警中获取数据。

```
def usage_demo():
 print('-'*88)
 print("Welcome to the Amazon CloudWatch metrics and alarms demo!")
 print('-'*88)

 logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

 cw_wrapper = CloudWatchWrapper(boto3.resource('cloudwatch'))

 minutes = 20
 metric_namespace = 'doc-example-metric'
 metric_name = 'page_views'
 start = datetime.utcnow() - timedelta(minutes=minutes)
 print(f"Putting data into metric {metric_namespace}.{metric_name} spanning the
"
 f"last {minutes} minutes.")
 for offset in range(0, minutes):
 stamp = start + timedelta(minutes=offset)
 cw_wrapper.put_metric_data_set(
 metric_namespace, metric_name, stamp, 'Count', {
 'values': [
 random.randint(bound, bound * 2)
 for bound in range(offset + 1, offset + 11)],
 'counts': [random.randint(1, offset + 1) for _ in range(10)]
 })

 alarm_name = 'high_page_views'
 period = 60
 eval_periods = 2
 print(f"Creating alarm {alarm_name} for metric {metric_name}.")
 alarm = cw_wrapper.create_metric_alarm(
 metric_namespace, metric_name, alarm_name, 'Maximum', period, eval_periods,
 100, 'GreaterThanThreshold')
 print(f"Alarm ARN is {alarm.alarm_arn}.")
 print(f"Current alarm state is: {alarm.state_value}.")

 print(f"Sending data to trigger the alarm. This requires data over the
threshold "
 f"for {eval_periods} periods of {period} seconds each.")
 while alarm.state_value == 'INSUFFICIENT_DATA':
 print("Sending data for the metric.")
 cw_wrapper.put_metric_data(
 metric_namespace, metric_name, random.randint(100, 200), 'Count')
 alarm.load()
 print(f"Current alarm state is: {alarm.state_value}.")
 if alarm.state_value == 'INSUFFICIENT_DATA':
 print(f"Waiting for {period} seconds...")
 time.sleep(period)
 else:
 print("Wait for a minute for eventual consistency of metric data.")
 time.sleep(period)
 if alarm.state_value == 'OK':
 alarm.load()
 print(f"Current alarm state is: {alarm.state_value}.")

 print(f"Getting data for metric {metric_namespace}.{metric_name} during
timespan "
 f"of {start} to {datetime.utcnow()} (times are UTC).")
 stats = cw_wrapper.get_metric_statistics(
 metric_namespace, metric_name, start, datetime.utcnow(), 60,
```

```
 ['Average', 'Minimum', 'Maximum'])
print(f"Got {len(stats['Datapoints'])} data points for metric "
 f"{metric_namespace}.{metric_name}.")
pprint(sorted(stats['Datapoints'], key=lambda x: x['Timestamp']))

print(f"Getting alarms for metric {metric_name}.")
alarms = cw_wrapper.get_metric_alarms(metric_namespace, metric_name)
for alarm in alarms:
 print(f"Alarm {alarm.name} is currently in state {alarm.state_value}.")

print(f"Deleting alarms for metric {metric_name}.")
cw_wrapper.delete_metric_alarms(metric_namespace, metric_name)

print("Thanks for watching!")
print('-'*88)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的以下主。
  - [删除警报](#)
  - [DescribeAlarmsForMetric](#)
  - [DisableAlarmActions](#)
  - [EnableAlarmActions](#)
  - [GetMetricStatistics](#)
  - [ListMetrics](#)
  - [PutMetricAlarm](#)
  - [PutMetricData](#)

有关 Amazon 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch 与 Amazon SDK 结合使用 \(p. 788\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

# Amazon CloudWatch 中的安全性

Amazon 十分重视云安全性。作为 Amazon 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是 Amazon 和您的共同责任。[责任共担模式](#)将其描述为云的安全性和云中的安全性：

- 云的安全性 – Amazon 负责保护在 Amazon 云中运行 Amazon 服务的基础设施。Amazon 还向您提供可安全使用的服务。第三方审核员定期测试和验证我们的安全性的有效性，作为 [Amazon Compliance Programs](#) 的一部分。要了解适用于 CloudWatch 的合规性计划，请参阅[合规性计划范围内的 Amazon 服务](#)。
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Amazon CloudWatch 时应用责任共担模式。它介绍了如何配置 Amazon CloudWatch 以实现您的安全性和合规性目标。您还将了解如何使用其他 Amazon 服务来帮助您的 CloudWatch 资源。

## 目录

- [Amazon CloudWatch 中的数据保护 \(p. 836\)](#)
- [适用于 Amazon CloudWatch 的 Identity and Access Management \(p. 837\)](#)
- [Amazon CloudWatch 的合规性验证 \(p. 890\)](#)
- [Amazon CloudWatch 中的恢复能力 \(p. 890\)](#)
- [Amazon CloudWatch 中的基础设施安全性 \(p. 890\)](#)
- [将 CloudWatch 和 CloudWatch Synthetics 与接口 VPC 终端节点结合使用 \(p. 891\)](#)
- [Synthetics 金丝雀的安全注意事项 \(p. 894\)](#)

## Amazon CloudWatch 中的数据保护

Amazon [责任共担模式](#) 适用于 Amazon CloudWatch 中的数据保护。如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的 Amazon Web Services 的安全配置和管理任务。有关数据隐私的更多信息，请参阅[数据隐私常见问题](#)。

出于数据保护目的，我们建议您保护 Amazon Web Services 账户 凭证并使用 Amazon Identity and Access Management (IAM) 设置单独的用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 Amazon 资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 Amazon CloudTrail 设置 API 和用户活动日志记录。
- 使用 Amazon 加密解决方案以及 Amazon 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Simple Storage Service (Amazon S3) 中的个人数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅 [《美国联邦信息处理标准 \(FIPS\) 第 140-2 版》](#)。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括在通过控制台、API、Amazon CLI 或 Amazon 开发工具包使用 CloudWatch 或其他



Amazon 服务时。您在用于名称的标签或自由格式字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

## 传输中加密

CloudWatch 为传输中的数据使用端到端加密。

# 适用于 Amazon CloudWatch 的 Identity and Access Management

访问 Amazon CloudWatch 需要凭证。这些凭证必须有权访问 Amazon 资源，例如检索有关您的云资源的 CloudWatch 指标数据。下面几节提供详细信息，说明如何使用 [Amazon Identity and Access Management \(IAM\)](#) 和 CloudWatch 通过控制可以访问资源的对象来保护资源：

- [身份验证 \(p. 837\)](#)
- [访问控制 \(p. 838\)](#)

## 身份验证

您可以以下面任一类型的身份访问 Amazon：

- Amazon 账户根用户 – 注册 Amazon 时，您需要提供与您的 Amazon 账户关联的电子邮件地址和密码。这些是您的 Amazon 账户用户根凭证，它们提供对您所有 Amazon 资源的完全访问权限。

### Important

出于安全考虑，我们建议您仅使用 Amazon 账户用户凭证创建管理员，该用户是对您的账户具有完全访问权限的 IAM 用户。随后，您可以使用此管理员来创建具有有限权限的其他 IAM 用户和角色。有关更多信息，请参阅 IAM 用户指南中的 [IAM 最佳实践](#) 和 [创建管理员用户和组](#)。

- IAM 用户 – [IAM 用户](#) 是您的 Amazon 账户中具有特定自定义权限（例如，用于在 CloudWatch 中查看指标的权限）的身份。您可以使用 IAM 用户名和密码登录以保护 Amazon 网页（如 [Amazon Web Services Management Console](#)、[Amazon 开发论坛](#) 或 [Amazon Web Services Support 中心](#)）。

除了用户名和密码之外，您还可以为每个用户生成 [访问密钥](#)。在通过 [几个 SDK 之一](#) 或使用 [Amazon Command Line Interface \(CLI\)](#) 以编程方式访问 Amazon 服务时，可以使用这些密钥。软件开发工具包和 Amazon CLI 工具使用访问密钥对您的请求进行加密签名。如果您不使用 Amazon 工具，则必须自行对请求签名。CloudWatch 支持签名版本 4（一种用于对入站 API 请求进行身份验证的协议）。有关验证请求的更多信息，请参阅《Amazon 一般参考》中的 [签名版本 4 签名流程](#)。

- IAM 角色 – [IAM 角色](#) 是可在账户中创建的另一种具有特定权限的 IAM 身份。它类似于 IAM 用户，但未与特定人员关联。利用 IAM 角色，可以获得临时访问密钥，用于访问 Amazon 服务和资源。具有临时凭证的 IAM 角色在以下情况下很有用：
- 联合身份用户访问 – 您可以不创建 IAM 用户，而是使用来自 Amazon Directory Service、您的企业用户目录或 Web 身份提供商 (IdP) 的既有身份。这些用户被称为联合身份用户。在通过 [IdP](#) 请求访问权限时，Amazon 将为联合身份用户分配角色。有关更多信息，请参阅 IAM 用户指南中的 [联合身份用户和角色](#)。

- 跨账户访问 – 您可以使用您账户中的 IAM 角色向另一个 Amazon 账户授予对您账户的资源的访问权限。有关示例，请参阅 IAM 用户指南中的[教程：使用 IAM 角色委派跨 Amazon 账户的访问权限](#)。
- Amazon 服务访问 – 可以使用您账户中的 IAM 角色向 Amazon 服务授予访问您账户的资源所需的权限。例如，您可以创建一个角色以允许 Amazon Redshift 代表您访问 Amazon S3 存储桶，然后将存储在存储桶中的数据加载到 Amazon Redshift 集群中。有关更多信息，请参阅 IAM 用户指南中的[创建向 Amazon 服务委派权限的角色](#)。
- 在 Amazon EC2 上运行的应用程序 – 您不用将访问密钥存储在 EC2 实例中以供实例上运行的应用程序使用并发出 API 请求，而是可以使用 IAM 角色管理这些应用程序的临时凭证。要将 Amazon 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南中的[对 Amazon EC2 上的应用程序使用角色](#)。

## 访问控制

您可以使用有效的凭证来对自己的请求进行身份验证，但您还必须拥有权限才能创建或访问 CloudWatch 资源。例如，您必须拥有权限才能创建 CloudWatch 控制面板组件、查看指标，等等。

下面几节介绍如何管理 CloudWatch 的权限。我们建议您先阅读概述。

- [管理您的 CloudWatch 资源的访问权限概览 \(p. 839\)](#)
- [对 CloudWatch 使用基于身份的策略 \(IAM 策略\) \(p. 841\)](#)
- [Amazon CloudWatch 权限参考 \(p. 882\)](#)

## CloudWatch 控制面板权限更新

从 2018 年 5 月 1 日起，Amazon 改变了访问 CloudWatch 控制面板所需的权限。现在，CloudWatch 控制台的控制面板访问需要在 2017 年推出的权限，以支持控制面板 API 操作：

- cloudwatch:GetDashboard
- cloudwatch:ListDashboards
- cloudwatch:PutDashboard
- cloudwatch:DeleteDashboards

要访问 CloudWatch 控制面板，您需要以下项目之一：

- AdministratorAccess 策略。
- CloudWatchFullAccess 策略。
- 包含以下一个或多个特定权限的自定义策略：
  - cloudwatch:GetDashboard 和 cloudwatch:ListDashboards 能够查看控制面板
  - cloudwatch:PutDashboard 能够创建或修改控制面板
  - cloudwatch:DeleteDashboards 能够删除控制面板

有关使用策略更改 IAM 用户权限的更多信息，请参阅[更改 IAM 用户的权限](#)。

有关 CloudWatch 权限的更多信息，请参阅[Amazon CloudWatch 权限参考 \(p. 882\)](#)。

有关控制面板 API 操作的更多信息，请参阅《Amazon CloudWatch API 参考》中的 [PutDashboard](#)。

## 管理您的 CloudWatch 资源的访问权限概览

每个 Amazon 资源都归某个 Amazon Web Services 账户 账户所有，创建或访问资源的权限由权限策略进行管理。账户管理员可以向 IAM 身份（即：用户、组和角色）附加权限策略，某些服务（如 Amazon Lambda）也支持向资源附加权限策略。

### Note

账户管理员（或管理员 IAM 用户）是具有管理员权限的用户。有关更多信息，请参阅 IAM 用户指南中的 [IAM 最佳实践](#)。

在授予权限时，您要决定谁获得权限，获得对哪些资源的权限，以及您允许对这些资源执行的具体操作。

### 主题

- [CloudWatch 资源和操作](#) (p. 839)
- [了解资源所有权](#) (p. 840)
- [管理对资源的访问](#) (p. 840)
- [指定策略元素：操作、效果和委托人](#) (p. 841)
- [在策略中指定条件](#) (p. 841)

## CloudWatch 资源和操作

您可以通过在策略中使用 Amazon 资源名称 (ARN) 来限制对特定警报和控制面板的访问。有关更多信息，请参阅 IAM 用户指南中的 [Amazon CloudWatch 的操作、资源和条件键](#)。

当您编写策略来控制对 CloudWatch 操作的访问权限时，请使用 \*(星号) 作为资源。例如：

```
{
 "Version": "2012-10-17",
 "Statement": [{
 "Effect": "Allow",
 "Action": ["cloudwatch:GetMetricData", "cloudwatch:ListMetrics"],
 "Resource": "*",
 "Condition": {
 "Bool": {
 "aws:SecureTransport": "true"
 }
 }
 }]
}
```

有关 ARN 的更多信息，请参阅 IAM 用户指南中的 [ARN](#)。有关 CloudWatch Logs ARN 的更多信息，请参阅 Amazon Web Services 一般参考中的 [Amazon Resource Name \(ARN\)](#) 和 [Amazon 服务命名空间](#)。要查看包含 CloudWatch 操作的策略示例，请参阅 [对 CloudWatch 使用基于身份的策略 \(IAM 策略\)](#) (p. 841)。

| 操作   | ARN ( 包含区域 )                        | ARN ( 与 IAM 角色一起使用 )                                                                       |
|------|-------------------------------------|--------------------------------------------------------------------------------------------|
| Stop | arn:aws:automate:us-east-1:ec2:stop | arn:aws:swf:us-east-1: <i>customer-account</i> :action/actions/AWS_EC2.InstanceId.Stop/1.0 |

| 操作        | ARN ( 包含区域 )                             | ARN ( 与 IAM 角色一起使用 )                                                                            |
|-----------|------------------------------------------|-------------------------------------------------------------------------------------------------|
| Terminate | arn:aws:automate:us-east-1:ec2:terminate | arn:aws:swf:us-east-1: <b>customer-account</b> :action/actions/AWS_EC2.InstanceId.Terminate/1.0 |
| Reboot    | arn:aws:automate:us-east-1:ec2:reboot    | arn:aws:swf:us-east-1: <b>customer-account</b> :action/actions/AWS_EC2.InstanceId.Reboot/1.0    |
| Recover   | arn:aws:automate:us-east-1:ec2:recover   | arn:aws:swf:us-east-1: <b>customer-account</b> :action/actions/AWS_EC2.InstanceId.Recover/1.0   |

## 了解资源所有权

Amazon 账户对在该账户中创建的资源具有所有权，而无论创建资源的人员是谁。具体而言，资源所有者是对资源创建请求进行身份验证的**委托人实体**（即 Amazon 账户根用户、IAM 用户或 IAM 角色）的 Amazon 账户。CloudWatch 没有您可以拥有的任何资源。

## 管理对资源的访问

权限策略规定谁可以访问哪些内容。下一节介绍创建权限策略时的可用选项。

### Note

本节讨论如何在 CloudWatch 范围内使用 IAM。这里不提供有关 IAM 服务的详细信息。有关完整的 IAM 文档，请参阅 IAM 用户指南中的[什么是 IAM？](#)。有关 IAM policy 语法和说明的信息，请参阅 IAM 用户指南中的[IAM policy 参考](#)。

附加到 IAM 身份的策略称作基于身份的策略（IAM policy），附加到资源的策略称作基于资源的策略。CloudWatch 只支持基于身份的策略。

### 主题

- [基于身份的策略（IAM policy）](#) (p. 840)
- [基于资源的策略（IAM 策略）](#) (p. 841)

## 基于身份的策略（IAM policy）

您可以向 IAM 身份附加策略。例如，您可以执行以下操作：

- 将权限策略附加到账户中的用户或组 – 要向一个或一组用户授予创建 Amazon CloudWatch 资源（如指标）的权限，可以将权限策略附加到用户或用户所属的组。
- 向角色附加权限策略（授予跨账户权限） – 您可以向 IAM 角色附加基于身份的权限策略，以授予跨账户的权限。例如，账户 A 中的管理员可以创建一个角色，以向其他 Amazon Web Services 账户（如账户 B）或某个 Amazon Web Service 授予跨账户权限，如下所述：
  1. 账户 A 管理员可以创建一个 IAM 角色，然后向该角色附加授予其访问账户 A 中资源的权限策略。
  2. 账户 A 管理员可以向角色挂载信任策略，将账户 B 标识为能够担任该角色的委托人。
  3. 之后，账户 B 管理员可以委托权限，指派账户 B 中的任何用户代入该角色。这样，账户 B 中的用户就可以在账户 A 中创建或访问资源了。如果您需要授予 Amazon Web Service 权限来代入该角色，则信任策略中的主体也可以是 Amazon Web Service 主体。

有关使用 IAM 委派权限的更多信息，请参阅 [IAM 用户指南](#) 中的访问权限管理。

有关对 CloudWatch 使用基于身份的策略的更多信息，请参阅 [对 CloudWatch 使用基于身份的策略 \( IAM 策略 \)](#) (p. 841)。有关用户、组、角色和权限的更多信息，请参阅 [IAM 用户指南](#) 中的身份 ( 用户、组和角色 )。

## 基于资源的策略 ( IAM 策略)

其他服务 [如 Simple Storage Service ( Amazon S3 ) ] 还支持基于资源的权限策略。例如，您可以将策略附加到 Amazon S3 存储桶以管理对该存储桶的访问权限。CloudWatch 不支持基于资源的策略。

## 指定策略元素：操作、效果和委托人

对于每种 CloudWatch 资源，该服务都定义了一组 API 操作。为授予执行这些 API 操作的权限，CloudWatch 定义了一组您可以在策略中指定的操作。某些 API 操作可能需要多个操作的权限才能执行 API 操作。有关资源和 API 操作的更多信息，请参阅 [CloudWatch 资源和操作](#) (p. 839) 和 [CloudWatch 操作](#)。

以下是基本的策略元素：

- Resource – 使用 Amazon Resource Name (ARN) 指定策略应用到的资源。CloudWatch 没有可使用策略资源控制的任何资源，因此，请在 IAM 策略中使用通配符 (\*)。有关更多信息，请参阅 [CloudWatch 资源和操作](#) (p. 839)。
- 操作 – 使用操作关键字指定要允许或拒绝的资源操作。例如，cloudwatch:ListMetrics 权限允许执行 ListMetrics 操作的用户权限。
- 效果 – 用于指定用户请求特定操作时的效果 ( 可以是允许或拒绝 )。如果没有显式授予 ( 允许 ) 对资源的访问权限，则隐式拒绝访问。您也可显式拒绝对资源的访问，这样可确保用户无法访问该资源，即使有其他策略授予了访问权限的情况下也是如此。
- 主体 – 在基于身份的策略 ( IAM policy ) 中，附加了策略的用户是隐式主体。对于基于资源的策略，您可以指定要接收权限的用户、账户、服务或其他实体 ( 仅适用于基于资源的策略 )。CloudWatch 不支持基于资源的策略。

要了解 IAM policy 语法和说明的更多信息，请参阅 IAM 用户指南中的 [Amazon IAM JSON 策略参考](#)。

有关显示所有 CloudWatch API 操作及其适用资源的表，请参阅 [Amazon CloudWatch 权限参考](#) (p. 882)。

## 在策略中指定条件

当您授予权限时，可使用访问策略语言来指定规定策略何时生效的条件。例如，您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息，请参阅 IAM 用户指南中的 [条件](#)。

要表示条件，您可以使用预定义的条件键。有关每个 Amazon 服务支持的上下文键列表以及 Amazon 范围的策略键列表，请参阅 IAM 用户指南中的 [Amazon 服务的操作、资源和条件键](#) 和 [全局和 IAM 条件上下文键](#)。

## 对 CloudWatch 使用基于身份的策略 ( IAM 策略 )

本主题提供了基于身份的策略的示例，这些示例展示了账户管理员如何将权限策略附加到 IAM 身份 ( 即用户、组和角色 )，从而授予对 CloudWatch 资源执行操作的权限。

### Important

我们建议您首先阅读以下介绍性主题，这些主题讲解了管理 CloudWatch 资源访问的基本概念和选项。有关更多信息，请参阅 [访问控制](#) (p. 838)。

本主题的各个部分涵盖以下内容：

- [使用 CloudWatch 控制台所需的权限 \(p. 842\)](#)
- [用于 CloudWatch 的 Amazon 托管式 \( 预定义 \) 策略 \(p. 845\)](#)
- [客户托管式策略示例 \(p. 857\)](#)

下面介绍权限策略示例。

```
{
 "Version": "2012-10-17",
 "Statement": [{
 "Effect": "Allow",
 "Action": ["cloudwatch:GetMetricData", "cloudwatch:ListMetrics"],
 "Resource": "*",
 "Condition": {
 "Bool": {
 "aws:SecureTransport": "true"
 }
 }
 }]
}
```

本示例策略包含一个语句，该语句向一个组授予执行两个 CloudWatch 操作 ( `cloudwatch:GetMetricData` 和 `cloudwatch:ListMetrics` ) 的权限，但前提是该组对请求使用 SSL ( `"aws:SecureTransport": "true"` )。有关 IAM policy 语句中各元素的更多信息，请参阅 IAM 用户指南中的 [指定策略元素：操作、效果和委托人 \(p. 841\)](#) 和 [IAM policy 元素参考](#)。

## 使用 CloudWatch 控制台所需的权限

用户若要能够使用 CloudWatch 控制台，则必须拥有一组最低的权限来允许其描述自己账户中的其他 Amazon 资源。CloudWatch 控制台需要来自以下服务的权限：

- Amazon EC2 Auto Scaling
- CloudTrail
- CloudWatch
- CloudWatch Events
- CloudWatch Logs
- Amazon EC2
- OpenSearch Service
- IAM
- Kinesis
- Lambda
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon SWF
- X-Ray ( 如果您使用的是 ServiceLens 功能 )

如果创建比必需的最低权限更为严格的 IAM policy，对于附加了该 IAM policy 的用户，控制台将无法按预期正常运行。要确保这些用户仍可使用 CloudWatch 控制台，也可向用户附加 `CloudWatchReadOnlyAccess` 托管式策略，如 [用于 CloudWatch 的 Amazon 托管式 \( 预定义 \) 策略 \(p. 845\)](#) 中所述。

对于只需要调用 Amazon CLI 或 CloudWatch API 的用户，无需为其提供最低控制台权限。

下面列出了使用 CloudWatch 控制台所需的一整套权限：

- application-autoscaling:DescribeScalingPolicies
- autoscaling:DescribeAutoScalingGroups
- autoscaling:DescribePolicies
- cloudtrail:DescribeTrails
- cloudwatch:DeleteAlarms
- cloudwatch:DescribeAlarmHistory
- cloudwatch:DescribeAlarms
- cloudwatch:GetMetricData
- cloudwatch:GetMetricStatistics
- cloudwatch:ListMetrics
- cloudwatch:PutMetricAlarm
- cloudwatch:PutMetricData
- ec2:DescribeInstances
- ec2:DescribeTags
- ec2:DescribeVolumes
- es:DescribeElasticsearchDomain
- es:ListDomainNames
- events:DeleteRule
- events:DescribeRule
- events:DisableRule
- events:EnableRule
- events:ListRules
- events:PutRule
- iam:AttachRolePolicy
- iam:CreateRole
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:ListAttachedRolePolicies
- iam:ListRoles
- kinesis:DescribeStream
- kinesis:ListStreams
- lambda:AddPermission
- lambda:CreateFunction
- lambda:GetFunctionConfiguration
- lambda:ListAliases
- lambda:ListFunctions
- lambda:ListVersionsByFunction
- lambda:RemovePermission
- logs:CancelExportTask
- logs:CreateExportTask
- logs:CreateLogGroup

- logs:CreateLogStream
- logs>DeleteLogGroup
- logs>DeleteLogStream
- logs>DeleteMetricFilter
- logs>DeleteRetentionPolicy
- logs>DeleteSubscriptionFilter
- logs:DescribeExportTasks
- logs:DescribeLogGroups
- logs:DescribeLogStreams
- logs:DescribeMetricFilters
- logs:DescribeQueries
- logs:DescribeSubscriptionFilters
- logs:FilterLogEvents
- logs:GetLogGroupFields
- logs:GetLogRecord
- logs:GetLogEvents
- logs:GetQueryResults
- logs:PutMetricFilter
- logs:PutRetentionPolicy
- logs:PutSubscriptionFilter
- logs:StartQuery
- logs:StopQuery
- logs:TestMetricFilter
- s3:CreateBucket
- s3:ListBucket
- sns:CreateTopic
- sns:GetTopicAttributes
- sns:ListSubscriptions
- sns:ListTopics
- sns:SetTopicAttributes
- sns:Subscribe
- sns:Unsubscribe
- sqs:GetQueueAttributes
- sqs:GetQueueUrl
- sqs:ListQueues
- sqs:SetQueueAttributes
- swf:CreateAction
- swf:DescribeAction
- swf:ListActionTemplates
- swf:RegisterAction
- swf:RegisterDomain
- swf:UpdateAction

此外，要查看 ServiceLens 中的服务地图，您需要 `AWSXrayReadOnlyAccess`



## 用于 CloudWatch 的 Amazon 托管式 ( 预定义 ) 策略

Amazon 通过提供由 Amazon 创建和管理的独立 IAM policy 来满足许多常用案例的要求。这些 Amazon 托管策略可针对常用案例授予必要的权限，使您免去调查所需权限的工作。有关更多信息，请参阅《IAM 用户指南》中的 [Amazon 托管式策略](#)。

以下 Amazon 托管式策略 ( 可附加到您账户中的用户 ) 特定于 CloudWatch。

### Note

您可以通过登录到 IAM 控制台并在该控制台中搜索特定策略来查看这些权限策略。

此外，您还可以创建您自己的自定义 IAM 策略，以授予对 CloudWatch 操作和资源的相关权限。您可以将这些自定义策略附加到需要这些权限的 IAM 用户或组。

### 主题

- [CloudWatchFullAccess](#) (p. 845)
- [CloudWatchActionsEC2Access](#) (p. 846)
- [CloudWatchReadOnlyAccess](#) (p. 846)
- [CloudWatchAutomaticDashboardsAccess](#) (p. 846)
- [CloudWatchAgentServerPolicy](#) (p. 847)
- [CloudWatchAgentAdminPolicy](#) (p. 848)
- [用于 CloudWatch Synthetics 的 Amazon 托管式 \( 预定义 \) 策略](#) (p. 849)
- [适用于 Amazon CloudWatch RUM 的 Amazon 托管式 \( 预定义 \) 策略](#) (p. 853)
- [适用于 CloudWatch Evidently 的 Amazon 托管式 \( 预定义 \) 策略](#) (p. 855)
- [适用于 Amazon Systems Manager Incident Manager 的 Amazon 托管式策略](#) (p. 857)

## CloudWatchFullAccess

CloudWatchFullAccess 策略授予对所有 CloudWatch 操作和资源的完全访问权限。其包含以下内容：

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "autoscaling:Describe*",
 "cloudwatch:*",
 "logs:*",
 "sns:*",
 "iam:GetPolicy",
 "iam:GetPolicyVersion",
 "iam:GetRole"
],
 "Effect": "Allow",
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": "iam:CreateServiceLinkedRole",
 "Resource": "arn:aws:iam::*:role/aws-service-role/events.amazonaws.com/AWSServiceRoleForCloudWatchEvents*",
 "Condition": {
 "StringLike": {
 "iam:AWSServiceName": "events.amazonaws.com"
 }
 }
 }
]
}
```

```
]
}
```

## CloudWatchActionsEC2Access

CloudWatchActionsEC2Access 策略授予对 CloudWatch 告警和指标，以及 Amazon EC2 元数据的只读访问权限。它还授予对 EC2 实例的停止、终止和重启 API 操作的访问权限。

CloudWatchActionsEC2Access 策略的内容如下。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:Describe*",
 "ec2:Describe*",
 "ec2:RebootInstances",
 "ec2:StopInstances",
 "ec2:TerminateInstances"
],
 "Resource": "*"
 }
]
}
```

## CloudWatchReadOnlyAccess

CloudWatchReadOnlyAccess 策略授予对 CloudWatch 的只读访问权限。

CloudWatchReadOnlyAccess 策略的内容如下。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "autoscaling:Describe*",
 "cloudwatch:Describe*",
 "cloudwatch:Get*",
 "cloudwatch:List*",
 "logs:Get*",
 "logs:List*",
 "logs:StartQuery",
 "logs:StopQuery",
 "logs:Describe*",
 "logs:TestMetricFilter",
 "logs:FilterLogEvents",
 "sns:Get*",
 "sns:List*"
],
 "Effect": "Allow",
 "Resource": "*"
 }
]
}
```

## CloudWatchAutomaticDashboardsAccess

CloudWatch-CrossAccountAccess 托管式策略由 CloudWatch-CrossAccountSharingRole IAM 角色使用。此角色和策略使跨账户控制面板的用户能够查看共享仪表板的各个账户中的自动控制面板。

CloudWatchAutomaticDashboardsAccess 策略的内容如下所示：

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "autoscaling:DescribeAutoScalingGroups",
 "cloudfront:GetDistribution",
 "cloudfront:ListDistributions",
 "dynamodb:DescribeTable",
 "dynamodb:ListTables",
 "ec2:DescribeInstances",
 "ec2:DescribeVolumes",
 "ecs:DescribeClusters",
 "ecs:DescribeContainerInstances",
 "ecs:ListClusters",
 "ecs:ListContainerInstances",
 "ecs:ListServices",
 "elasticache:DescribeCacheClusters",
 "elasticbeanstalk:DescribeEnvironments",
 "elasticfilesystem:DescribeFileSystems",
 "elasticloadbalancing:DescribeLoadBalancers",
 "kinesis:DescribeStream",
 "kinesis:ListStreams",
 "lambda:GetFunction",
 "lambda:ListFunctions",
 "rds:DescribeDBClusters",
 "rds:DescribeDBInstances",
 "resource-groups:ListGroupResources",
 "resource-groups:ListGroups",
 "route53:GetHealthCheck",
 "route53:ListHealthChecks",
 "s3:ListAllMyBuckets",
 "s3:ListBucket",
 "sns:ListTopics",
 "sqs:GetQueueAttributes",
 "sqs:GetQueueUrl",
 "sqs:ListQueues",
 "synthetics:DescribeCanariesLastRun",
 "tag:GetResources"
],
 "Effect": "Allow",
 "Resource": "*"
 },
 {
 "Action": [
 "apigateway:GET"
],
 "Effect": "Allow",
 "Resource": [
 "arn:aws:apigateway:*::/restapis*"
]
 }
]
}
```

## CloudWatchAgentServerPolicy

CloudWatchAgentServerPolicy 策略可用于附加到 Amazon EC2 实例的 IAM 角色中，以允许 CloudWatch 代理从实例读取信息并将其写入 CloudWatch。

```
{
 "Version": "2012-10-17",
```

```
"Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:PutMetricData",
 "ec2:DescribeVolumes",
 "ec2:DescribeTags",
 "logs:PutLogEvents",
 "logs:DescribeLogStreams",
 "logs:DescribeLogGroups",
 "logs:CreateLogStream",
 "logs:CreateLogGroup"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetParameter"
],
 "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-*"
 }
]
```

## CloudWatchAgentAdminPolicy

CloudWatchAgentAdminPolicy 策略可用于附加到 Amazon EC2 实例的 IAM 角色。此策略允许 CloudWatch 代理从实例读取信息并将其写入 CloudWatch，还可以将信息写入 Parameter Store。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:PutMetricData",
 "ec2:DescribeTags",
 "logs:PutLogEvents",
 "logs:DescribeLogStreams",
 "logs:DescribeLogGroups",
 "logs:CreateLogStream",
 "logs:CreateLogGroup"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetParameter",
 "ssm:PutParameter"
],
 "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-*"
 }
]
}
```

### Note

您可以通过登录到 IAM 控制台并在该控制台中搜索特定策略来查看这些权限策略。

此外，您还可以创建您自己的自定义 IAM 策略，以授予对 CloudWatch 操作和资源的相关权限。您可以将这些自定义策略附加到需要这些权限的 IAM 用户或组。

## 用于 CloudWatch Synthetics 的 Amazon 托管式 ( 预定义 ) 策略

CloudWatchSyntheticsFullAccess 和 CloudWatchSyntheticsReadOnlyAccess Amazon 托管式策略可供您分配给将要管理或使用 CloudWatch Synthetics 的用户。以下其他策略也是相关的：

- AmazonS3ReadOnlyAccess 和 CloudWatchReadOnlyAccess – 在 CloudWatch 控制台中读取所有 Synthetics 数据所必需的策略。
- AWSLambdaReadOnlyAccess – 可查看 Canary 使用的源代码。
- CloudWatchSyntheticsFullAccess 使您能够创建 Canary。此外，要创建和删除将为其创建新 IAM 角色的 Canary，您还需要以下内联策略语句：

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iam:CreateRole",
 "iam:DeleteRole",
 "iam:CreatePolicy",
 "iam:DeletePolicy",
 "iam:AttachRolePolicy",
 "iam:DetachRolePolicy",
],
 "Resource": [
 "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*",
 "arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*"
]
 }
]
}
```

### Important

#### 授予用户

iam:CreateRole、iam:DeleteRole、iam:CreatePolicy、iam:DeletePolicy、iam:AttachRolePolicy 和 iam:DetachRolePolicy 权限，将授予用户完全管理访问权限，用户可以创建、附加和删除具有匹配 `arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*` 和 `arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*` 的 ARN 的角色和策略。例如，拥有这些权限的用户可以创建一个对所有资源具有完全权限的策略，并将该策略附加到匹配该 ARN 模式的任何角色。请谨慎地为相关人员授予这些权限。

有关附加策略和向用户授予权限的信息，请参阅[更改 IAM 用户的权限](#)和[为用户或角色嵌入内联策略](#)。

## CloudWatchSyntheticsFullAccess

CloudWatchSyntheticsFullAccess 策略的内容如下。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "synthetics:*"
],
 "Resource": "*"
 }
]
}
```

```
 "Effect": "Allow",
 "Action": [
 "s3:CreateBucket",
 "s3:PutEncryptionConfiguration"
],
 "Resource": [
 "arn:aws:s3:::cw-syn-results-*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "iam:ListRoles",
 "s3:ListAllMyBuckets",
 "xray:GetTraceSummaries",
 "xray:BatchGetTraces",
 "apigateway:GET"
],
 "Resource": "*"
},
{
 "Effect": "Allow",
 "Action": [
 "s3:GetBucketLocation"
],
 "Resource": "arn:aws:s3:::*"
},
{
 "Effect": "Allow",
 "Action": [
 "s3:GetObject",
 "s3:ListBucket"
],
 "Resource": "arn:aws:s3:::cw-syn-*"
},
{
 "Effect": "Allow",
 "Action": [
 "s3:GetObjectVersion"
],
 "Resource": "arn:aws:s3:::aws-synthetics-library-*"
},
{
 "Effect": "Allow",
 "Action": [
 "iam:PassRole"
],
 "Resource": [
 "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
],
 "Condition": {
 "StringEquals": {
 "iam:PassedToService": [
 "lambda.amazonaws.com",
 "synthetics.amazonaws.com"
]
 }
 }
},
{
 "Effect": "Allow",
 "Action": [
 "iam:GetRole",
 "iam:ListAttachedRolePolicies"
],
 "Resource": [
```

```
 "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
],
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:GetMetricData",
 "cloudwatch:GetMetricStatistics"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:PutMetricAlarm",
 "cloudwatch:DeleteAlarms"
],
 "Resource": [
 "arn:aws:cloudwatch::*:alarm:Synthetics-*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:DescribeAlarms"
],
 "Resource": [
 "arn:aws:cloudwatch::*:alarm:*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "lambda:CreateFunction",
 "lambda:AddPermission",
 "lambda:PublishVersion",
 "lambda:UpdateFunctionCode",
 "lambda:UpdateFunctionConfiguration",
 "lambda:GetFunctionConfiguration",
 "lambda>DeleteFunction"
],
 "Resource": [
 "arn:aws:lambda::*:function:cwsyn-*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "lambda:GetLayerVersion",
 "lambda:PublishLayerVersion",
 "lambda>DeleteLayerVersion"
],
 "Resource": [
 "arn:aws:lambda::*:layer:cwsyn-*",
 "arn:aws:lambda::*:layer:Synthetics:*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "ec2:DescribeVpcs",
 "ec2:DescribeSubnets",
 "ec2:DescribeSecurityGroups"
],
 "Resource": [
 "*"
]
 }
}
```

```
],
 {
 "Effect": "Allow",
 "Action": [
 "sns:ListTopics"
],
 "Resource": [
 "*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "sns:CreateTopic",
 "sns:Subscribe",
 "sns:ListSubscriptionsByTopic"
],
 "Resource": [
 "arn:*:sns:*:*:Synthetics-*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "kms:ListAliases"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "kms:DescribeKey"
],
 "Resource": "arn:aws:kms:*:*:key/*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "kms:Decrypt"
],
 "Resource": "arn:aws:kms:*:*:key/*",
 "Condition": {
 "StringLike": {
 "kms:ViaService": [
 "s3.*.amazonaws.com"
]
 }
 }
 }
]
}
```

### CloudWatchSyntheticsReadOnlyAccess

CloudWatchSyntheticsReadOnlyAccess 策略的内容如下。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "synthetics:Describe*",

```



```
 "synthetics:Get*",
 "synthetics:List*",
 "lambda:GetFunctionConfiguration"
],
 "Resource": "*"
 }
]
```

## 适用于 Amazon CloudWatch RUM 的 Amazon 托管式 ( 预定义 ) 策略

您可以将 Amazon 托管式策略 AmazonCloudWatchRUMFullAccess 和 AmazonCloudWatchRUMReadOnlyAccess 分配给将管理或使用 CloudWatch RUM 的用户。

### AmazonCloudWatchRUMFullAccess

AmazonCloudWatchRUMFullAccess 策略的内容如下所示。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "rum:*"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "iam:GetRole",
 "iam:CreateServiceLinkedRole"
],
 "Resource": [
 "arn:aws:iam::*:role/aws-service-role/rum.amazonaws.com/AWSServiceRoleForRealUserMonitoring"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "iam:PassRole"
],
 "Resource": [
 "arn:aws:iam::*:role/RUM-Monitor*"
],
 "Condition": {
 "StringEquals": {
 "iam:PassedToService": [
 "cognito-identity.amazonaws.com"
]
 }
 }
 },
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:GetMetricData",
 "cloudwatch:GetMetricStatistics",
 "cloudwatch:ListMetrics"
],
 "Resource": "*"
 }
]
}
```

```
 },
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:DescribeAlarms"
],
 "Resource": "arn:aws:cloudwatch:*:*:alarm:*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "cognito-identity:CreateIdentityPool",
 "cognito-identity:ListIdentityPools",
 "cognito-identity:DescribeIdentityPool",
 "cognito-identity:GetIdentityPoolRoles",
 "cognito-identity:SetIdentityPoolRoles"
],
 "Resource": "arn:aws:cognito-identity:*:*:identitypool/*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "logs:CreateLogGroup",
 "logs:DeleteLogGroup",
 "logs:PutRetentionPolicy",
 "logs:CreateLogStream"
],
 "Resource": "arn:aws:logs:*:*:log-group:*RUMService*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "logs:CreateLogDelivery",
 "logs:GetLogDelivery",
 "logs:UpdateLogDelivery",
 "logs>DeleteLogDelivery",
 "logs:ListLogDeliveries",
 "logs:DescribeResourcePolicies"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "logs:DescribeLogGroups"
],
 "Resource": "arn:aws:logs:*:*:log-group::log-stream:*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "synthetics:describeCanaries",
 "synthetics:describeCanariesLastRun"
],
 "Resource": "arn:aws:synthetics:*:*:canary:*"
 }
]
}
```

### [AmazonCloudWatchRUMReadOnlyAccess](#)

AmazonCloudWatchRUMReadOnlyAccess 策略的内容如下所示。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "rum:GetAppMonitor",
 "rum:GetAppMonitorData",
 "rum:ListAppMonitors"
],
 "Resource": "*"
 }
]
```

### AmazonCloudWatchRUMServiceRolePolicy

您无法将 AmazonCloudWatchRUMServiceRolePolicy 附加到 IAM 实体。此策略会附加到允许 CloudWatch RUM 向其他相关 Amazon 服务发布监控数据的服务相关角色。有关此服务相关角色的更多信息，请参阅 [对 CloudWatch RUM 使用服务相关角色 \(p. 868\)](#)。

## 适用于 CloudWatch Evidently 的 Amazon 托管式 ( 预定义 ) 策略

您可以将 Amazon 托管式策略 CloudWatchSyntheticsFullAccess 和 CloudWatchSyntheticsReadOnlyAccess 分配给将管理或使用 CloudWatch Evidently 的用户。

### CloudWatchEvidentlyFullAccess

CloudWatchEvidentlyFullAccess 策略的内容如下所示。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "evidently:*"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "iam:ListRoles"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "iam:GetRole"
],
 "Resource": [
 "arn:aws:iam::*:role/service-role/CloudWatchRUMEvidentlyRole-*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "s3:GetBucketLocation",
 "s3:ListAllMyBuckets"
],
 "Resource": "arn:aws:s3:::*"
 }
]
}
```

```
 },
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:GetMetricData",
 "cloudwatch:GetMetricStatistics",
 "cloudwatch:DescribeAlarmHistory",
 "cloudwatch:DescribeAlarmsForMetric",
 "cloudwatch:ListTagsForResource"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:DescribeAlarms",
 "cloudwatch:TagResource",
 "cloudwatch:UntagResource"
],
 "Resource": [
 "arn:aws:cloudwatch:*:*:alarm:*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "cloudtrail:LookupEvents"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:PutMetricAlarm"
],
 "Resource": [
 "arn:aws:cloudwatch:*:*:alarm:Evidently-Alarm-*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "sns:ListTopics"
],
 "Resource": [
 "*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "sns:CreateTopic",
 "sns:Subscribe",
 "sns:ListSubscriptionsByTopic"
],
 "Resource": [
 "arn:*:sns:*:*:Evidently-*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "logs:DescribeLogGroups"
],
 "Resource": [
 "*"
]
 }
]
}
```

```
]
 }
]
}
```

### CloudWatchEvidentlyReadOnlyAccess

CloudWatchEvidentlyReadOnlyAccess 策略的内容如下所示。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "evidently:GetExperiment",
 "evidently:GetFeature",
 "evidently:GetLaunch",
 "evidently:GetProject",
 "evidently:GetSegment",
 "evidently:ListExperiments",
 "evidently:ListFeatures",
 "evidently:ListLaunches",
 "evidently:ListProjects",
 "evidently:ListSegments",
 "evidently:ListSegmentReferences"
],
 "Resource": "*"
 }
]
}
```

### 适用于 Amazon Systems Manager Incident Manager 的 Amazon 托管式策略

AWSCloudWatchAlarms\_ActionSSMIncidentsServiceRolePolicy 策略附加到一个与服务相关的角色，该角色允许 CloudWatch 在 Amazon Systems Manager Incident Manager 中代表您启动事件。有关更多信息，请参阅[CloudWatch 告警 Systems Manager Incident Manager 操作的服务相关角色权限 \(p. 864\)](#)。

该策略具有以下权限：

- ssm-incidents:StartIncident

## 客户托管式策略示例

本节的用户策略示例介绍如何授予对各 CloudWatch 操作的权限。当您使用 CloudWatch API、Amazon SDK 或 Amazon CLI 时，可以使用这些策略。

示例

- [示例 1：允许用户对 CloudWatch 进行完全访问 \(p. 857\)](#)
- [示例 2：允许对 CloudWatch 进行只读访问 \(p. 858\)](#)
- [示例 3：停止或终止 Amazon EC2 实例 \(p. 858\)](#)

### 示例 1：允许用户对 CloudWatch 进行完全访问

要授予用户对 CloudWatch 的完全访问权限，您可以使用授予用户 CloudWatchFullAccess 托管式策略，而不必创建客户托管式策略。CloudWatchFullAccess 策略的内容列在 [CloudWatchFullAccess \(p. 845\)](#) 中。

## 示例 2：允许对 CloudWatch 进行只读访问

以下策略允许用户对 CloudWatch 进行只读访问以及查看 Amazon EC2 Auto Scaling 操作、CloudWatch 指标、CloudWatch Logs 数据以及告警相关 Amazon SNS 数据。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "autoscaling:Describe*",
 "cloudwatch:Describe*",
 "cloudwatch:Get*",
 "cloudwatch:List*",
 "logs:Get*",
 "logs:Describe*",
 "sns:Get*",
 "sns:List*"
],
 "Effect": "Allow",
 "Resource": "*"
 }
]
}
```

## 示例 3：停止或终止 Amazon EC2 实例

以下策略允许 CloudWatch 告警操作停止或终止 EC2 实例。在以下示例中，GetMetricData、ListMetrics 和 DescribeAlarms 操作是可选的。建议您选择这些操作以确保正确停止或终止了实例。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "cloudwatch:PutMetricAlarm",
 "cloudwatch:GetMetricData",
 "cloudwatch:ListMetrics",
 "cloudwatch:DescribeAlarms"
],
 "Resource": [
 "*"
],
 "Effect": "Allow"
 },
 {
 "Action": [
 "ec2:DescribeInstanceStatus",
 "ec2:DescribeInstances",
 "ec2:StopInstances",
 "ec2:TerminateInstances"
],
 "Resource": [
 "*"
],
 "Effect": "Allow"
 }
]
}
```

## CloudWatch 对 Amazon 托管式策略的更新

查看有关 CloudWatch 的 Amazon 托管式策略更新的详细信息 ( 从该服务开始跟踪这些更改开始 )。有关此页面更改的自动提示，请订阅 CloudWatch 文档历史记录页面上的 RSS 源。

| 更改                                                                       | 说明                                                                                                                                                                                                                                                                               | 日期               |
|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">CloudWatchEvidentlyReadOnlyAccess (p. 348)</a><br>– 对现有策略的更新 | CloudWatch Evidently 添加了对 CloudWatchEvidentlyReadOnlyAccess 的权限。<br><br>已添加 <code>evidently:GetSegment</code> 、 <code>evidently:ListSegments</code> 和 <code>evidently:ListSegmentReferences</code> 权限，以便使用此策略的用户可以看到已创建的 Evidently 受众分段。                                         | 2022 年 8 月 12 日  |
| <a href="#">CloudWatchSyntheticsFullAccess (p. 348)</a><br>– 对现有策略的更新    | CloudWatch Synthetics 添加了对 CloudWatchSyntheticsFullAccess 的权限。<br><br>添加了 <code>lambda:DeleteFunction</code> 和 <code>lambda:DeleteLayerVersion</code> 权限，以便 CloudWatch Synthetics 可在 Canary 时删除相关资源。添加了 <code>iam:ListAttachedRolePolicies</code> ，以便客户可以查看附加到 Canary IAM 角色的策略。 | 2022 年 5 月 6 日   |
| <a href="#">AmazonCloudWatchRUMFullAccess (p. 348)</a><br>– 新策略          | CloudWatch 添加了一项启用对 CloudWatch RUM 的全面管理的新策略。<br><br>CloudWatch RUM 允许您对 Web 应用程序执行真实的用户监控。有关更多信息，请参阅 <a href="#">使用 CloudWatch Rum (p. 323)</a> 。                                                                                                                               | 2021 年 11 月 29 日 |
| <a href="#">AmazonCloudWatchRUMReadOnlyAccess (p. 348)</a><br>– 新策略      | CloudWatch 添加了一项启用对 CloudWatch RUM 的只读访问的新策略。<br><br>CloudWatch RUM 允许您对 Web 应用程序执行真实的用户监控。有关更多信息，请参阅 <a href="#">使用 CloudWatch Rum (p. 323)</a> 。                                                                                                                               | 2021 年 11 月 29 日 |
| <a href="#">CloudWatchEvidentlyFullAccess (p. 348)</a><br>– 新策略          | CloudWatch 添加了一项启用对 CloudWatch Evidently 的全面管理的新策略。<br><br>CloudWatch Evidently 允许您对 Web 应用程序执行 A/B 实验，并逐步执行这些实验。有关更多信息，请参阅 <a href="#">使用 CloudWatch</a>                                                                                                                        | 2021 年 11 月 29 日 |

| 更改                                                                                 | 说明                                                                                                                                                                                                                                                                                                                                                 | 日期               |
|------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
|                                                                                    | <a href="#">Evidently 执行启动和 A/B 实验 (p. 292)</a> 。                                                                                                                                                                                                                                                                                                  |                  |
| <a href="#">CloudWatchEvidentlyReadOnlyAccess (p. 57)</a><br>– 新策略                 | CloudWatch 添加了一项启用对 CloudWatch Evidently 的只读访问的新策略。<br><br>CloudWatch Evidently 允许您对 Web 应用程序执行 A/B 实验，并逐步执行这些实验。有关更多信息，请参阅 <a href="#">使用 CloudWatch Evidently 执行启动和 A/B 实验 (p. 292)</a> 。                                                                                                                                                        | 2021 年 11 月 29 日 |
| <a href="#">AWSServiceRoleForCloudWatchRUM (p. 57)</a><br>– 新的托管式策略                | CloudWatch 添加了一项新的服务相关角色的策略，以允许 CloudWatch RUM 将监控数据发布给其他相关 Amazon 服务。                                                                                                                                                                                                                                                                             | 2021 年 11 月 29 日 |
| <a href="#">CloudWatchSyntheticsFullAccess (p. 49)</a><br>– 对现有策略的更新               | CloudWatch Synthetics 向 CloudWatchSyntheticsFullAccess 添加了权限，还更改了一个权限的范围。<br><br>添加了 kms:ListAliases 权限，以便用户可以列出可用于加密 canary 构件的可用 Amazon KMS 密钥。添加了 kms:DescribeKey 权限，以便用户可以查看将用于加密 canary 构件的密钥的详细信息。此外，还添加了 kms:Decrypt 权限，以便用户能够解密 canary 构件。此解密功能仅限用于 Amazon S3 存储桶中的资源。<br><br>s3:GetBucketLocation 权限的 Resource 范围从 * 更改为了 arn:aws:s3:::*。 | 2021 年 9 月 29 日  |
| <a href="#">CloudWatchSyntheticsFullAccess (p. 49)</a><br>– 对现有策略的更新               | CloudWatch Synthetics 添加了一个对 CloudWatchSyntheticsFullAccess 策略的权限。<br><br>lambda:UpdateFunctionCode 权限，以便使用此策略的用户可以更改 Canary 的运行时版本。                                                                                                                                                                                                               | 2021 年 7 月 20 日  |
| <a href="#">AWSCloudWatchAlarms_ActionSSMIncidentResponse (p. 857)</a><br>– 新托管式策略 | CloudWatch 添加了一个新的托管式 IAM policy，以允许 CloudWatch 在 Amazon Systems Manager Incident Manager 中创建事件。                                                                                                                                                                                                                                                   | 2021 年 5 月 10 日  |



| 更改                                                             | 说明                                                                                                                                                                  | 日期              |
|----------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| <a href="#">CloudWatchAutomaticDashboardsAccess</a> – 对现有策略的更新 | CloudWatch 添加了一个对 CloudWatchAutomaticDashboardsAccess 托管式策略的权限。synthetics:DescribeCanariesLastRun 权限添加到此策略中，以使跨账户控制面板用户能够查看有关 CloudWatch Synthetics canary 运行的详细信息。 | 2021 年 4 月 20 日 |
| CloudWatch 开始跟踪更改                                              | CloudWatch 开始跟踪其 Amazon 托管式策略的更改。                                                                                                                                   | 2021 年 4 月 14 日 |

## 使用条件键限制对 CloudWatch 命名空间的访问

使用 IAM 条件键限制用户仅在指定的 CloudWatch 命名空间中发布指标。

仅允许在一个命名空间中发布

以下策略将用户限制为仅在名为 MyCustomNamespace 的命名空间中发布指标。

```
{
 "Version": "2012-10-17",
 "Statement": {
 "Effect": "Allow",
 "Resource": "*",
 "Action": "cloudwatch:PutMetricData",
 "Condition": {
 "StringEquals": {
 "cloudwatch:namespace": "MyCustomNamespace"
 }
 }
 }
}
```

排除从命名空间发布

以下策略允许用户在除 CustomNamespace2 之外的任何命名空间中发布指标。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Resource": "*",
 "Action": "cloudwatch:PutMetricData"
 },
 {
 "Effect": "Deny",
 "Resource": "*",
 "Action": "cloudwatch:PutMetricData",
 "Condition": {
 "StringEquals": {
 "cloudwatch:namespace": "CustomNamespace2"
 }
 }
 }
]
}
```

```
}
```

## 使用条件键限制 Contributor Insights 用户对日志组的访问

要在 Contributor Insights 中创建规则并查看其结果，用户必须具有 `cloudwatch:PutInsightRule` 权限。默认情况下，具有此权限的用户可以创建 Contributor Insights 规则，用于评估 CloudWatch Logs 中的任何日志组，然后查看结果。结果可以包含这些日志组的贡献者数据。

您可以使用条件密钥创建 IAM 策略，以授予用户为某些日志组编写 Contributor Insights 规则的权限，同时阻止用户为其他日志组编写规则和查看此数据。

有关 IAM 策略中 `Condition` 元素的更多信息，请参阅 [IAM JSON 策略元素：条件](#)。

仅允许对特定日志组的编写规则和查看结果的访问权限

以下策略允许用户访问对名为 `AllowedLogGroup` 的日志组和名称以 `AllowedWildcard` 开头的所有日志组的编写规则和查看结果的访问权限。它不授予对其他日志组的编写规则或查看规则结果的访问权限。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowCertainLogGroups",
 "Effect": "Allow",
 "Action": "cloudwatch:PutInsightRule",
 "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*",
 "Condition": {
 "ForAllValues:StringEqualsIgnoreCase": {
 "cloudwatch:requestInsightRuleLogGroups": [
 "AllowedLogGroup",
 "AllowedWildcard*"
]
 }
 }
 }
]
}
```

拒绝为特定日志组编写规则，但允许为所有其他日志组编写规则

以下策略显式拒绝用户对名为 `ExplicitlyDeniedLogGroup` 的日志组编写规则和查看结果的访问权限，但允许对所有其他日志组编写规则和查看规则结果。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowInsightRulesOnLogGroupsByDefault",
 "Effect": "Allow",
 "Action": "cloudwatch:PutInsightRule",
 "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*"
 },
 {
 "Sid": "ExplicitDenySomeLogGroups",
 "Effect": "Deny",
 "Action": "cloudwatch:PutInsightRule",

```

```
 "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*",
 "Condition": {
 "ForAllValues:StringEqualsIgnoreCase": {
 "cloudwatch:requestInsightRuleLogGroups": [
 "/test/alpine/ExplicitlyDeniedLogGroup"
]
 }
 }
 }
}
```

## 使用条件键限制告警操作

当 CloudWatch 告警更改状态时，它们可以执行不同的操作，例如停止和终止 EC2 实例以及执行 Systems Manager 操作。当告警变为任何状态（包括 ALARM（告警）、OK（正常）或 INSUFFICIENT\_DATA（数据不足））时，可以启动这些操作。

使用 `cloudwatch:AlarmActions` 条件键，以允许用户创建告警，这些告警只能在告警状态发生变化时执行您指定的操作。例如，您可以允许用户创建只能执行非 EC2 操作的告警。

允许用户创建只能发送 Amazon SNS 通知或执行 Systems Manager 操作的告警

以下策略限制用户只能创建发送 Amazon SNS 通知或执行 Systems Manager 操作的告警。用户不能创建执行 EC2 操作的告警。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "CreateAlarmsThatCanPerformOnlySNSandSSMActions",
 "Effect": "Allow",
 "Action": "cloudwatch:PutMetricAlarm",
 "Resource": "*",
 "Condition": {
 "ForAllValues:StringLike": {
 "cloudwatch:AlarmActions": [
 "arn:aws:sns:*",
 "arn:aws:ssm:*"
]
 }
 }
 }
]
}
```

## 为 CloudWatch 使用服务相关角色

Amazon CloudWatch 使用 Amazon Identity and Access Management (IAM) [服务相关角色](#)。服务相关角色是一种独特类型的 IAM 角色，它与 CloudWatch 直接相关。服务相关角色由 CloudWatch 预定义，并包含该服务代表您调用其他 Amazon 服务所需的一切权限。

CloudWatch 中的一个服务相关角色将设置可终止、停止或重启 Amazon EC2 实例的 CloudWatch 告警，而无需您手动添加必要的权限。另一个服务相关角色可让监控账户访问您指定的其他账户的 CloudWatch 数据，从而构建跨账户跨区域的控制面板。

CloudWatch 定义这些服务相关角色的权限，除非另行定义，否则仅 CloudWatch 能够代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

只有在首先删除角色的相关资源后，才能删除角色。此限制将保护您的 CloudWatch 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其它服务的信息，请参阅[使用 IAM 的 Amazon 服务](#)并查找 Service-Linked Role ( 服务相关角色 ) 列中显示为 Yes ( 是 ) 的服务。请选择 Yes 与查看该服务的服务相关角色文档的链接。

## CloudWatch EC2 操作的服务相关角色权限

CloudWatch 使用名为 AWSServiceRoleForCloudWatchEvents 的服务相关角色 – CloudWatch 使用此服务相关角色执行 Amazon EC2 告警操作。

AWSServiceRoleForCloudWatchEvents 服务相关角色信任 CloudWatch Events 服务来代入该角色。CloudWatch Events 在被告警调起时，调用终止、停止或重启实例操作。

AWSServiceRoleForCloudWatchEvents 服务相关角色权限策略允许 CloudWatch Events 对 Amazon EC2 实例完成以下操作：

- `ec2:StopInstances`
- `ec2:TerminateInstances`
- `ec2:RecoverInstances`
- `ec2:DescribeInstanceRecoveryAttribute`
- `ec2:DescribeInstances`
- `ec2:DescribeInstanceState`

AWSServiceRoleForCloudWatchCrossAccount 服务相关角色权限策略允许 CloudWatch 完成以下操作：

- `sts:AssumeRole`

## CloudWatch 告警 Systems Manager OpsCenter 操作的服务相关角色权限

CloudWatch 使用名为 AWSServiceRoleForCloudWatchAlarms\_ActionSSM 的服务相关角色 – CloudWatch 在告警进入 ALARM ( 告警 ) 状态时使用此服务相关角色执行 Systems Manager OpsCenter 操作。

AWSServiceRoleForCloudWatchAlarms\_ActionSSM 服务相关角色信任 CloudWatch 服务来代入该角色。CloudWatch 告警在被告警调用时，将调用 Systems Manager OpsCenter 操作。

AWSServiceRoleForCloudWatchAlarms\_ActionSSM 服务相关角色权限策略允许 Systems Manager 完成以下操作：

- `ssm:CreateOpsItem`

## CloudWatch 告警 Systems Manager Incident Manager 操作的服务相关角色权限

CloudWatch 使用名为 AWSServiceRoleForCloudWatchAlarms\_ActionSSMIncidents 的服务相关角色 – CloudWatch 在告警进入 ALARM ( 告警 ) 状态时使用此服务相关角色启动 Incident Manager 事件。

AWSServiceRoleForCloudWatchAlarms\_ActionSSMIncidents 服务相关角色信任 CloudWatch 服务来代入该角色。CloudWatch 告警在被告警调用时，将调用 Systems Manager Incident Manager 操作。

AWSServiceRoleForCloudWatchAlarms\_ActionSSMIncidents 服务相关角色权限策略允许 Systems Manager 完成以下操作：

- `ssm-incidents:StartIncident`

## CloudWatch 跨账户跨区域的服务相关角色权限

CloudWatch 使用名为 AWSServiceRoleForCloudWatchCrossAccount 的服务相关角色 – CloudWatch 使用此角色访问您指定的其他 Amazon 账户中的 CloudWatch 数据。SLR 仅提供代入角色权限以允许 CloudWatch 服务代入共享账户中的角色。它是提供对数据的访问权限的共享角色。

AWSServiceRoleForCloudWatchCrossAccount 服务相关角色权限策略允许 CloudWatch 完成以下操作：

- `sts:AssumeRole`

AWSServiceRoleForCloudWatchCrossAccount 服务相关角色信任 CloudWatch 服务来代入该角色。

## 为 CloudWatch 创建服务相关角色

您无需手动创建这两个服务相关角色。当您首次在 Amazon Web Services Management Console、IAM CLI 或 IAM API 中创建告警时，CloudWatch 会为您创建 AWSServiceRoleForCloudWatchEvents 和 AWSServiceRoleForCloudWatchAlarms\_ActionSSM 服务。当您首次启用一个账户作为跨账户跨区域功能的监控账户时，CloudWatch 会为您创建 AWSServiceRoleForCloudWatchCrossAccount。

有关更多信息，请参阅《IAM 用户指南》中的[创建服务相关角色](#)。

## 为 CloudWatch 编辑服务相关角色

CloudWatch 不允许您编辑

AWSServiceRoleForCloudWatchEvents、AWSServiceRoleForCloudWatchAlarms\_ActionSSM 或 AWSServiceRoleForCloudWatchCrossAccount 角色。在创建这些角色后，您无法更改这些角色的名称，因为可能有不同的实体引用它们。但是，您可以使用 IAM 编辑这些角色的描述。

### 编辑服务相关角色描述 (IAM 控制台)

您可以使用 IAM 控制台编辑服务相关角色的描述。

编辑服务相关角色的描述 (控制台)

1. 在 IAM 控制台的导航窗格中，选择 Roles。
2. 以下代码示例显示如何将 IAM policy 附加到用户。
3. 在 Role description 的最右侧，选择 Edit。
4. 在框中键入新描述，然后选择 Save (保存)。

### 编辑服务相关角色描述 (Amazon CLI)

您可以从 Amazon Command Line Interface 使用 IAM 命令编辑服务相关角色的描述。

更改服务相关角色描述 (Amazon CLI)

1. (可选) 要查看角色的当前描述，请使用以下命令：

```
$ aws iam get-role --role-name role-name
```

可以在 Amazon CLI 命令中使用角色名称 (而不是 ARN) 引用角色。例如, 如果某个角色的 ARN 为 `arn:aws:iam::123456789012:role/myrole`, 则将该角色称为 **myrole**。

2. 要更新服务相关角色的描述, 请使用以下命令:

```
$ aws iam update-role-description --role-name role-name --description description
```

## 编辑服务相关角色描述 (IAM API)

您可以使用 IAM API 编辑服务相关角色的描述。

### 更改服务相关角色的描述 (API)

1. (可选) 要查看角色的当前描述, 请使用以下命令:

`GetRole`

2. 要更新角色的描述, 请使用以下命令:

`UpdateRoleDescription`

## 为 CloudWatch 删除服务相关角色

如果您不再具有自动停止、终止或重启 EC2 实例的警报, 我们建议您删除 `AWSServiceRoleForCloudWatchEvents` 角色。

如果您不再具有执行 Systems Manager OpsCenter 操作的告警, 我们建议您删除 `AWSServiceRoleForCloudWatchAlarms_ActionSSM` 角色。

这样您就没有未被主动监控或维护的未使用实体。但是, 您必须先清除您的服务相关角色, 然后才能将其删除。

### 清除服务相关角色

必须先确认服务相关角色没有活动会话并删除该角色使用的任何资源, 然后才能使用 IAM 删除服务相关角色。

在 IAM 控制台中检查服务相关角色是否具有活动会话

1. 通过以下网址打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中, 选择 Roles (角色)。选择 `AWSServiceRoleForCloudWatchEvents` 角色的名称 (不是复选框)。
3. 在选定角色的摘要页上, 选择访问顾问并查看服务相关角色的近期活动。

#### Note

如果您不确定 CloudWatch 是否正在使用 `AWSServiceRoleForCloudWatchEvents` 角色, 请尝试删除该角色。如果服务正在使用该角色, 则删除操作会失败, 并且您可以查看正在使用该角色的区域。如果该角色已被使用, 则您必须等待会话结束, 然后才能删除该角色。您无法撤销服务相关角色对会话的权限。

## 删除服务相关角色 (IAM 控制台)

您可以使用 IAM 控制台删除服务相关角色。

### 删除服务相关角色 ( 控制台 )

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Roles ( 角色 )。选中要删除的角色名称旁的复选框，而不是名称或行本身。
3. 对于角色操作，请选择删除角色。
4. 在确认对话框中，查看上次访问服务数据，该数据显示每个选定角色上次访问Amazon服务的时间。这样可帮助您确认角色当前是否处于活动状态。要继续，请选择 Yes, Delete ( 是的，删除 )。
5. 监视 IAM 控制台通知，以监控服务相关角色的删除进度。由于 IAM 服务相关角色的删除是异步操作，因此，在提交要删除的角色后，删除任务可能会成功，也可能会失败。如果该任务失败，请从通知中选择查看详细信息或查看资源以了解删除失败的原因。如果因角色正在使用服务中的资源而导致删除操作失败，则失败原因将包含一个资源列表。

### 删除服务相关角色 (Amazon CLI)

您可以从 Amazon Command Line Interface 使用 IAM 命令删除服务相关角色。

#### 删除服务相关角色 (Amazon CLI)

1. 如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 `deletion-task-id` 以检查删除任务的状态。键入以下命令以提交服务相关角色的删除请求：

```
$ aws iam delete-service-linked-role --role-name service-linked-role-name
```

2. 键入以下命令以检查删除任务的状态：

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

删除任务的状态可能是 NOT\_STARTED、IN\_PROGRESS、SUCCEEDED 或 FAILED。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。

### 删除服务相关角色 ( IAM API )

您可以使用 IAM API 删除服务相关角色。

#### 删除服务相关角色 (API)

1. 要提交服务相关角色的删除请求，请调用 `DeleteServiceLinkedRole`。在请求中，指定您要删除的角色名称。

如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 `DeletionTaskId` 以检查删除任务的状态。

2. 要检查删除的状态，请调用 `GetServiceLinkedRoleDeletionStatus`。在请求中，指定 `DeletionTaskId`。

删除任务的状态可能是 NOT\_STARTED、IN\_PROGRESS、SUCCEEDED 或 FAILED。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。

## 对 Amazon 服务相关角色的 CloudWatch 更新

查看有关 CloudWatch 的 Amazon 托管策略更新的详细信息 ( 从该服务开始跟踪这些更改开始 )。有关此页面更改的自动提示，请订阅 CloudWatch 文档历史记录页面上的 RSS 源。



| 更改                                                          | 说明                                                                                     | 日期              |
|-------------------------------------------------------------|----------------------------------------------------------------------------------------|-----------------|
| <a href="#">AWSServiceRoleForCloudWatchAlarm</a> – 新的服务相关角色 | CloudWatch 添加了新的服务相关角色，以允许 CloudWatch 在 Amazon Systems Manager Incident Manager 中创建事件。 | 2021 年 4 月 26 日 |
| CloudWatch 开始跟踪更改                                           | CloudWatch 为其服务相关角色开启了跟踪的更改。                                                           | 2021 年 4 月 26 日 |

## 对 CloudWatch RUM 使用服务相关角色

CloudWatch RUM 使用 Amazon Identity and Access Management (IAM) [服务相关角色](#)。服务相关角色是一种独特类型的 IAM 角色，它与 RUM 直接相关。服务相关角色由 RUM 预定义，并包含服务代表您调用其他 Amazon 服务所需的所有权限。

RUM 定义这些服务相关角色的权限，除非另有定义，否则仅 RUM 能够代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

只有在首先删除角色的相关资源后，才能删除角色。此限制可保护您的 RUM 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅[使用 IAM 的 Amazon 服务](#)并查找 Service-Linked Role (服务相关角色) 列中显示为 Yes (是) 的服务。请选择 Yes 与查看该服务的服务相关角色文档的链接。

## RUM 的服务相关角色权限

RUM 对您为其启用了 X-Ray 跟踪的应用程序监控使用名为 `AWSServiceRoleForCloudWatchRUM` 的服务相关角色 – 此角色允许 RUM 将 Amazon X-Ray 跟踪数据发送到您的账户。

`AWSServiceRoleForCloudWatchRUM` 服务相关角色信任 X-Ray 服务来代入该角色。X-Ray 将跟踪数据发送到您的账户。

`AWSServiceRoleForCloudWatchRUM` 服务相关角色附加有名为 `AmazonCloudWatchRUMServiceRolePolicy` 的 IAM policy。此策略向 CloudWatch RUM 授予将监控数据发布到其他相关 Amazon 服务的权限。其中包括允许 RUM 完成以下操作的权限：

- `xray:PutTraceSegments`

## 为 RUM 创建服务相关角色

您无需手动为 CloudWatch RUM 创建服务相关角色。当您首次创建启用了 X-Ray 跟踪的应用程序监控，或更新应用程序监控以使用 X-Ray 跟踪时，RUM 会为您创建 `AWSServiceRoleForCloudWatchRUM`。

有关更多信息，请参阅《IAM 用户指南》中的[创建服务相关角色](#)。

## 编辑 RUM 的服务相关角色

CloudWatch RUM 不允许您编辑 `AWSServiceRoleForCloudWatchRUM` 角色。在创建这些角色后，您无法更改这些角色的名称，因为可能有不同的实体引用它们。但是，您可以使用 IAM 编辑这些角色的描述。

### 编辑服务相关角色描述 (IAM 控制台)

您可以使用 IAM 控制台编辑服务相关角色的描述。



### 编辑服务相关角色的描述 (控制台)

1. 在 IAM 控制台的导航窗格中，选择 Roles。
2. 以下代码示例显示如何将 IAM policy 附加到用户。
3. 在 Role description 的最右侧，选择 Edit。
4. 在框中键入新描述，然后选择 Save (保存)。

### 编辑服务相关角色描述 (Amazon CLI)

您可以从 Amazon Command Line Interface 使用 IAM 命令编辑服务相关角色的描述。

#### 更改服务相关角色描述 (Amazon CLI)

1. (可选) 要查看角色的当前描述，请使用以下命令：

```
$ aws iam get-role --role-name role-name
```

可以在 Amazon CLI 命令中使用角色名称 (而不是 ARN) 引用角色。例如，如果某个角色的 ARN 为 `arn:aws:iam::123456789012:role/myrole`，则将该角色称为 **myrole**。

2. 要更新服务相关角色的描述，请使用以下命令：

```
$ aws iam update-role-description --role-name role-name --description description
```

### 编辑服务相关角色描述 (IAM API)

您可以使用 IAM API 编辑服务相关角色的描述。

#### 更改服务相关角色的描述 (API)

1. (可选) 要查看角色的当前描述，请使用以下命令：

[GetRole](#)

2. 要更新角色的描述，请使用以下命令：

[UpdateRoleDescription](#)

## 删除 RUM 的服务相关角色

如果您不再拥有启用了 X-Ray 的应用程序监控，我们建议您删除 `AWSServiceRoleForCloudWatchRUM` 角色。

这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除您的服务相关角色，然后才能将其删除。

### 清除服务相关角色

必须先确认服务相关角色没有活动会话并删除该角色使用的任何资源，然后才能使用 IAM 删除服务相关角色。

在 IAM 控制台中检查服务相关角色是否具有活动会话

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。

2. 在导航窗格中，选择 Roles ( 角色 )。选择 AWSServiceRoleForCloudWatchRUM 角色的名称 ( 而不是复选框 )。
3. 在选定角色的摘要页上，选择访问顾问并查看服务相关角色的近期活动。

#### Note

如果您不确定 RUM 是否正在使用 AWSServiceRoleForCloudWatchRUM 角色，请尝试删除该角色。如果服务正在使用该角色，则删除操作会失败，并且您可以查看正在使用该角色的区域。如果该角色已被使用，则您必须等待会话结束，然后才能删除该角色。您无法撤销服务相关角色对会话的权限。

## 删除服务相关角色 ( IAM 控制台 )

您可以使用 IAM 控制台删除服务相关角色。

### 删除服务相关角色 ( 控制台 )

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Roles ( 角色 )。选中要删除的角色名称旁的复选框，而不是名称或行本身。
3. 对于角色操作，请选择删除角色。
4. 在确认对话框中，查看上次访问服务数据，该数据显示每个选定角色上次访问 Amazon 服务的时间。这样可帮助您确认角色当前是否处于活动状态。要继续，请选择 Yes, Delete ( 是的，删除 )。
5. 监视 IAM 控制台通知，以监控服务相关角色的删除进度。由于 IAM 服务相关角色的删除是异步操作，因此，在提交要删除的角色后，删除任务可能会成功，也可能会失败。如果该任务失败，请从通知中选择查看详细信息或查看资源以了解删除失败的原因。如果因角色正在使用服务中的资源而导致删除操作失败，则失败原因将包含一个资源列表。

## 删除服务相关角色 (Amazon CLI)

您可以从 Amazon Command Line Interface 使用 IAM 命令删除服务相关角色。

### 删除服务相关角色 (Amazon CLI)

1. 如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 deletion-task-id 以检查删除任务的状态。键入以下命令以提交服务相关角色的删除请求：

```
$ aws iam delete-service-linked-role --role-name service-linked-role-name
```

2. 键入以下命令以检查删除任务的状态：

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

删除任务的状态可能是 NOT\_STARTED、IN\_PROGRESS、SUCCEEDED 或 FAILED。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。

## 删除服务相关角色 ( IAM API )

您可以使用 IAM API 删除服务相关角色。

### 删除服务相关角色 (API)

1. 要提交服务相关角色的删除请求，请调用 [DeleteServiceLinkedRole](#)。在请求中，指定您要删除的角色名称。

如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 `DeletionTaskId` 以检查删除任务的状态。

2. 要检查删除的状态，请调用 [GetServiceLinkedRoleDeletionStatus](#)。在请求中，指定 `DeletionTaskId`。

删除任务的状态可能是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。

## 在 CloudWatch Application Insights 中使用服务相关角色

CloudWatch Application Insights 使用 Amazon Identity and Access Management (IAM) [服务相关角色](#)。服务相关角色是一种独特类型的 IAM 角色，它与 CloudWatch Application Insights 直接相关。服务相关角色由 CloudWatch Application Insights 预定义，并包含该服务代表您调用其他 Amazon 服务所需的一切权限。

服务相关角色更方便设置 CloudWatch Application Insights，因为无需手动添加必要权限。CloudWatch Application Insights 定义其服务相关角色的权限，除非另外定义，否则只有 CloudWatch Application Insights 可以代入其角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

有关支持服务链接角色的其他服务的信息，请参阅[使用 IAM 的 Amazon 服务](#)，并在服务链接角色列中查找 Yes (是) 的服务。选择是链接，查看该服务的[服务相关角色文档](#)。

## CloudWatch Application Insights 的服务相关角色权限

CloudWatch Application Insights 使用名为 `AWSServiceRoleForApplicationInsights` 的服务相关角色。Application Insights 使用此角色执行操作，例如分析客户的资源组、创建 CloudFormation 堆栈以创建有关指标的告警，以及在 EC2 实例上配置 CloudWatch 代理。此服务相关角色附加到以下托管策略上：`CloudwatchApplicationInsightsServiceLinkedRolePolicy`。有关此策略的更新，请参阅[对 Amazon 托管策略的 Application Insights 更新 \(p. 880\)](#)。

角色权限策略允许 CloudWatch Application Insights 对资源完成以下操作。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cloudwatch:DescribeAlarmHistory",
 "cloudwatch:DescribeAlarms",
 "cloudwatch:GetMetricData",
 "cloudwatch:ListMetrics",
 "cloudwatch:PutMetricAlarm",
 "cloudwatch:DeleteAlarms",
 "cloudwatch:PutAnomalyDetector",
 "cloudwatch:DeleteAnomalyDetector",
 "cloudwatch:DescribeAnomalyDetectors"
],
 "Resource": [
 "*"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "logs:FilterLogEvents",
 "logs:GetLogEvents",

```

```
 "logs:DescribeLogStreams",
 "logs:DescribeLogGroups"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "events:DescribeRule"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "cloudFormation:CreateStack",
 "cloudFormation:UpdateStack",
 "cloudFormation>DeleteStack",
 "cloudFormation:DescribeStackResources"
],
 "Resource": [
 "arn:aws:cloudformation:*:*:stack/ApplicationInsights-*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "cloudFormation:DescribeStacks",
 "cloudFormation:ListStackResources"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "tag:GetResources"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "resource-groups:ListGroupResources",
 "resource-groups:GetGroupQuery",
 "resource-groups:GetGroup"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "resource-groups:CreateGroup",
 "resource-groups>DeleteGroup"
],
 "Resource": [
 "arn:aws:resource-groups:*:*:group/ApplicationInsights-*"
]
}
```

```
]
},
{
 "Effect": "Allow",
 "Action": [
 "elasticloadbalancing:DescribeLoadBalancers",
 "elasticloadbalancing:DescribeTargetGroups",
 "elasticloadbalancing:DescribeTargetHealth"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "autoscaling:DescribeAutoScalingGroups"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "ssm:PutParameter",
 "ssm:DeleteParameter",
 "ssm:AddTagsToResource",
 "ssm:RemoveTagsFromResource",
 "ssm:GetParameters"
],
 "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-ApplicationInsights-*"
},
{
 "Effect": "Allow",
 "Action": [
 "ssm:CreateAssociation",
 "ssm:UpdateAssociation",
 "ssm:DeleteAssociation",
 "ssm:DescribeAssociation"
],
 "Resource": [
 "arn:aws:ec2:*:*:instance/*",
 "arn:aws:ssm:*:*:association/*",
 "arn:aws:ssm:*:*:managed-instance/*",
 "arn:aws:ssm:*:*:document/AWSEC2-ApplicationInsightsCloudwatchAgentInstallAndConfigure",
 "arn:aws:ssm:*:*:document/AWS-ConfigureAWSPackage",
 "arn:aws:ssm:*:*:document/AmazonCloudWatch-ManageAgent"
]
},
{
 "Effect": "Allow",
 "Action": [
 "ssm:GetOpsItem",
 "ssm:CreateOpsItem",
 "ssm:DescribeOpsItems",
 "ssm:UpdateOpsItem",
 "ssm:DescribeInstanceInformation"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
```

```
"Action": [
 "ssm:AddTagsToResource"
],
"Resource": "arn:aws:ssm:*:*:opsitem/*"
},
{
 "Effect": "Allow",
 "Action": [
 "ssm:ListCommandInvocations"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": "ssm:SendCommand",
 "Resource": [
 "arn:aws:ec2:*:*:instance/*",
 "arn:aws:ssm:*:*:document/AWSEC2-CheckPerformanceCounterSets",
 "arn:aws:ssm:*:*:document/AWS-ConfigureAWSPackage",
 "arn:aws:ssm:*:*:document/AWSEC2-DetectWorkload",
 "arn:aws:ssm:*:*:document/AmazonCloudWatch-ManageAgent"
]
},
{
 "Effect": "Allow",
 "Action": [
 "ec2:DescribeInstances",
 "ec2:DescribeVolumes",
 "ec2:DescribeVolumeStatus"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "rds:DescribeDBInstances",
 "rds:DescribeDBClusters"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "lambda:ListFunctions",
 "lambda:GetFunctionConfiguration",
 "lambda:ListEventSourceMappings"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "events:PutRule",
 "events:PutTargets",
 "events:RemoveTargets",
 "events>DeleteRule"
],
 "Resource": [
```

```
 "arn:aws:events:*:*:rule/AmazonCloudWatch-ApplicationInsights-*"
],
},
{
 "Effect": "Allow",
 "Action": [
 "xray:GetServiceGraph",
 "xray:GetTraceSummaries",
 "xray:GetTimeSeriesServiceStatistics",
 "xray:GetTraceGraph"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "dynamodb:ListTables",
 "dynamodb:DescribeTable",
 "dynamodb:DescribeContributorInsights",
 "dynamodb:DescribeTimeToLive"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "application-autoscaling:DescribeScalableTargets"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "s3:ListAllMyBuckets",
 "s3:GetMetricsConfiguration",
 "s3:GetReplicationConfiguration"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "states:ListStateMachines",
 "states:DescribeExecution",
 "states:DescribeStateMachine",
 "states:GetExecutionHistory"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "apigateway:GET"
],
 "Resource": [
 "*"
]
}
```

```
]
},
{
 "Effect": "Allow",
 "Action": [
 "ecs:DescribeClusters",
 "ecs:DescribeContainerInstances",
 "ecs:DescribeServices",
 "ecs:DescribeTaskDefinition",
 "ecs:DescribeTasks",
 "ecs:DescribeTaskSets",
 "ecs:ListClusters",
 "ecs:ListContainerInstances",
 "ecs:ListServices",
 "ecs:ListTasks"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "ecs:UpdateClusterSettings"
],
 "Resource": [
 "arn:aws:ecs:*:*:cluster/*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "eks:DescribeCluster",
 "eks:DescribeFargateProfile",
 "eks:DescribeNodegroup",
 "eks:ListClusters",
 "eks:ListFargateProfiles",
 "eks:ListNodegroups",
 "fsx:DescribeFileSystems"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "sns:GetSubscriptionAttributes",
 "sns:GetTopicAttributes",
 "sns:GetSMSAttributes",
 "sns:ListSubscriptionsByTopic",
 "sns:ListTopics"
],
 "Resource": [
 "*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "sqs:ListQueues"
],
 "Resource": "*"
}
{
 "Effect": "Allow",
```



```
{
 "Action": [
 "logs:DeleteSubscriptionFilter"
],
 "Resource": [
 "arn:aws:logs:*:*:log-group:*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "logs:PutSubscriptionFilter"
],
 "Resource": [
 "arn:aws:logs:*:*:log-group:*",
 "arn:aws:logs:*:*:destination:AmazonCloudWatch-ApplicationInsights-LogIngestionDestination*"
]
}
]
```

必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅 IAM 用户指南中的[服务相关角色权限](#)。

## 为 CloudWatch Application Insights 创建服务相关角色

无需手动创建服务相关角色。当您在 Amazon Web Services Management Console 中创建新的 Application Insights 应用程序时，CloudWatch Application Insights 将为您创建服务相关角色。

如果删除该服务相关角色，然后希望再次创建该角色，您可以使用相同的过程在您的账户中重新创建该角色。当您创建新的 Application Insights 应用程序时，CloudWatch Application Insights 将为您再次创建服务相关角色。

## 为 CloudWatch Application Insights 编辑服务相关角色

CloudWatch Application Insights 不允许您编辑 AWSServiceRoleForApplicationInsights 服务相关角色。创建服务相关角色后，将无法更改角色名称，因为可能有多个实体引用该角色。但是可以使用 IAM 编辑角色说明。有关更多信息，请参见《IAM 用户指南》中的[编辑服务相关角色](#)。

## 为 CloudWatch Application Insights 删除服务相关角色

如果您不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样，您就可以避免使用当前未监控或维护的未使用实体。不过，您必须先删除 Application Insights 中的所有应用程序，然后才能手动删除该角色。

### Note

在尝试删除资源时，如果 CloudWatch Application Insights 服务正在使用该角色，删除可能会失败。如果发生这种情况，请等待几分钟后重试。

若要删除 AWSServiceRoleForApplicationInsights 使用的 CloudWatch Application Insights 资源

- 删除所有 CloudWatch Application Insights 应用程序。有关更多信息，请参阅《CloudWatch Application Insights 用户指南》中的“删除应用程序”。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台、Amazon CLI 或 Amazon API 删除 AWSServiceRoleForApplicationInsights 服务相关角色。有关更多信息，请参见 IAM 用户指南中的[删除服务相关角色](#)。

## CloudWatch Application Insights 服务相关角色支持的区域

CloudWatch Application Insights 支持在服务可用的所有 Amazon 区域中使用服务相关角色。有关更多信息，请参阅 [CloudWatch Application Insights 区域和端点](#)。

## Amazon 适用于 Amazon CloudWatch Application Insights 的 托管式策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 Amazon 托管式策略更简单。创建仅为团队提供所需权限的 [IAM 客户托管式策略](#) 需要时间和专业知识。要快速入门，您可以使用我们的 Amazon 托管式策略。这些策略涵盖常见使用案例，可在您的 Amazon Web Services 账户 中使用。有关 Amazon 托管式策略的更多信息，请参阅 IAM 用户指南中的 [Amazon 托管式策略](#)。

Amazon Web Services 负责维护和更新 Amazon 托管式策略。您无法更改 Amazon 托管式策略中的权限。服务偶尔会向 Amazon 托管式策略添加额外权限以支持新功能。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新功能或新操作可用时，服务最有可能更新 Amazon 托管式策略。服务不会从 Amazon 托管式策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，Amazon 还支持跨多种服务的工作职能的托管式策略。例如，`ViewOnlyAccess` Amazon 托管式策略提供对许多 Amazon Web Services 和资源的只读访问权限。当服务启动新功能时，Amazon 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅 IAM 用户指南中的 [适用于工作职能的 Amazon 托管策略](#)。

### 托管式策略

- [Amazon 托管式策略：CloudWatchApplicationInsightsFullAccess](#) (p. 878)
- [Amazon 托管式策略：CloudWatchApplicationInsightsReadOnlyAccess](#) (p. 880)
- [Amazon 托管式策略：CloudwatchApplicationInsightsServiceLinkedRolePolicy](#) (p. 880)
- [对 Amazon 托管式策略的 Application Insights 更新](#) (p. 880)

### 托管式策略

- [Amazon 托管式策略：CloudWatchApplicationInsightsFullAccess](#) (p. 878)
- [Amazon 托管式策略：CloudWatchApplicationInsightsReadOnlyAccess](#) (p. 880)
- [Amazon 托管式策略：CloudwatchApplicationInsightsServiceLinkedRolePolicy](#) (p. 880)
- [对 Amazon 托管式策略的 Application Insights 更新](#) (p. 880)

## Amazon 托管式策略：CloudWatchApplicationInsightsFullAccess

您可以将 `CloudWatchApplicationInsightsFullAccess` 策略附加得到 IAM 身份。

此策略授予管理权限，允许完全访问 Application Insights 功能。

## 权限详细信息

此策略包含以下权限。

- `applicationinsights` – 允许完全访问 Application Insights 功能。
- `iam` – 允许 Application Insights 创建服务相关角色，`AWSServiceRoleForApplicationInsights`。这是必需的，以使 Application Insights 可用执行操作，例如分析客户的资源组、创建 CloudFormation 堆栈以创建有关指标的告警，以及在 EC2 实例上配置 CloudWatch 代理。有关更多信息，请参阅 [在 CloudWatch Application Insights 中使用服务相关角色 \(p. 871\)](#)。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "applicationinsights:*",
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "ec2:DescribeInstances",
 "ec2:DescribeVolumes",
 "rds:DescribeDBInstances",
 "rds:DescribeDBClusters",
 "sqs:ListQueues",
 "elasticloadbalancing:DescribeLoadBalancers",
 "elasticloadbalancing:DescribeTargetGroups",
 "elasticloadbalancing:DescribeTargetHealth",
 "autoscaling:DescribeAutoScalingGroups",
 "lambda:ListFunctions",
 "dynamodb:ListTables",
 "s3:ListAllMyBuckets",
 "sns:ListTopics",
 "states:ListStateMachines",
 "apigateway:GET",
 "ecs:ListClusters",
 "ecs:DescribeTaskDefinition",
 "ecs:ListServices",
 "ecs:ListTasks",
 "eks:ListClusters",
 "eks:ListNodegroups",
 "fsx:DescribeFileSystems",
 "logs:DescribeLogGroups"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": [
 "iam:CreateServiceLinkedRole"
],
 "Resource": [
 "arn:aws:iam:*:role/aws-service-role/application-insights.amazonaws.com/AWSServiceRoleForApplicationInsights"
],
 "Condition": {
 "StringEquals": {
 "iam:AWSServiceName": "application-insights.amazonaws.com"
 }
 }
 }
]
}
```

```
}
 }
]
}
```

## Amazon 托管式策略： CloudWatchApplicationInsightsReadOnlyAccess

您可以将 CloudWatchApplicationInsightsReadOnlyAccess 策略附加得到 IAM 身份。

此策略授予管理权限，允许只读访问所有 Application Insights 功能。

权限详细信息

此策略包含以下权限。

- applicationinsights – 允许只读访问 Application Insights 功能。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "applicationinsights:Describe*",
 "applicationinsights:List*"
],
 "Resource": "*"
 }
]
}
```

## Amazon 托管式策略： CloudwatchApplicationInsightsServiceLinkedRolePolicy

您无法将 CloudwatchApplicationInsightsServiceLinkedRolePolicy 策略附加到 IAM 实体。此策略附加到一个与服务相关的角色，该角色允许 Application Insights 监控客户资源。有关更多信息，请参阅 [在 CloudWatch Application Insights 中使用服务相关角色 \(p. 871\)](#)。

## 对 Amazon 托管式策略的 Application Insights 更新

查看有关对 Amazon 托管式策略的 Application Insights 更新的详细信息（从该服务开始跟踪这些更改开始）。有关此页面更改的自动提示，请订阅 Application Insights [Document history \(文档历史记录\)](#) (p. 909) 页面上的 RSS 源。

| 更改                                                       | 描述                                                                                                                                                                                                                                                                          | 日期              |
|----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| CloudWatchApplicationInsightsFullAccess<br>– 现有策略的更新     | Application Insights 添加了描述日志组的新权限。<br><br>Amazon CloudWatch Application Insights 需要此权限，以确保在创建新应用程序时，账户中具有监控日志组的正确权限。                                                                                                                                                        | 2022 年 1 月 24 日 |
| CloudwatchApplicationInsightsServiceAccess<br>– 对现有策略的更新 | Application Insights 添加创建和删除 CloudWatch Log 订阅筛选器的新权限。<br><br>Amazon CloudWatch Application Insights 需要这些权限才能创建订阅筛选器，以便于对已配置应用程序中的资源进行日志监控。                                                                                                                                 | 2022 年 1 月 24 日 |
| CloudWatchApplicationInsightsFullAccess<br>– 现有策略的更新     | Application Insights 添加了新的权限来描述 Elastic Load Balancer 的目标组和目标健康状况。<br><br>Amazon CloudWatch Application Insights 需要这些权限，才能通过查询账户中的所有受支持资源来创建基于账户的应用程序。                                                                                                                      | 2021 年 11 月 4 日 |
| CloudwatchApplicationInsightsServiceAccess<br>– 对现有策略的更新 | Application Insights 添加在 Amazon EC2 实例上运行 AmazonCloudWatch-ManagedAgent SSM 文档的新权限。<br><br>Amazon CloudWatch Application Insights 需要此权限才能清除由 Application Insights 创建的 CloudWatch 代理配置文件。                                                                                    | 2021 年 9 月 30 日 |
| CloudwatchApplicationInsightsServiceAccess<br>– 对现有策略的更新 | Application Insights 添加新的权限，以支持基于账户的应用程序监控载入并监控账户中所有受支持的资源。<br><br>Amazon CloudWatch Application Insights 需要这些权限才能查询、标记资源并为这些资源创建组。<br><br>Application Insights 添加了新的权限以支持对 SNS 主题的监控。<br><br>Amazon CloudWatch Application Insights 需要这些权限，才能收集 SNS 资源中的元数据以配置对 SNS 主题的监控。 | 2021 年 9 月 15 日 |
| CloudWatchApplicationInsightsFullAccess<br>– 现有策略的更新     | Application Insights 添加了描述和列出受支持资源的新权限。                                                                                                                                                                                                                                     | 2021 年 9 月 15 日 |

| 更改                                                                 | 描述                                                                                                                                                                                                                                   | 日期              |
|--------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
|                                                                    | Amazon CloudWatch Application Insights 需要这些权限，才能通过查询账户中的所有受支持资源来创建基于账户的应用程序。                                                                                                                                                         |                 |
| CloudwatchApplicationInsightsServiceApplicationInsights – 对现有策略的更新 | ApplicationInsights 添加了对 FSx 资源的新权限。<br><br>Amazon CloudWatch Application Insights 需要这些权限才能读取客户 FSx 资源配置，并帮助客户使用 CloudWatch 自动设置最佳实践 FSx 监控。                                                                                         | 2021 年 8 月 31 日 |
| CloudwatchApplicationInsightsServiceApplicationInsights – 对现有策略的更新 | ApplicationInsights 添加了对 ECS 和 EKS 服务资源的新权限。<br><br>Amazon CloudWatch Application Insights 需要此权限才能读取客户容器资源配置，并帮助客户使用 CloudWatch 自动设置最佳实践容器监控。                                                                                        | 2021 年 5 月 18 日 |
| CloudwatchApplicationInsightsServiceApplicationInsights – 对现有策略的更新 | ApplicationInsights 添加了新权限，允许 OpsCenter 使用对 opsitem 资源类型的资源的 ssm:AddTagsToResource 操作来为 OpsItems 贴标签。<br><br>OpsCenter 需要此权限。Amazon CloudWatch Application Insights 会创建 OpsItems，以便客户可以使用 <a href="#">Amazon SSM OpsCenter</a> 解决问题。 | 2021 年 4 月 13 日 |
| Athena 开启了跟踪更改                                                     | Athena 为其 Amazon 托管式策略开启了跟踪更改。                                                                                                                                                                                                       | 2021 年 4 月 13 日 |

## Amazon CloudWatch 权限参考

在设置 [访问控制](#) (p. 838) 和编写您可挂载到 IAM 身份的权限策略（基于身份的策略）时，可以使用下表作为参考。此表列出每个 CloudWatch API 操作以及您可授权执行该操作的相应操作。可在策略的 Action 字段中指定操作，在策略的 Resource 字段中指定通配符 (\*) 作为资源值。

您可以在 CloudWatch 策略中使用 Amazon 范围的条件键来表达条件。有关 Amazon 范围的键的完整列表，请参阅 IAM 用户指南中的 [Amazon 全局和 IAM 条件上下文键](#)。

### Note

要指定操作，请在 API 操作名称之前使用 cloudwatch: 前缀。例如：`cloudwatch:GetMetricData`、`cloudwatch:ListMetrics` 或 `cloudwatch:*`（适用于所有 CloudWatch 操作）。

### 主题

- [CloudWatch API 操作和必需的操作权限](#) (p. 883)

- [CloudWatch Contributor Insights API 操作和所需的操作权限 \(p. 885\)](#)
- [CloudWatch Events API 操作和所需的操作权限 \(p. 885\)](#)
- [CloudWatch Logs API 操作和所需的操作权限 \(p. 886\)](#)
- [Amazon EC2 API 操作和所需的操作权限 \(p. 889\)](#)
- [Amazon EC2 Auto Scaling API 操作和所需的操作权限 \(p. 890\)](#)

## CloudWatch API 操作和必需的操作权限

| CloudWatch API 操作                       | 所需权限 ( API 操作 )                                                                                                                                                                                             |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">删除警报</a>                    | <code>cloudwatch:DeleteAlarms</code><br>要求删除警报。                                                                                                                                                             |
| <a href="#">DeleteDashboards</a>        | <code>cloudwatch:DeleteDashboards</code><br>删除控制面板所必需。                                                                                                                                                      |
| <a href="#">DeleteMetricStream</a>      | <code>cloudwatch:DeleteMetricStream</code><br>删除指标流所需。                                                                                                                                                      |
| <a href="#">DescribeAlarmHistory</a>    | <code>cloudwatch:DescribeAlarmHistory</code><br>要求查看警报历史记录。要检索有关复合告警的信息， <code>cloudwatch:DescribeAlarmHistory</code> 权限必须具有 * 范围。如果您的 <code>cloudwatch:DescribeAlarmHistory</code> 权限的范围较窄，则无法返回有关复合告警的信息。 |
| <a href="#">DescribeAlarms</a>          | <code>cloudwatch:DescribeAlarms</code><br>检索有关告警的信息所需。<br><br>要检索有关复合告警的信息， <code>cloudwatch:DescribeAlarms</code> 权限必须具有 * 范围。如果您的 <code>cloudwatch:DescribeAlarms</code> 权限的范围较窄，则无法返回有关复合告警的信息。          |
| <a href="#">DescribeAlarmsForMetric</a> | <code>cloudwatch:DescribeAlarmsForMetric</code><br>要求查看指标的警报。                                                                                                                                               |
| <a href="#">DisableAlarmActions</a>     | <code>cloudwatch:DisableAlarmActions</code><br>要求禁用警报操作。                                                                                                                                                    |
| <a href="#">EnableAlarmActions</a>      | <code>cloudwatch:EnableAlarmActions</code><br>要求启用警报操作。                                                                                                                                                     |
| <a href="#">GetDashboard</a>            | <code>cloudwatch:GetDashboard</code><br>若要显示有关现有控制面板的数据，则是必需的。                                                                                                                                              |
| <a href="#">GetMetricData</a>           | <code>cloudwatch:GetMetricData</code>                                                                                                                                                                       |

| CloudWatch API 操作                    | 所需权限 ( API 操作 )                                                                                                                                                                                        |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                      | 在 CloudWatch 控制台中检索大量指标数据以及对<br>该数据执行指标数学运算所需。                                                                                                                                                         |
| <a href="#">GetMetricStatistics</a>  | <code>cloudwatch:GetMetricStatistics</code><br><br>在 CloudWatch 控制台的其他部分和控制面板小部件中查看图表所需。                                                                                                               |
| <a href="#">GetMetricStream</a>      | <code>cloudwatch:GetMetricStream</code><br><br>查看指标流信息所需。                                                                                                                                              |
| <a href="#">GetMetricWidgetImage</a> | <code>cloudwatch:GetMetricWidgetImage</code><br><br>将一个或多个 CloudWatch 指标的图表快照作为位图图像检索所需。                                                                                                               |
| <a href="#">ListDashboards</a>       | <code>cloudwatch:ListDashboards</code><br><br>查看您的账户中 CloudWatch 控制面板的列表所需。                                                                                                                            |
| <a href="#">ListMetrics</a>          | <code>cloudwatch:ListMetrics</code><br><br>在 CloudWatch 控制台和 CLI 中查看或搜索指标名称所需。要求在控制面板小部件上选择指标。                                                                                                         |
| <a href="#">ListMetricStreams</a>    | <code>cloudwatch:ListMetricStreams</code><br><br>查看或搜索账户中指标流列表所需。                                                                                                                                      |
| <a href="#">PutCompositeAlarm</a>    | <code>cloudwatch:PutCompositeAlarm</code><br><br>创建复合告警所需<br><br>要创建复合告警， <code>cloudwatch:PutCompositeAlarm</code> 权限必须具有 * 范围。如果您的 <code>cloudwatch:PutCompositeAlarm</code> 权限的范围较窄，则无法返回有关复合告警的信息。 |
| <a href="#">PutDashboard</a>         | <code>cloudwatch:PutDashboard</code><br><br>创建控制面板或更新现有控制面板所必需。                                                                                                                                        |
| <a href="#">PutMetricAlarm</a>       | <code>cloudwatch:PutMetricAlarm</code><br><br>要求创建或更新警报。                                                                                                                                               |
| <a href="#">PutMetricData</a>        | <code>cloudwatch:PutMetricData</code><br><br>若要创建指标，则是必需的。                                                                                                                                             |
| <a href="#">PutMetricStream</a>      | <code>cloudwatch:PutMetricStream</code><br><br>创建指标流所需                                                                                                                                                 |
| <a href="#">SetAlarmState</a>        | <code>cloudwatch:SetAlarmState</code><br><br>要求手动设置警报的状态。                                                                                                                                              |



| CloudWatch API 操作                  | 所需权限 ( API 操作 )                                                                                     |
|------------------------------------|-----------------------------------------------------------------------------------------------------|
| <a href="#">StartMetricStreams</a> | <code>cloudwatch:StartMetricStreams</code><br>开启指标流中的指标流程所需。                                        |
| <a href="#">StopMetricStreams</a>  | <code>cloudwatch:StopMetricStreams</code><br>临时停止指标流中的指标流程所需。                                       |
| <a href="#">TagResource</a>        | <code>cloudwatch:TagResource</code><br>在 CloudWatch 资源 ( 如告警和 Contributor Insights 规则 ) 上添加或更新标签所需。 |
| <a href="#">UntagResource</a>      | <code>cloudwatch:UntagResource</code><br>从 CloudWatch 资源中移除标签所需。                                    |

## CloudWatch Contributor Insights API 操作和所需的操作权限

### Important

当您向用户授予 `cloudwatch:PutInsightRule` 权限时，默认情况下，该用户可以创建一个规则来评估 CloudWatch Logs 中的任何日志组。您可以添加 IAM policy 条件，以限制用户的这些权限，使其包含和排除特定的日志组。有关更多信息，请参阅[使用条件键限制 Contributor Insights 用户对日志组的访问](#) (p. 862)。

| CloudWatch Contributor Insights API 操作 | 所需权限 ( API 操作 )                                                                                        |
|----------------------------------------|--------------------------------------------------------------------------------------------------------|
| <a href="#">DeleteInsightRules</a>     | <code>cloudwatch:DeleteInsightRules</code><br>删除 Contributor Insights 规则所需。                            |
| <a href="#">DescribeInsightRules</a>   | <code>cloudwatch:DescribeInsightRules</code><br>查看您账户中的 Contributor Insights 规则所需。                     |
| <a href="#">EnableInsightRules</a>     | <code>cloudwatch:EnableInsightRules</code><br>启用 Contributor Insights 规则所需。                            |
| <a href="#">GetInsightRuleReport</a>   | <code>cloudwatch:GetInsightRuleReport</code><br>检索 Contributor Insights 规则收集的时间序列数据和其他统计数据所需。          |
| <a href="#">PutInsightRule</a>         | <code>cloudwatch:PutInsightRule</code><br>创建 Contributor Insights 规则所需。请参阅此表开头的 Important ( 重要提示 ) 信息。 |

## CloudWatch Events API 操作和所需的操作权限

| CloudWatch Events API 操作   | 所需权限 ( API 操作 )                |
|----------------------------|--------------------------------|
| <a href="#">DeleteRule</a> | <code>events:DeleteRule</code> |

| CloudWatch Events API 操作              | 所需权限 ( API 操作 )                                              |
|---------------------------------------|--------------------------------------------------------------|
|                                       | 删除规则所必需的。                                                    |
| <a href="#">DescribeRule</a>          | <code>events:DescribeRule</code><br>列出有关规则的详细信息所必需的。         |
| <a href="#">DisableRule</a>           | <code>events:DisableRule</code><br>禁用规则所必需的。                 |
| <a href="#">EnableRule</a>            | <code>events:EnableRule</code><br>启用规则所必需的。                  |
| <a href="#">ListRuleNamesByTarget</a> | <code>events:ListRuleNamesByTarget</code><br>列出与目标关联的规则所必需的。 |
| <a href="#">ListRules</a>             | <code>events:ListRules</code><br>列出您账户中的所有规则所必需的。            |
| <a href="#">ListTargetsByRule</a>     | <code>events:ListTargetsByRule</code><br>列出与规则关联的所有目标所必需的。   |
| <a href="#">PutEvents</a>             | <code>events:PutEvents</code><br>添加可匹配到规则的自定义活动所必需的。         |
| <a href="#">PutRule</a>               | <code>events:PutRule</code><br>创建或更新规则所必需的。                  |
| <a href="#">PutTargets</a>            | <code>events:PutTargets</code><br>将目标添加到规则所必需的。              |
| <a href="#">RemoveTargets</a>         | <code>events:RemoveTargets</code><br>从规则中删除目标所必需的。           |
| <a href="#">TestEventPattern</a>      | <code>events:TestEventPattern</code><br>针对给定事件测试事件模式所必需的。    |

## CloudWatch Logs API 操作和所需的操作权限

| CloudWatch Logs API 操作           | 所需权限 ( API 操作 )                                                   |
|----------------------------------|-------------------------------------------------------------------|
| <a href="#">CancelExportTask</a> | <code>logs:CancelExportTask</code><br>需要取消待处理或正在运行的导出任务。          |
| <a href="#">CreateExportTask</a> | <code>logs:CreateExportTask</code><br>将数据从日志组导出到 Amazon S3 存储桶所需。 |

| CloudWatch Logs API 操作                   | 所需权限 ( API 操作 )                                                                    |
|------------------------------------------|------------------------------------------------------------------------------------|
| <a href="#">CreateLogGroup</a>           | <code>logs:CreateLogGroup</code><br>需要创建新的日志组。                                     |
| <a href="#">CreateLogStream</a>          | <code>logs:CreateLogStream</code><br>需要在日志组中创建新的日志流。                               |
| <a href="#">DeleteDestination</a>        | <code>logs:DeleteDestination</code><br>需要删除日志目标并对其禁用所有订阅筛选器。                       |
| <a href="#">DeleteLogGroup</a>           | <code>logs:DeleteLogGroup</code><br>需要删除日志组 和所有关联的存档日志事件。                          |
| <a href="#">DeleteLogStream</a>          | <code>logs:DeleteLogStream</code><br>需要删除日志流和所有关联的存档日志事件。                          |
| <a href="#">DeleteMetricFilter</a>       | <code>logs:DeleteMetricFilter</code><br>需要删除与日志组关联的指标筛选器。                          |
| <a href="#">DeleteQueryDefinition</a>    | <code>logs:DeleteQueryDefinition</code><br>删除 CloudWatch Logs Insights 中保存的查询定义所需。 |
| <a href="#">DeleteResourcePolicy</a>     | <code>logs:DeleteResourcePolicy</code><br>删除 CloudWatch Logs 资源策略所需。               |
| <a href="#">DeleteRetentionPolicy</a>    | <code>logs:DeleteRetentionPolicy</code><br>需要删除日志组的保留策略。                           |
| <a href="#">DeleteSubscriptionFilter</a> | <code>logs:DeleteSubscriptionFilter</code><br>需要删除与日志组关联的订阅筛选器。                    |
| <a href="#">DescribeDestinations</a>     | <code>logs:DescribeDestinations</code><br>需要查看与账户关联的所有目标。                          |
| <a href="#">DescribeExportTasks</a>      | <code>logs:DescribeExportTasks</code><br>需要查看与账户关联的所有导出任务。                         |
| <a href="#">DescribeLogGroups</a>        | <code>logs:DescribeLogGroups</code><br>需要查看与账户关联的所有日志组。                            |
| <a href="#">DescribeLogStreams</a>       | <code>logs:DescribeLogStreams</code><br>需要查看与日志组关联的所有日志流。                          |
| <a href="#">DescribeMetricFilters</a>    | <code>logs:DescribeMetricFilters</code><br>需要查看与日志组关联的所有指标。                        |

| CloudWatch Logs API 操作                      | 所需权限 ( API 操作 )                                                                         |
|---------------------------------------------|-----------------------------------------------------------------------------------------|
| <a href="#">DescribeQueryDefinitions</a>    | <code>logs:DescribeQueryDefinitions</code><br>查看 CloudWatch Logs Insights 中保存的查询定义列表所需。 |
| <a href="#">DescribeQueries</a>             | <code>logs:DescribeQueries</code><br>查看已计划、正在执行或最近已执行的 CloudWatch Logs Insights 查询列表所需。 |
| <a href="#">DescribeResourcePolicies</a>    | <code>logs:DescribeResourcePolicies</code><br>查看 CloudWatch Logs 资源策略列表所需。              |
| <a href="#">DescribeSubscriptionFilters</a> | <code>logs:DescribeSubscriptionFilters</code><br>需要查看与日志组关联的所有订阅筛选器。                    |
| <a href="#">FilterLogEvents</a>             | <code>logs:FilterLogEvents</code><br>需要按照日志组筛选器模式对日志事件进行排序。                             |
| <a href="#">GetLogEvents</a>                | <code>logs:GetLogEvents</code><br>需要从日志流中检索日志事件。                                        |
| <a href="#">GetLogGroupFields</a>           | <code>logs:GetLogGroupFields</code><br>需要检索日志组中日志事件内包含的字段列表。                            |
| <a href="#">GetLogRecord</a>                | <code>logs:GetLogRecord</code><br>需要从单个日志事件中检索详细信息。                                     |
| <a href="#">GetQueryResults</a>             | <code>logs:GetQueryResults</code><br>检索 CloudWatch Logs Insights 查询的结果所需。               |
| <a href="#">ListTagsLogGroup</a>            | <code>logs:ListTagsLogGroup</code><br>需要列出与日志组关联的标签。                                    |
| <a href="#">PutDestination</a>              | <code>logs:PutDestination</code><br>创建或更新目标日志流 ( 例如, Kinesis 流 ) 所需。                    |
| <a href="#">PutDestinationPolicy</a>        | <code>logs:PutDestinationPolicy</code><br>需要创建或更新与现有日志目标关联的访问策略。                        |
| <a href="#">PutLogEvents</a>                | <code>logs:PutLogEvents</code><br>需要将一批日志事件上传到日志流。                                      |
| <a href="#">PutMetricFilter</a>             | <code>logs:PutMetricFilter</code><br>需要创建或更新指标筛选器并将其与日志组关联。                             |

| CloudWatch Logs API 操作                | 所需权限 ( API 操作 )                                                             |
|---------------------------------------|-----------------------------------------------------------------------------|
| <a href="#">PutQueryDefinition</a>    | <code>logs:PutQueryDefinition</code><br>在 CloudWatch Logs Insights 中保存查询所需。 |
| <a href="#">PutResourcePolicy</a>     | <code>logs:PutResourcePolicy</code><br>创建 CloudWatch Logs 资源策略所需。           |
| <a href="#">PutRetentionPolicy</a>    | <code>logs:PutRetentionPolicy</code><br>需要设置日志组中的日志事件的保存 ( 保留 ) 天数。         |
| <a href="#">PutSubscriptionFilter</a> | <code>logs:PutSubscriptionFilter</code><br>需要创建或更新订阅筛选器并将其与日志组关联。           |
| <a href="#">StartQuery</a>            | <code>logs:StartQuery</code><br>启动 CloudWatch Logs Insights 查询所需。           |
| <a href="#">StopQuery</a>             | <code>logs:StopQuery</code><br>停止正在执行的 CloudWatch Logs Insights 查询所需。       |
| <a href="#">TagLogGroup</a>           | <code>logs:TagLogGroup</code><br>需要添加或更新日志组标签。                              |
| <a href="#">TestMetricFilter</a>      | <code>logs:TestMetricFilter</code><br>需要针对日志事件消息采样测试筛选器模式。                  |

## Amazon EC2 API 操作和所需的操作权限

| Amazon EC2 API 操作                      | 所需权限 ( API 操作 )                                                 |
|----------------------------------------|-----------------------------------------------------------------|
| <a href="#">DescribeInstanceStatus</a> | <code>ec2:DescribeInstanceStatus</code><br>查看 EC2 实例状态详细信息所必需的。 |
| <a href="#">DescribeInstances</a>      | <code>ec2:DescribeInstances</code><br>查看 EC2 实例详细信息所必需的。        |
| <a href="#">RebootInstances</a>        | <code>ec2:RebootInstances</code><br>重启 EC2 实例所必需的。              |
| <a href="#">StopInstances</a>          | <code>ec2:StopInstances</code><br>停止 EC2 实例所必需的。                |
| <a href="#">TerminateInstances</a>     | <code>ec2:TerminateInstances</code><br>终止 EC2 实例所必需的。           |

## Amazon EC2 Auto Scaling API 操作和所需的操作权限

| Amazon EC2 Auto Scaling API 操作 | 所需权限 ( API 操作 )                                           |
|--------------------------------|-----------------------------------------------------------|
| 扩缩                             | <code>autoscaling:Scaling</code><br>扩展 Auto Scaling 组所需。  |
| 触发器                            | <code>autoscaling:Trigger</code><br>触发 Auto Scaling 操作所需。 |

## Amazon CloudWatch 的合规性验证

作为多项 Amazon 合规性计划的一部分，第三方审计员将评估 Amazon CloudWatch 的安全性和合规性。其中包括 SOC、PCI、FedRAMP、HIPAA 及其它。

有关特定合规性计划范围内的 Amazon 服务列表，请参阅[合规性计划范围内的 Amazon 服务](#)。有关常规信息，请参阅[Amazon 合规性计划](#)。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅[下载 Amazon Artifact 中的报告](#)。

您使用 Amazon CloudWatch 的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用法律法规。Amazon 提供以下资源来帮助您满足合规性：

- [安全性与合规性 Quick Start 指南](#) - 这些部署指南讨论了架构注意事项，并提供了在 Amazon 上部署基于安全性和合规性的基准环境的步骤。
- [《设计符合 HIPAA 安全性和合规性要求的架构》白皮书](#) - 此白皮书介绍公司如何使用 Amazon 创建符合 HIPAA 标准的应用程序。
- [Amazon 合规性资源](#) - 此业务手册和指南集合可能适用于您的行业和位置。
- [《Amazon Config 开发人员指南》中的使用规则评估资源](#) - Amazon Config；评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- [Amazon Security Hub](#) - 此 Amazon 服务提供了 Amazon 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

## Amazon CloudWatch 中的恢复能力

Amazon 全球基础设施围绕 Amazon 区域和可用区构建。区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅[Amazon 全球基础设施](#)。

## Amazon CloudWatch 中的基础设施安全性

作为一项托管式服务，Amazon CloudWatch 由[亚马逊云科技：安全流程概览](#)白皮书中所述的 Amazon 全球网络安全程序提供保护。

您可以使用 Amazon 发布的 API 调用通过网络访问 Amazon CloudWatch。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码

套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 [Amazon Security Token Service](#) (Amazon STS) 生成临时安全凭证来对请求进行签名。

## 网络隔离

Virtual Private Cloud (VPC) 是亚马逊云科技云内您自己的逻辑隔离区域中的虚拟网络。子网是 VPC 中的 IP 地址范围。您可以在 VPC 的子网中部署各种 Amazon 资源。例如，可以在子网中部署 Amazon EC2 实例、EMR 集群和 DynamoDB 表。有关更多信息，请参阅《[Amazon VPC 用户指南](#)》。

要使 CloudWatch 能够在不访问公有 Internet 的情况下与 VPC 中的资源进行通信，请使用 Amazon PrivateLink。有关更多信息，请参阅 [将 CloudWatch 和 CloudWatch Synthetics 与接口 VPC 终端节点结合使用](#) (p. 891)。

私有子网是不具有到公共 Internet 的默认路由的子网。在私有子网中部署 Amazon 资源不会阻止 Amazon CloudWatch 从资源中收集内置指标。

如果您需要在私有子网发布来自 Amazon 资源的自定义指标，则可使用代理服务器执行此操作。代理服务器会将这些 HTTPS 请求转发到 CloudWatch 的公有 API 端点。

## 将 CloudWatch 和 CloudWatch Synthetics 与接口 VPC 终端节点结合使用

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 托管 Amazon 资源，则可以在您的 VPC、CloudWatch 和 CloudWatch Synthetics 之间建立私有连接。您可以使用此连接实现 CloudWatch 和 CloudWatch Synthetics 与您的 VPC 上的资源的通信而不用访问公共互联网。

Amazon VPC 是一项 Amazon 服务，可用来启动在虚拟网络中定义的 Amazon 资源。借助 VPC，您可以控制您的网络设置，如 IP 地址范围、子网、路由表和网络网关。要将您的 VPC 连接到 CloudWatch 或 CloudWatch Synthetics，您需要定义一个接口 VPC 终端节点以将您的 VPC 连接到 Amazon 服务。该端点提供了到 CloudWatch 或 CloudWatch Synthetics 的可靠、可扩展的连接，无需互联网网关、网络地址转换 (NAT) 实例或 VPN 连接。有关更多信息，请参阅 Amazon VPC 用户指南中的[什么是 Amazon VPC](#)。

接口 VPC 终端节点由 Amazon PrivateLink 提供支持，后者是一种 Amazon 技术，可将弹性网络接口与私有 IP 地址结合使用来支持 Amazon 服务之间的私有通信。有关更多信息，请参阅[新增 – 适用于 Amazon 服务的 Amazon PrivateLink](#) 博客文章。

以下步骤适用于 Amazon VPC 的用户。有关更多信息，请参阅 Amazon VPC 用户指南中的[开始使用](#)。

## CloudWatch VPC 终端节点

CloudWatch 当前在以下 Amazon 区域中支持 VPC 终端节点：

- 美国东部（俄亥俄）
- 美国东部（弗吉尼亚北部）
- 美国西部（加利福尼亚北部）
- 美国西部（俄勒冈州）
- 亚太地区（香港）
- 亚太地区（孟买）
- 亚太地区（首尔）

- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太区域（东京）
- 加拿大（中部）
- 欧洲（法兰克福）
- 欧洲（爱尔兰）
- 欧洲（伦敦）
- 欧洲（巴黎）
- 中东（阿联酋）
- 南美洲（圣保罗）
- Amazon GovCloud（美国东部）
- Amazon GovCloud（美国西部）

## 为 CloudWatch 创建 VPC 终端节点

要开始使用 CloudWatch 与 VPC，请为 CloudWatch 创建一个接口 VPC 终端节点。要选择的服务名称为 `com.amazonaws.region.monitoring`。有关更多信息，请参阅 Amazon VPC 用户指南中的[创建接口端点](#)。

您不需要更改 CloudWatch 的设置。CloudWatch 使用公有端点或私有接口 VPC 终端节点（二者中在使用中的那个）调用其他 Amazon 服务。例如，如果为 CloudWatch 创建接口 VPC 终端节点，并且您已拥有从位于 VPC 上的资源流向 CloudWatch 的指标，默认情况下，这些指标将开始流过接口 VPC 终端节点。

## 控制对 CloudWatch VPC 终端节点的访问

VPC 终端节点策略是一种 IAM 资源策略，您在创建或修改端点时可将它附加到端点。如果您在创建端点时未附加策略，Amazon VPC 会为您附加一个默认策略，该策略允许对服务的完全访问。端点策略不会覆盖或替换 IAM 用户策略或服务特定的策略。这是一个单独的策略，用于控制从终端节点中对指定服务进行的访问。

终端节点策略必须采用 JSON 格式编写。

有关更多信息，请参阅《Amazon VPC 用户指南》中的[使用 VPC 终端节点控制对服务的访问](#)。

下面是用于 CloudWatch 的端点策略示例。该策略允许通过 VPC 连接到 CloudWatch 的用户将指标数据发送到 CloudWatch，并禁止他们执行其他 CloudWatch 操作。

```
{
 "Statement": [
 {
 "Sid": "PutOnly",
 "Principal": "*",
 "Action": [
 "cloudwatch:PutMetricData"
],
 "Effect": "Allow",
 "Resource": "*"
 }
]
}
```

### 编辑 CloudWatch 的 VPC 终端节点策略

1. 通过以下网址打开 Amazon VPC 控制台：<https://console.aws.amazon.com/vpc/>。



2. 在导航窗格中，选择 Endpoints (端点)。
3. 如果还没有为 CloudWatch 创建端点，请选择 Create Endpoint (创建端点)。接下来，选择 `com.amazonaws.region.monitoring`，然后选择 Create Endpoint (创建终端节点)。
4. 选择 `com.amazonaws.region.monitoring` 终端节点，然后选择 Policy (策略) 选项卡。
5. 选择 Edit Policy (编辑策略)，然后进行更改。

## CloudWatch Synthetics VPC 终端节点

CloudWatch Synthetics 当前在以下 Amazon 区域中支持 VPC 终端节点：

- 美国东部 (俄亥俄)
- 美国东部 (弗吉尼亚北部)
- 美国西部 (加利福尼亚北部)
- 美国西部 (俄勒冈州)
- 亚太地区 (香港)
- 亚太地区 (孟买)
- 亚太地区 (首尔)
- 亚太地区 (新加坡)
- 亚太地区 (悉尼)
- 亚太区域 (东京)
- 加拿大 (中部)
- 欧洲 (法兰克福)
- 欧洲 (爱尔兰)
- 欧洲 (伦敦)
- 欧洲 (巴黎)
- 南美洲 (圣保罗)

## 为 CloudWatch Synthetics 创建 VPC 终端节点

要开始将 CloudWatch Synthetics 与您的 VPC 结合使用，请为 CloudWatch Synthetics 创建一个接口 VPC 终端节点。要选择的服务名称为 `com.amazonaws.region.synthetics`。有关更多信息，请参阅 Amazon VPC 用户指南中的 [创建接口端点](#)。

您不需要更改 CloudWatch Synthetics 的设置。CloudWatch Synthetics 使用公有终端节点或私有接口 VPC 终端节点 (二者中在使用的那个) 与其他 Amazon 服务通信。例如，如果您为 CloudWatch Synthetics 创建了接口 VPC 终端节点，并且您已有 Amazon S3 的接口端点，则默认情况下，CloudWatch Synthetics 通过接口 VPC 终端节点开始与 Amazon S3 通信。

## 控制对您的 CloudWatch Synthetics VPC 终端节点的访问

VPC 终端节点策略是一种 IAM 资源策略，您在创建或修改端点时可将它附加到端点。如果在创建端点时未附加策略，我们将为您附加默认策略以允许对服务进行完全访问。端点策略不会覆盖或替换 IAM 用户策略或服务特定的策略。这是一个单独的策略，用于控制从终端节点中对指定服务进行的访问。

端点策略会影响由 VPC 私有托管的 canary。在私有子网上运行的 Canary 不需要这些策略。

终端节点策略必须采用 JSON 格式编写。

有关更多信息，请参阅《Amazon VPC 用户指南》中的 [使用 VPC 终端节点控制对服务的访问](#)。

以下是 CloudWatch Synthetics 的端点策略示例。此策略允许通过 VPC 连接到 CloudWatch Synthetics 的用户查看有关 Canary 及其运行的信息，但不能创建、修改或删除 Canary。

```
{
 "Statement": [
 {
 "Action": [
 "synthetics:DescribeCanaries",
 "synthetics:GetCanaryRuns"
],
 "Effect": "Allow",
 "Resource": "*",
 "Principal": "*"
 }
]
}
```

编辑 CloudWatch Synthetics 的 VPC 终端节点策略

1. 通过以下网址打开 Amazon VPC 控制台：<https://console.aws.amazon.com/vpc/>。
2. 在导航窗格中，选择 Endpoints（端点）。
3. 如果还没有为 CloudWatch Synthetics 创建端点，请选择 Create Endpoint（创建端点）。选择 com.amazonaws.**region**.synthetics，然后选择 Create endpoint（创建终端节点）。
4. 选择 com.amazonaws.**region**.synthetics 终端节点，然后选择 Policy（策略）选项卡。
5. 选择 Edit Policy（编辑策略），然后进行更改。

## Synthetics 金丝雀的安全注意事项

以下部分说明了在 Synthetics 中创建和运行金丝雀时应考虑的安全问题。

### 使用安全连接

由于金丝雀代码和金丝雀测试运行的结果可能包含敏感信息，因此，请不要让金丝雀通过未加密的连接来连接到终端节点。始终使用加密连接，例如以 `https://` 开头的连接。

### 金丝雀命名注意事项

金丝雀的 Amazon Resource Name (ARN) 包含在用户代理标头中，作为从 CloudWatch Synthetics 包装器库中包含的 Puppeteer 驱动的 Chromium 浏览器发出的出站调用的一部分。这有助于标识 CloudWatch Synthetics 金丝雀流量，并再次将其与正在发出调用的金丝雀关联。

金丝雀 ARN 包含金丝雀名称。选择未显示专有信息的水晶名称。

此外，请确保仅将您的水晶指向由您控制的网站和终端节点。

### 水晶代码中的密钥和敏感信息

如果您使用 zip 文件将水晶代码直接传递到水晶中，则脚本的内容可以在 Amazon CloudTrail 日志中查看。

如果您在水晶脚本中有敏感信息或密钥（例如访问密钥或数据库证书），我们强烈建议您将脚本作为版本控制对象存储在 Amazon S3 中，并将 Amazon S3 位置传递到水晶中，而不是通过 zip 文件传递水晶代码。

如果您确实使用 zip 文件传递金丝雀脚本，我们强烈建议您不要将密钥或敏感信息包含在金丝雀源代码中。有关如何使用 Amazon Secrets Manager 来帮助保护密钥安全的更多信息，请参阅[什么是 Amazon Secrets Manager？](#)。

## 权限注意事项

我们建议您限制对由 CloudWatch Synthetics 创建或使用的资源的访问。对金丝雀存储测试运行结果和其他构件（例如，日志和屏幕截图）的 Amazon S3 存储桶使用严格权限。

同样，对金丝雀源代码的存储位置保留严格权限，以免用户意外或恶意删除用于金丝雀的 Lambda 层或 Lambda 函数。

要帮助确保您运行所需的金丝雀代码，您可以对用于存储金丝雀代码的 Amazon S3 存储桶使用对象版本控制。之后，在您指定此代码作为金丝雀运行时，您可以将对象 `versionId` 作为路径的一部分，如以下示例所示。

```
https://bucket.s3.amazonaws.com/path/object.zip?versionId=version-id
https://s3.amazonaws.com/bucket/path/object.zip?versionId=version-id
https://bucket.s3-region.amazonaws.com/path/object.zip?versionId=version-id
```

## 堆栈跟踪和异常消息

预设情况下，CloudWatch Synthetics 金丝雀会捕获金丝雀脚本引发的所有异常，无论该脚本是自定义脚本还是蓝图中的脚本。CloudWatch Synthetics 将异常消息和堆栈跟踪记录到三个位置：

- 记录到 CloudWatch Synthetics 服务中，以便在描述测试运行时加快调试速度
- 记录到 CloudWatch Logs 中，具体取决于创建 Lambda 函数时使用的配置
- 记录到 Synthetics 日志文件中，该文件是一个纯文本文件，已上载到由您为金丝雀的 `resultsLocation` 设置的值所指定的 Amazon S3 位置

如果您要发送和存储的信息较少，则可以在异常返回到 CloudWatch Synthetics 包装器库之前捕获异常。

您也可以将请求 URL 包含在错误中。CloudWatch Synthetics 会扫描脚本引发的错误中的任何 URL，并根据 `restrictedUriParameters` 配置编辑其中的受限 URL 参数。如果您在脚本中记录错误消息，则可以使用 `getSanitizedErrorMessage` (p. 256) 在记录日志之前编辑 URL。

## 缩小 IAM 角色的范围

建议您不要将金丝雀配置为访问潜在的恶意 URL 或终端节点。将金丝雀指向不受信任或未知的网站或端点可能会将您的 Lambda 函数代码暴露给恶意用户的脚本。假设一个恶意网站可以破解 Chromium，它可能会以与您使用 Internet 浏览器连接时类似的方式访问您的 Lambda 代码。

使用缩小了权限范围的 IAM 执行角色来运行您的 Lambda 函数。这样，如果 Lambda 函数受到恶意脚本的影响，它在作为金丝雀的 Amazon 账户运行时可采取的操作会受到限制。

在使用 CloudWatch 控制台创建金丝雀时，会使用缩小了权限范围的 IAM 执行角色来进行创建。

## 敏感数据编辑

CloudWatch Synthetics 捕获 URL、状态代码、故障原因（如有）以及请求和响应的标头及请求体和响应体。这使得金丝雀用户能够理解、监控和调试金丝雀。

以下各节中所述配置可以在金丝雀执行的任何时间点进行设置。您还可以选择对不同的 Synthetics 步骤应用不同的配置。

## 请求 URL

预设情况下，CloudWatch Synthetics 会将每个 URL 的请求 URL、状态代码和状态原因记录在金丝雀日志中。请求 URL 也可以出现在金丝雀执行报告、HAR 文件等中。请求 URL 可能会包含敏感查询参数，例如访问令牌或密码。您可以编辑敏感信息，以免被 CloudWatch Synthetics 记录下来。

若要修改敏感信息，请设置配置属性 `restrictedUriParameters`。有关更多信息，请参阅 [SyntheticsConfiguration 类 \(p. 246\)](#)。配置此属性后，CloudWatch Synthetics 会在记录日志前根据 `restrictedUriParameters` 属性修改 URL 参数，包括路径和查询参数值。如果您在脚本中记录 URL，则可以使用 `getSanitizedUrl(url, stepConfig = null)` (p. 255) 在记录日志之前编辑 URL。有关更多信息，请参阅 [SyntheticsLogHelper 类 \(p. 255\)](#)。

## 标头

预设情况下，CloudWatch Synthetics 不记录请求/响应标头。对于 UI 金丝雀，此为使用 `syn-nodejs-puppeteer-3.2` 和更高版本运行时金丝雀的默认行为。

如果标头不包含敏感信息，则可以通过将 `includeRequestHeaders` 和 `includeResponseHeaders` 属性设置为 `true` 来在 HAR 文件和 HTTP 报告中启用标头。您可以启用所有标头，但选择限制敏感标头键的值。例如，您可以选择只编辑金丝雀生成的构件中的 `Authorization` 标头。

## 请求体和响应体

预设情况下，CloudWatch Synthetics 不会将请求/响应体记录在金丝雀日志或报告中。此信息对 API 金丝雀特别有用。Synthetics 捕获所有 HTTP 请求，并可显示标头、请求体和响应体。有关更多信息，请参阅 `executeHttpStep(stepName, requestOptions, [callback], [stepConfig])` (p. 264)。您可以选择将 `includeRequestBody` 和 `includeResponseBody` 属性设置为 `true` 来启用请求/响应体。

# 使用 Amazon CloudTrail 记录 Amazon CloudWatch API 调用

Amazon CloudWatch 和 CloudWatch Synthetics 与 Amazon CloudTrail 集成，后者是一种提供用户、角色或 Amazon 服务所采取操作的记录的服务。CloudTrail 捕获由您的 Amazon 账户发出或代表该账户发出的 API 调用。捕获的调用包含来自控制台的调用和对 API 操作的代码调用。

如果您创建了一个跟踪记录，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 CloudWatch 的事件）。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 CloudWatch 发出的请求内容、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息（包括如何对其进行配置和启用），请参阅 [Amazon CloudTrail 用户指南](#)。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 Amazon Identity and Access Management (IAM) 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 Amazon 服务发出。

有关更多信息，请参阅 [CloudTrail userIdentity 元素](#)。

要持续记录 Amazon 账户中的事件（包括 CloudWatch 和 CloudWatch Synthetics 的事件），请创建跟踪记录。通过跟踪，CloudTrail 可将日志文件传送到 S3 存储桶。默认情况下，在控制台中创建跟踪时，此跟踪应用于所有 Amazon 区域。此跟踪在 Amazon 分区中记录所有区域中的事件，并将日志文件传送到您指定的 S3 存储桶。您可以配置其他 Amazon 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取措施。有关更多信息，请参阅下列内容：

- [创建跟踪概览](#)
- [CloudTrail 支持的服务和集成](#)
- [为 CloudTrail 配置 Amazon SNS 通知](#)
- [从多个区域接收 CloudTrail 日志文件](#)和[从多个账户接收 CloudTrail 日志文件](#)

## 主题

- [CloudTrail 中的 CloudWatch 信息 \(p. 897\)](#)
- [CloudTrail 中的 CloudWatch Synthetics 信息 \(p. 900\)](#)

## CloudTrail 中的 CloudWatch 信息

CloudWatch 支持将以下操作记录为 CloudTrail 日志文件中的事件：

- [删除警报](#)
- [DeleteAnomalyDetector](#)
- [DeleteDashboards](#)
- [DescribeAlarmHistory](#)

- [DescribeAlarms](#)
- [DescribeAlarmsForMetric](#)
- [DescribeAnomalyDetectors](#)
- [DisableAlarmActions](#)
- [EnableAlarmActions](#)
- [GetDashboard](#)
- [ListDashboards](#)
- [PutAnomalyDetector](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [SetAlarmState](#)

## 示例：CloudWatch 日志文件条目

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 PutMetricAlarm 操作。

```
{
 "Records": [{
 "eventVersion": "1.01",
 "userIdentity": {
 "type": "Root",
 "principalId": "EX_PRINCIPAL_ID",
 "arn": "arn:aws:iam::123456789012:root",
 "accountId": "123456789012",
 "accessKeyId": "EXAMPLE_KEY_ID"
 },
 "eventTime": "2014-03-23T21:50:34Z",
 "eventSource": "monitoring.amazonaws.com",
 "eventName": "PutMetricAlarm",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "127.0.0.1",
 "userAgent": "aws-sdk-ruby/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
 "requestParameters": {
 "threshold": 50.0,
 "period": 60,
 "metricName": "CloudTrail Test",
 "evaluationPeriods": 3,
 "comparisonOperator": "GreaterThanThreshold",
 "namespace": "AWS/CloudWatch",
 "alarmName": "CloudTrail Test Alarm",
 "statistic": "Sum"
 },
 "responseElements": null,
 "requestID": "29184022-b2d5-11e3-a63d-9b463e6d0ff0",
 "eventID": "b096d5b7-dcf2-4399-998b-5a53eca76a27"
 },
 ..additional entries
]
}
```

以下日志文件条目显示某个用户调用了 CloudWatch Events PutRule 操作。

```
{
 "eventVersion": "1.03",
 "userIdentity": {
 "type": "Root",
 "principalId": "123456789012",
```

```
 "arn": "arn:aws:iam::123456789012:root",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "sessionContext": {
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2015-11-17T23:56:15Z"
 }
 }
 },
 "eventTime": "2015-11-18T00:11:28Z",
 "eventSource": "events.amazonaws.com",
 "eventName": "PutRule",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS CloudWatch Console",
 "requestParameters": {
 "description": "",
 "name": "ctest2",
 "state": "ENABLED",
 "eventPattern": "{\"source\": [\"aws.ec2\"], \"detail-type\": [\"EC2 Instance State-change Notification\"]}",
 "scheduleExpression": ""
 },
 "responseElements": {
 "ruleArn": "arn:aws:events:us-east-1:123456789012:rule/ctest2"
 },
 "requestID": "e9caf887-8d88-11e5-a331-3332aa445952",
 "eventID": "49d14f36-6450-44a5-a501-b0fdcdfaeb98",
 "eventType": "AwsApiCall",
 "apiVersion": "2015-10-07",
 "recipientAccountId": "123456789012"
}
```

以下日志文件条目显示某个用户调用了 CloudWatch Logs CreateExportTask 操作。

```
{
 "eventVersion": "1.03",
 "userIdentity": {
 "type": "IAMUser",
 "principalId": "EX_PRINCIPAL_ID",
 "arn": "arn:aws:iam::123456789012:user/someuser",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "userName": "someuser"
 },
 "eventTime": "2016-02-08T06:35:14Z",
 "eventSource": "logs.amazonaws.com",
 "eventName": "CreateExportTask",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "127.0.0.1",
 "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
 "requestParameters": {
 "destination": "yourdestination",
 "logGroupName": "yourloggroup",
 "to": 123456789012,
 "from": 0,
 "taskName": "yourtask"
 },
 "responseElements": {
 "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
 },
 "requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
 "eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
 "eventType": "AwsApiCall",
}
```

```
"apiVersion": "20140328",
"recipientAccountId": "123456789012"
}
```

## CloudTrail 中的 CloudWatch Synthetics 信息

CloudWatch Synthetics 支持在 CloudTrail 日志文件中将以下操作记录为事件：

- [CreateCanary](#)
- [DeleteCanary](#)
- [DescribeCanaries](#)
- [DescribeCanariesLastRun](#)
- [DescribeRuntimeVersions](#)
- [GetCanary](#)
- [GetCanaryRuns](#)
- [ListTagsForResource](#)
- [StartCanary](#)
- [StopCanary](#)
- [TagResource](#)
- [UntagResource](#)
- [UpdateCanary](#)

## 示例：CloudWatch Synthetics 日志文件条目

下面的示例显示了一个 CloudTrail Synthetics 日志条目，该条目说明了 DescribeCanaries 操作。

```
{
 "eventVersion": "1.05",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "EX_PRINCIPAL_ID",
 "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "EX_PRINCIPAL_ID",
 "arn": "arn:aws:iam::111222333444:role/Administrator",
 "accountId": "123456789012",
 "userName": "SAMPLE_NAME"
 },
 "webIdFederationData": {},
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2020-04-08T21:43:24Z"
 }
 }
 },
 "eventTime": "2020-04-08T23:06:47Z",
 "eventSource": "synthetics.amazonaws.com",
 "eventName": "DescribeCanaries",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "127.0.0.1",
}
```



```
 "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
java/1.8.0_212 vendor/Oracle_Corporation",
 "requestParameters": null,
 "responseElements": null,
 "requestID": "201ed5f3-15db-4f87-94a4-123456789",
 "eventID": "73ddbd81-3dd0-4ada-b246-123456789",
 "readOnly": true,
 "eventType": "AwsApiCall",
 "recipientAccountId": "11112223333"
}
```

下面的示例显示了一个 CloudTrail Synthetics 日志条目，该条目说明了 UpdateCanary 操作。

```
{
 "eventVersion": "1.05",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "EX_PRINCIPAL_ID",
 "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "EX_PRINCIPAL_ID",
 "arn": "arn:aws:iam::111222333444:role/Administrator",
 "accountId": "123456789012",
 "userName": "SAMPLE_NAME"
 },
 "webIdFederationData": {},
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2020-04-08T21:43:24Z"
 }
 }
 },
 "eventTime": "2020-04-08T23:06:47Z",
 "eventSource": "synthetics.amazonaws.com",
 "eventName": "UpdateCanary",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "127.0.0.1",
 "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
java/1.8.0_212 vendor/Oracle_Corporation",
 "requestParameters": {
 "Schedule": {
 "Expression": "rate(1 minute)"
 },
 "name": "sample_canary_name",
 "Code": {
 "Handler": "myOwnScript.handler",
 "ZipFile": "SAMPLE_ZIP_FILE"
 }
 },
 "responseElements": null,
 "requestID": "fe4759b0-0849-4e0e-be71-1234567890",
 "eventID": "9dc60c83-c3c8-4fa5-bd02-1234567890",
 "readOnly": false,
 "eventType": "AwsApiCall",
 "recipientAccountId": "11112223333"
}
```

下面的示例显示了一个 CloudTrail Synthetics 日志条目，该条目说明了 GetCanaryRuns 操作。

```
{
 "eventVersion": "1.05",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "EX_PRINCIPAL_ID",
 "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "EX_PRINCIPAL_ID",
 "arn": "arn:aws:iam::111222333444:role/Administrator",
 "accountId": "123456789012",
 "userName": "SAMPLE_NAME"
 },
 "webIdFederationData": {},
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2020-04-08T21:43:24Z"
 }
 }
 },
 "eventTime": "2020-04-08T23:06:30Z",
 "eventSource": "synthetics.amazonaws.com",
 "eventName": "GetCanaryRuns",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "127.0.0.1",
 "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
Linux/4.9.184-0.1.ac.235.83.329.metall1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
java/1.8.0_212 vendor/Oracle_Corporation",
 "requestParameters": {
 "Filter": "TIME_RANGE",
 "name": "sample_canary_name",
 "FilterValues": [
 "2020-04-08T23:00:00.000Z",
 "2020-04-08T23:10:00.000Z"
]
 },
 "responseElements": null,
 "requestID": "2f56318c-cfbd-4b60-9d93-1234567890",
 "eventID": "52723fd9-4a54-478c-ac55-1234567890",
 "readOnly": true,
 "eventType": "AwsApiCall",
 "recipientAccountId": "111122223333"
}
```

# 标记 Amazon CloudWatch 资源

标签 是您或 Amazon 分配给 Amazon 资源的自定义属性标签。每个标签具有两个部分：

- 标签键（例如，`CostCenter`、`Environment` 或 `Project`）。标签键区分大小写。
- 一个称为标签值的可选字段（例如，`111122223333` 或 `Production`）。省略标签值与使用空字符串相同。与标签键一样，标签值区分大小写。

标签可帮助您：

- 标识和整理您的 Amazon 资源。许多 Amazon 服务支持标记，因此，您可以将同一标签分配给来自不同服务的资源，以指示这些资源是相关的。例如，您可以将您分配给 EC2 实例的同一标签分配给 CloudWatch 规则。
- 跟踪您的 Amazon 成本。您可以在 Amazon Billing and Cost Management 控制面板上激活这些标签。Amazon 使用标签对您的成本进行分类，并向您提供每月成本分配报告。有关更多信息，请参阅 [Amazon Billing 用户指南](#) 中的 [使用成本分配标签](#)。

以下各节提供有关 CloudWatch 的标签的更多信息。

## CloudWatch 中支持的资源

CloudWatch 中的以下资源支持标记：

- 警报 – 您可以使用 [tag-resource](#) Amazon CLI 和 [TagResource](#) API 标记告警。
- Canary – 您可以使用 CloudWatch 控制台标记 Canary。有关更多信息，请参阅 [创建金丝雀 \(p. 219\)](#)。
- Contributor Insights 规则 – 您可以在创建 Contributor Insights 规则时使用 [put-insight-rule](#) Amazon CLI 命令和 [PutInsightRule](#) API 来标记它们。您可以使用 [tag-resource](#) Amazon CLI 命令和 [TagResource](#) API 向现有规则添加标签。
- 指标流 – 您可以在创建指标流时使用 [put-metric-stream](#) Amazon CLI 命令和 [PutMetricStream](#) API 来标记它们。您可以使用 [tag-resource](#) Amazon CLI 命令和 [TagResource](#) API 向现有指标流添加标签。

有关添加和管理标签的信息，请参阅 [管理标签 \(p. 903\)](#)。

## 管理标签

标签由资源上的 `Key` 和 `Value` 属性构成。您可以使用 CloudWatch 控制台、Amazon CLI 或 CloudWatch API 添加、编辑或删除这些属性的值。有关使用标签的信息，请参阅以下内容：

- Amazon CloudWatch API 引用中的 [TagResource](#)、[UntagResource](#) 和 [ListTagsForResource](#)
- Amazon CloudWatch CLI 引用中的 [tag-resource](#)、[untag-resource](#) 和 [list-tags-for-resource](#)
- Resource Groups 用户指南中的 [使用标签编辑器](#)

## 标签命名和使用约定

以下基本命名和使用约定适用于将标签与 CloudWatch 资源一起使用的情况：

- 每个资源最多可以有 50 个标签。
- 对于每个资源，每个标签键都必须是唯一的，每个标签键只能有一个值。
- 最大标签键长度为 128 个 Unicode 字符 (采用 UTF-8 格式)。
- 最大标签值长度为 256 个 Unicode 字符 (采用 UTF-8 格式)。
- 允许使用的字符包括可用 UTF-8 格式表示的字母、数字和空格，以及以下字符：`.:+=@_/-` (连字符)。
- 标签键和值区分大小写。最佳实践是，决定利用标签的策略并在所有资源类型中一致地实施该策略。例如，决定是否使用 `Costcenter`、`costcenter` 或 `CostCenter`，以及是否对所有标签使用相同的约定。避免将类似的标签用于不一致的案例处理。
- 对标签禁止使用 `aws:` 前缀，因为它是为使用 Amazon 而保留的。您无法编辑或删除带此前缀的标签键或值。具有此前缀的标签不计入每个资源的标签数限制。

# Grafana 集成

您可以使用 Grafana 版本 6.5.0 及更高版本，通过 CloudWatch 控制台在上下文中前进，并使用通配符查询指标的动态列表。这可以帮助您监控 Amazon 资源（例如 Amazon Elastic Compute Cloud 实例或容器）的指标。当新实例作为 Auto Scaling 事件的一部分创建时，它们将自动显示在图形中。您无需跟踪新实例 ID。预构建的控制面板可帮助简化监控 Amazon EC2、Amazon Elastic Block Store 和 Amazon Lambda 资源的入门体验。

您可以使用 Grafana 版本 7.0 及更高版本对 CloudWatch Logs 中的日志组执行 CloudWatch Logs Insights 查询。您可以在条形图、折线图和堆叠图形中以及以表格格式显示查询结果。有关 CloudWatch Logs Insights 的更多信息，请参阅[使用 CloudWatch Logs Insights 分析日志数据](#)。

有关如何入门的更多信息，请参阅 Grafana Labs 文档中的[在 Grafana 中使用 Amazon CloudWatch](#)。

# CloudWatch 服务配额

CloudWatch 对指标、告警、API 请求和告警电子邮件通知有以下配额。

| 资源                          | 默认配额                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 告警操作                        | 每个警报 5 个。无法更改此配额。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 告警                          | <p>每月为每个客户免费提供 10 个。额外的告警会产生费用。</p> <p>每个账户的告警总数没有限制。</p> <p>基于指标数学表达式的警报可以拥有最多 10 个指标。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 异常检测模型                      | 每账户每区域 500 个。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| API 请求数量                    | 每月为每个客户免费提供 1,000,000 个。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 金丝雀                         | <p>每账户每区域 200 个。</p> <p>您可以<a href="#">请求提高配额</a>。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Contributor Insights API 请求 | <p>以下 API 具有每个区域每秒 20 个事务 ( TPS ) 的配额。</p> <ul style="list-style-type: none"> <li>• <a href="#">DescribeInsightRules</a></li> </ul> <p>此配额无法更改。</p> <ul style="list-style-type: none"> <li>• <a href="#">GetInsightRuleReport</a></li> </ul> <p>您可以<a href="#">请求提高配额</a>。</p> <p>以下 API 具有每个区域 5 TPS 的配额。无法更改此配额。</p> <ul style="list-style-type: none"> <li>• <a href="#">DeleteInsightRules</a></li> <li>• <a href="#">PutInsightRule</a></li> </ul> <p>以下 API 具有每个区域 1 TPS 的配额。无法更改此配额。</p> <ul style="list-style-type: none"> <li>• <a href="#">DisableInsightRules</a></li> <li>• <a href="#">EnableInsightRules</a></li> </ul> |
| Contributor Insights 规则     | <p>每区域每账户 100 条规则。</p> <p>您可以<a href="#">请求提高配额</a>。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 自定义指标                       | 没有配额。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 控制面板                        | <p>每个控制面板小部件最多 500 个指标。每个控制面板中的所有小部件最多 2500 个指标。</p> <p>这些配额包括检索的用于指标数学函数的所有指标，即使这些指标未显示在图表上也是如此。</p> <p>这些配额不能更改。</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

| 资源                                                                                                                                                                                                | 默认配额                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">DescribeAlarms</a>                                                                                                                                                                    | 每个区域 9 个事务每秒 (TPS)。在不受限制的情况下，每秒可发出的操作请求的最大数目。<br><br>您可以 <a href="#">请求提高配额</a> 。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <a href="#">DeleteAlarms</a> 请求<br><a href="#">DescribeAlarmHistory</a> 请求<br><a href="#">DisableAlarmActions</a> 请求<br><a href="#">EnableAlarmActions</a> 请求<br><a href="#">SetAlarmState</a> 请求 | 对于其中每个操作，每个区域 3 TPS。在不受限制的情况下，每秒可发出的操作请求的最大数目。<br><br>这些配额不能更改。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <a href="#">DescribeAlarmsForMetric</a> 请求                                                                                                                                                        | 每个区域 9 TPS。在不受限制的情况下，每秒可发出的操作请求的最大数目。<br><br>无法更改这些配额。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <a href="#">DeleteDashboards</a> 请求<br><a href="#">GetDashboard</a> 请求<br><a href="#">ListDashboards</a> 请求<br><a href="#">PutDashboard</a> 请求                                                    | 对于其中每个操作，每个区域 10 TPS。在不受限制的情况下，每秒可发出的操作请求的最大数目。<br><br>这些配额不能更改。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <a href="#">PutAnomalyDetector</a><br><a href="#">DescribeAnomalyDetectors</a>                                                                                                                    | 每个区域 10 TPS。在不受限制的情况下，每秒可发出的操作请求的最大数目。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <a href="#">DeleteAnomalyDetector</a>                                                                                                                                                             | 每个区域 5 TPS。在不受限制的情况下，每秒可发出的操作请求的最大数目。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 维度                                                                                                                                                                                                | 30/指标。无法更改此配额。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <a href="#">GetMetricData</a>                                                                                                                                                                     | <p>对于包含 Metrics Insights 查询的操作，每个区域 10 TPS。对于不包含 Metrics Insights 查询的操作，配额为每个区域 50 TPS。这是您在不受限制的情况下，每秒可发出的最大操作请求数。您可以<a href="#">请求提高配额</a>。</p> <p>对于包含 Metrics Insights 查询的 <a href="#">GetMetricData</a> 操作，最近 3 小时的配额为每秒 4,300,000 个数据点 (DPS)。这是根据查询扫描的数据点总数计算得出的（其中可包括最多 10,000 个指标。）</p> <p>如果在 API 中使用的 <code>StartTime</code> 小于或等于从当前时间开始 3 小时，则为 180,000 每秒数据点 (DPS)。如果 <code>StartTime</code> 大于从当前时间开始 3 小时，则为 396,000 DPS。这是您可以使用一个或多个 API 调用（而不被限制）每秒请求的最大数据点数。无法更改此配额。</p> <p>DPS 是根据估计数据点而非实际数据点计算的。使用请求的时间范围、时间段和保留期计算数据点估计值。这意味着，如果请求指标中的实际数据点稀疏或为空，那么，当估计数据点超过配额时，仍会发生限制。DPS 配额是按每个区域计算的。</p> |

| 资源                                   | 默认配额                                                                                                                                                                                          |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">GetMetricData</a>        | <p>单个 <a href="#">GetMetricData</a> 调用最多可以包含 500 个 <a href="#">MetricDataQuery</a> 结构。</p> <p>无法更改此配额。</p>                                                                                    |
| <a href="#">GetMetricStatistics</a>  | <p>每个区域 400 TPS。在不受限制的情况下，每秒可发出的操作请求的最大数目。</p> <p>您可以<a href="#">请求提高配额</a>。</p>                                                                                                              |
| <a href="#">GetMetricWidgetImage</a> | <p>每幅图像最多 500 个指标。无法更改此配额。</p> <p>每个区域 20 TPS。在不受限制的情况下，每秒可发出的操作请求的最大数目。无法更改此配额。</p>                                                                                                          |
| <a href="#">ListMetrics</a>          | <p>每个区域 25 TPS。在不受限制的情况下，每秒可发出的操作请求的最大数目。</p> <p>您可以<a href="#">请求提高配额</a>。</p>                                                                                                               |
| 指标数据存储                               | 15 个月。无法更改此配额。                                                                                                                                                                                |
| 指标数据值                                | 指标数据点的值必须在 $-2^{360}$ 到 $2^{360}$ 的范围内。不支持特殊值（例如，NaN、+无穷大、-无穷大）。无法更改此配额。                                                                                                                      |
| <a href="#">MetricDatum</a> 项目       | 每个 <a href="#">PutMetricData</a> 请求 1000 个。一个 <a href="#">MetricDatum</a> 对象可包含一个值或代表多个值的一个 <a href="#">StatisticSet</a> 对象。无法更改此配额。                                                          |
| 指标                                   | 每月为每个客户免费提供 10 个。                                                                                                                                                                             |
| Metrics Insights 查询                  | <p>单个查询可处理的指标不超过 10,000 个。这意味着，如果 SELECT、FROM 和 WHERE 子句匹配 10,000 个以上指标，查询只会处理找到的前 10,000 个指标。</p> <p>单个查询可返回的时间序列不超过 500 个。</p>                                                              |
| 告警评估期                                | 通过将告警周期乘以使用的评估周期数计算的最大值为 1 天（86,400 秒）。无法更改此配额。                                                                                                                                               |
| <a href="#">PutMetricAlarm</a> 请求    | <p>每个区域 3 TPS。在不受限制的情况下，每秒可发出的操作请求的最大数目。</p> <p>您可以<a href="#">请求提高配额</a>。</p>                                                                                                                |
| <a href="#">PutMetricData</a> 请求     | <p>对于 HTTP POST 请求为 1 MB。<a href="#">PutMetricData</a> 每秒可处理 500 个事务 (TPS)，这是您每秒可以发出而不会受到限制的操作请求的最大数量。<a href="#">PutMetricData</a> 可以处理每个请求 1,000 个指标。</p> <p>您可以<a href="#">请求提高配额</a>。</p> |
| Amazon SNS 电子邮件通知                    | 每月为每个客户免费提供 1,000 个。                                                                                                                                                                          |
| Synthetics Groups                    | <p>每个账户 20。</p> <p>您可以<a href="#">请求提高配额</a>。</p>                                                                                                                                             |



# 文档历史记录

下表列出了从 2018 年 6 月开始的每个版本的 Amazon CloudWatch 用户指南中的重要更改。如需对此文档更新的通知，您可以订阅 RSS 源。

| 变更                                                                                                             | 说明                                                                                                                                              | 日期               |
|----------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">Amazon CloudWatch Application Insights 对 Container Insights 控制台中的容器化应用程序和微服务的支持。</a> (p. 508)      | Container Insights 控制面板上可以显示 CloudWatch Application Insights 检测到的 Amazon ECS 和 Amazon EKS 的问题。                                                  | 2021 年 11 月 17 日 |
| <a href="#">适用于 SAP HANA 数据库的 Amazon CloudWatch Application Insights 监控。</a> (p. 508)                          | 您可以使用 Application Insights 监控 SAP HANA 数据库。                                                                                                     | 2021 年 11 月 15 日 |
| <a href="#">Amazon CloudWatch Application Insights 对监控账户中的所有资源的支持。</a> (p. 508)                                | 您可以载入和监控账户中的所有资源。                                                                                                                               | 2021 年 9 月 15 日  |
| <a href="#">Amazon CloudWatch Application Insights 对 Amazon FSx 的支持。</a> (p. 508)                              | 您可以监控从 Amazon FSx 检索的指标。                                                                                                                        | 2021 年 8 月 31 日  |
| <a href="#">不再支持软件开发工具包指标。</a> (p. 909)                                                                        | 不再支持 CloudWatch 软件开发工具包指标。                                                                                                                      | 2021 年 8 月 25 日  |
| <a href="#">Amazon CloudWatch Application Insights 对设置容器监控的支持。</a> (p. 508)                                    | 您可以使用 Amazon CloudWatch Application Insights 最佳实践监控容器。                                                                                          | 2021 年 5 月 18 日  |
| <a href="#">已正式发布指标流</a> (p. 909)                                                                              | 您可以使用指标流，持续地将 CloudWatch 指标流式传输到您所选的目标位置。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">指标流</a> 。                                              | 2021 年 3 月 31 日  |
| <a href="#">适用于 Amazon RDS 和 Amazon EC2 上的 Oracle 数据库的 Amazon CloudWatch Application Insights 监控。</a> (p. 508) | 您可以使用 Amazon CloudWatch Application Insights 监控从 Oracle 检索到的指标和日志。                                                                              | 2021 年 1 月 16 日  |
| <a href="#">Lambda Insights 已正式发布</a> (p. 909)                                                                 | CloudWatch Lambda Insights 是针对在 Amazon Lambda 上运行的无服务器应用程序的监控和故障排除解决方案。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">使用 Lambda Insights</a> 。 | 2020 年 12 月 3 日  |
| <a href="#">适用于 Prometheus JMX Exporter 指标的 Amazon CloudWatch Application Insights 监控。</a> (p. 508)            | 您可用使用 Amazon CloudWatch Application Insights 监控从 Prometheus JMX Exporter 检索到的指标。                                                                | 2020 年 11 月 20 日 |
| <a href="#">CloudWatch Synthetics 发布新的运行时版本</a> (p. 909)                                                       | CloudWatch Synthetics 已发布新的运行时版本。有关更多信息，请                                                                                                       | 2020 年 9 月 11 日  |

|                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                    |                  |
|-----------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
|                                                                                                                 | 参阅 Amazon CloudWatch 用户指南中的 <a href="#">Canary 运行时版本</a> 。                                                                                                                                                                                                                                                                                                         |                  |
| <a href="#">适用于 Amazon RDS 和 Amazon EC2 上的 PostgreSQL 的 Amazon CloudWatch Application Insights 监控。</a> (p. 508) | 您可以监控运行在 Amazon RDS 或 Amazon EC2 上的使用 PostgreSQL 构建的应用程序。                                                                                                                                                                                                                                                                                                          | 2020 年 9 月 11 日  |
| <a href="#">CloudWatch 支持控制面板共享</a> (p. 909)                                                                    | 现在，您可以与企业 and Amazon 账户之外的人员共享 CloudWatch 控制面板。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">共享 CloudWatch 控制面板</a> 。                                                                                                                                                                                                                                            | 2020 年 9 月 10 日  |
| <a href="#">通过 CloudWatch Application Insights 在后端使用 SQL Server 为 .NET 应用程序设置监控</a> (p. 508)                    | 您可用使用文档教程帮助您通过 CloudWatch Application Insights 在后端使用 SQL Server 为 .NET 应用程序设置监控。                                                                                                                                                                                                                                                                                   | 2020 年 8 月 19 日  |
| <a href="#">Amazon CloudFormation 对 Amazon CloudWatch Application Insights 应用程序的支持。</a> (p. 508)                | 您可以直接从 Amazon CloudFormation 模板中将 CloudWatch Application Insights 监控 (包括关键指标和遥测) 添加到应用程序、数据库和 Web 服务器。                                                                                                                                                                                                                                                             | 2020 年 7 月 30 日  |
| <a href="#">适用于 MySQL 数据库集群可用的 Aurora 的 Amazon CloudWatch Application Insights 监控。</a> (p. 508)                 | 您可用使用 Amazon CloudWatch Application Insights 监控适用于 MySQL 数据库集群 (RDS Aurora) 的 Aurora。                                                                                                                                                                                                                                                                              | 2020 年 7 月 2 日   |
| <a href="#">CloudWatch Contributor Insights 正式发布</a> (p. 909)                                                   | CloudWatch Contributor Insights 现已正式发布。它使您能够分析日志数据并创建显示贡献者数据的时间序列。您可以查看有关前 N 个贡献者、独特贡献者总数及其使用情况的指标。有关更多信息，请参阅《Amazon CloudWatch 用户指南》 <a href="https://docs.amazonaws.cn/AmazonCloudWatch/latest/monitoring/ContributorInsights.html">https://docs.amazonaws.cn/AmazonCloudWatch/latest/monitoring/ContributorInsights.html</a> 中的使用 Contributor Insights 分析高基数数据。 | 2020 年 4 月 2 日   |
| <a href="#">CloudWatch Synthetics 公开预览版</a> (p. 909)                                                            | CloudWatch Synthetics 现在处于公开预览版。它使您能够创建 Canary 来监控终端节点和 API。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">使用 Canary</a> 。                                                                                                                                                                                                                                        | 2019 年 11 月 25 日 |

|                                                |                                                                                                                                                                                                                                                                                                                                                                       |                  |
|------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| CloudWatch Contributor Insights 公开预览版 (p. 909) | CloudWatch Contributor Insights 现在处于公开预览版。它使您能够分析日志数据并创建显示贡献者数据的时间序列。您可以查看有关前 N 个贡献者、独特贡献者总数及其使用情况的指标。有关更多信息，请参阅《Amazon CloudWatch 用户指南》 <a href="https://docs.amazonaws.cn/AmazonCloudWatch/latest/monitoring/ContributorInsights.html">https://docs.amazonaws.cn/AmazonCloudWatch/latest/monitoring/ContributorInsights.html</a> 中的使用 Contributor Insights 分析高基数数据。 | 2019 年 11 月 25 日 |
| CloudWatch 启动 ServiceLens 功能 (p. 909)          | ServiceLens 使您能够将跟踪、指标、日志和警报集成到一个位置，从而提高服务和应用程序的可观察性。ServiceLens 将 CloudWatch 与 Amazon X-Ray 集成，以提供应用程序的端到端视图。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">使用 ServiceLens 监控应用程序的运行状况</a> 。                                                                                                                                                                        | 2019 年 11 月 21 日 |
| 使用 CloudWatch 主动管理您的 Amazon 服务限额 (p. 909)      | 使用 CloudWatch 主动管理您的 Amazon 服务配额。CloudWatch 使用情况指标可帮助您了解您账户的资源使用情况和 API 操作。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">Service Quotas 集成和使用情况指标</a> 。                                                                                                                                                                                                             | 2019 年 11 月 19 日 |
| 当告警状态更改时，CloudWatch 会发送事件 (p. 909)             | 现在，当任何 CloudWatch 告警更改状态时，CloudWatch 会向 Amazon EventBridge 发送事件。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">告警事件和 EventBridge</a> 。                                                                                                                                                                                                                               | 2019 年 10 月 8 日  |
| Container Insights (p. 909)                    | CloudWatch Container Insights 现已正式发布。它让您能够从容器化应用程序和微服务中收集、聚合和汇总指标与日志。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">使用 Container Insights</a> 。                                                                                                                                                                                                                      | 2019 年 8 月 30 日  |

|                                                                  |                                                                                                                                                                                                                                                                         |                 |
|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| Amazon EKS 和 Kubernetes 上的 Container Insights 预览指标更新 (p. 909)    | Amazon EKS 和 Kubernetes 上的 Container Insights 公开预览版已更新。InstanceID 现在作为一个维度包含到集群 EC2 实例中。这允许在这些指标上创建的警报触发以下 EC2 操作：停止、终止、重新启动或恢复。此外，Kubernetes 命名空间现在报告了 pod 和服务指标，以简化命名空间对指标的监控和警报。                                                                                     | 2019 年 8 月 19 日 |
| 针对 Amazon Systems Manager OpsCenter 集成的更新 (p. 909)               | 关于 CloudWatch Application Insights 如何与 Systems Manager OpsCenter 集成的更新。                                                                                                                                                                                                 | 2019 年 8 月 7 日  |
| CloudWatch 使用情况指标 (p. 909)                                       | CloudWatch 使用情况指标可帮助您跟踪 CloudWatch 资源的使用情况，并保持在服务限制范围内。有关更多信息，请参阅 <a href="https://docs.amazonaws.cn/AmazonCloudWatch/latest/monitoring/CloudWatch-Usage-Metrics.html">https://docs.amazonaws.cn/AmazonCloudWatch/latest/monitoring/CloudWatch-Usage-Metrics.html</a> 。 | 2019 年 8 月 6 日  |
| CloudWatch Container Insights 公开预览版 (p. 909)                     | CloudWatch Container Insights 现在处于公开预览版。它让您能够从容器化应用程序和微服务中收集、聚合和汇总指标与日志。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">使用 Container Insights</a> 。                                                                                                                     | 2019 年 7 月 9 日  |
| CloudWatch 异常检测公开预览版 (p. 909)                                    | CloudWatch 异常检测现在处于公开预览版。CloudWatch 将机器学习算法应用于指标的过去数据，以创建指标的预期值模型。您可以使用此模型进行可视化和设置警报。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">使用 CloudWatch 异常检测</a> 。                                                                                                           | 2019 年 7 月 9 日  |
| 适用于 .NET 和 SQL Server 的 CloudWatch Application Insights (p. 909) | 适用于 .NET 和 SQL Server 的 CloudWatch Application Insights 简化了观察 .NET 和 SQL Server 应用程序的过程。它可以帮助您为应用程序资源设置最佳的监视器以持续分析数据，以便找出应用程序出现问题的迹象。                                                                                                                                   | 2019 年 6 月 21 日 |
| 已重新组织 CloudWatch 代理部分 (p. 909)                                   | 已重新编写 CloudWatch 代理文档以提高清晰性，尤其是对于使用命令行安装和配置代理的客户。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">使用 CloudWatch 代理从 Amazon EC2 实例和本地服务器中收集指标和日志</a> 。                                                                                                                    | 2019 年 3 月 28 日 |

|                                                                  |                                                                                                                                                                       |                  |
|------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">SEARCH 函数已添加到指标数学表达式 (p. 909)</a>                    | 现在，您可以在指标数学表达式中使用 SEARCH 函数。这使您能够创建在创建与搜索查询匹配的新资源时自动更新的控制面板。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">在图表中使用搜索表达式</a> 。                                         | 2019 年 3 月 21 日  |
| <a href="#">企业支持的 Amazon SDK 指标 (p. 909)</a>                     | 软件开发工具包指标可帮助您评估 Amazon 服务的运行状况和诊断因达到账户用量限制或服务中断而引起的延迟。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">使用 Amazon SDK 指标监控应用程序</a> 。                                    | 2018 年 12 月 11 日 |
| <a href="#">针对数学表达式的告警 (p. 909)</a>                              | CloudWatch 支持根据指标数学表达式创建告警。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">针对数学表达式的告警</a> 。                                                                           | 2018 年 11 月 20 日 |
| <a href="#">新的 CloudWatch 控制台主页 (p. 909)</a>                     | Amazon 在 CloudWatch 控制台中创建了一个新的主页，此主页会自动显示您所使用的所有 Amazon 服务的关键指标和告警。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">Amazon CloudWatch 入门</a> 。                        | 2018 年 11 月 19 日 |
| <a href="#">CloudWatch 代理的 Amazon CloudFormation 模板 (p. 909)</a> | Amazon 已上传 Amazon CloudFormation 模板，您可以使用这些模板安装和更新 CloudWatch 代理。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">使用 Amazon CloudFormation 在新实例上安装 CloudWatch 代理</a> 。 | 2018 年 11 月 9 日  |
| <a href="#">CloudWatch 代理的增强功能 (p. 909)</a>                      | CloudWatch 代理已更新为使用 StatsD 和 collectd 协议。它还改进了跨账户支持。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">使用 StatsD 检索自定义指标、使用 collectd 检索自定义指标和将指标和日志发送到不同 Amazon 账户</a> 。 | 2018 年 9 月 28 日  |
| <a href="#">对 Amazon VPC 终端节点的支持 (p. 909)</a>                    | 您现在可以在 VPC 和 CloudWatch 之间建立私有连接。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 <a href="#">将 CloudWatch 与接口 VPC 终端节点结合使用</a> 。                                                  | 2018 年 6 月 28 日  |

下表介绍了 2018 年 6 月之前对 Amazon CloudWatch 用户指南的重要更改。

| 更改                                    | 描述                                                                                                                                                                                                                    | 发行日期             |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 指标数学                                  | 现在，您可以对 CloudWatch 指标执行数学表达式，从而生成可添加到控制面板上的图表中的新时间序列。有关更多信息，请参阅 <a href="#">使用指标数学 (p. 116)</a> 。                                                                                                                     | 2018 年 4 月 4 日   |
| “M out of N”警报                        | 现在，您可以将警报配置为根据任何警报评估间隔的 N 个数据点中的 M 个数据点触发。有关更多信息，请参阅 <a href="#">评估告警 (p. 152)</a> 。                                                                                                                                  | 2017 年 12 月 8 日  |
| CloudWatch 代理                         | 已发布新的统一 CloudWatch 代理。您可以使用统一多平台代理从 Amazon EC2 实例和本地服务器中收集自定义系统指标和日志文件。新代理支持 Windows 和 Linux 并允许自定义收集的指标，包括子资源指标，例如，每个 CPU 的内核数。有关更多信息，请参阅 <a href="#">使用 CloudWatch 代理从 Amazon EC2 实例和本地部署服务器中收集指标和日志 (p. 659)</a> 。 | 2017 年 9 月 7 日   |
| NAT 网关指标                              | 已为 Amazon VPC NAT 网关添加指标。                                                                                                                                                                                             | 2017 年 9 月 7 日   |
| 高精度指标                                 | 现在，您可以选择将自定义指标设置为高精度指标，其粒度可以低至一秒。有关更多信息，请参阅 <a href="#">高精度指标 (p. 147)</a> 。                                                                                                                                          | 2017 年 26 月 7 日  |
| 控制面板 API                              | 现在您可以使用 API 和 Amazon CLI 创建、修改和删除控制面板。有关更多信息，请参阅 <a href="#">创建 CloudWatch 控制面板 (p. 29)</a> 。                                                                                                                         | 2017 年 6 月 7 日   |
| Amazon Direct Connect 指标              | 为 Amazon Direct Connect 添加了指标。                                                                                                                                                                                        | 2017 年 6 月 29 日  |
| Amazon VPC VPN 指标                     | 已为 Amazon VPC VPN 添加指标。                                                                                                                                                                                               | 2017 年 5 月 15 日  |
| AppStream 2.0 指标                      | 已为 AppStream 2.0 添加指标。                                                                                                                                                                                                | 2017 年 3 月 8 日   |
| CloudWatch 控制台颜色选取器                   | 您现在可以在控制面板小部件上选择每个指标的颜色。有关更多信息，请参阅 <a href="#">在 CloudWatch 控制面板上编辑图表 (p. 36)</a> 。                                                                                                                                   | 2017 年 2 月 27 日  |
| 控制面板上的警报                              | 现在可将警报添加到控制面板。有关更多信息，请参阅 <a href="#">在 CloudWatch 控制面板上添加或删除警报小部件 (p. 52)</a> 。                                                                                                                                       | 2017 年 2 月 15 日  |
| 已为 Amazon Polly 添加指标                  | 已为 Amazon Polly 添加指标。                                                                                                                                                                                                 | 2016 年 12 月 1 日  |
| 已为 Amazon Kinesis Data Analytics 添加指标 | 已为 Amazon Kinesis Data Analytics 添加指标。                                                                                                                                                                                | 2016 年 12 月 1 日  |
| 增加了对百分位数统计数据的支持                       | 您可以指定任何百分位数，最多使用两位小数 (例如 p95.45)。有关更多信息，请参阅 <a href="#">百分位数 (p. 6)</a> 。                                                                                                                                             | 2016 年 11 月 17 日 |
| 已为 Amazon Simple Email Service 添加指标   | 已为 Amazon Simple Email Service 添加指标。                                                                                                                                                                                  | 2016 年 11 月 2 日  |

| 更改                                                             | 描述                                                                                                                               | 发行日期             |
|----------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|------------------|
| 更新的指标保留                                                        | Amazon CloudWatch 现在将指标数据保留 15 个月而非 14 天。                                                                                        | 2016 年 11 月 1 日  |
| 更新的指标控制台界面                                                     | 已使用对现有功能的改进和新功能更新 CloudWatch 控制台。                                                                                                | 2016 年 11 月 1 日  |
| 已为 Amazon Elastic Transcoder 添加指标                              | 已为 Amazon Elastic Transcoder 添加指标。                                                                                               | 2016 年 9 月 20 日  |
| 已为 Amazon API Gateway 添加指标                                     | 已为 Amazon API Gateway 添加指标。                                                                                                      | 2016 年 9 月 9 日   |
| 为 Amazon Key Management Service 添加了指标                          | 为 Amazon Key Management Service 添加了指标。                                                                                           | 2016 年 9 月 9 日   |
| 已为由 Elastic Load Balancing 支持的新 Application Load Balancer 添加指标 | 已为 Application Load Balancer 添加指标。                                                                                               | 2016 年 8 月 11 日  |
| 添加了针对 Amazon EC2 的新的 NetworkPacketsIn 和 NetworkPacketsOut 指标   | 添加了针对 Amazon EC2 的新的 NetworkPacketsIn 和 NetworkPacketsOut 指标。                                                                    | 2016 年 3 月 23 日  |
| 已为 Amazon EC2 Spot 实例集添加新指标                                    | 已为 Amazon EC2 Spot 实例集添加新指标。                                                                                                     | 2016 年 3 月 21 日  |
| 已添加新的 CloudWatch Logs 指标                                       | 已添加新的 CloudWatch Logs 指标。                                                                                                        | 2016 年 3 月 10 日  |
| 已添加 Amazon OpenSearch Service、Amazon WAF 指标和维度                 | 已添加 Amazon OpenSearch Service、Amazon WAF 指标和维度。                                                                                  | 2015 年 10 月 14 日 |
| 已添加对 CloudWatch 控制面板的支持                                        | 控制面板是 CloudWatch 控制台中的可自定义主页，可用于在单个视图中监控资源，即便是分布到不同区域的资源，也能对其进行监控。有关更多信息，请参阅 <a href="#">使用 Amazon CloudWatch 控制面板 (p. 29)</a> 。 | 2015 年 10 月 8 日  |
| 添加了 Amazon Lambda 指标和维度                                        | 添加了 Amazon Lambda 指标和维度。                                                                                                         | 2015 年 9 月 4 日   |



| 更改                                                   | 描述                                                                                                                                                                                                                                       | 发行日期            |
|------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 已添加 Amazon Elastic Container Service 指标和维度           | 已添加 Amazon Elastic Container Service 指标和维度。                                                                                                                                                                                              | 2015 年 8 月 17 日 |
| 已添加 Amazon Simple Storage Service 指标和维度              | 已添加 Amazon Simple Storage Service 指标和维度。                                                                                                                                                                                                 | 2015 年 7 月 26 日 |
| 新功能：重启警报操作                                           | 增加了重启警报操作和与警报操作一起使用的新 IAM 角色。有关更多信息，请参阅 <a href="#">创建告警以停止、终止、重启或恢复 EC2 实例 (p. 178)</a> 。                                                                                                                                               | 2015 年 7 月 23 日 |
| 已添加 Amazon WorkSpaces 指标和维度                          | 已添加 Amazon WorkSpaces 指标和维度。                                                                                                                                                                                                             | 2015 年 4 月 30 日 |
| 已添加 Amazon Machine Learning 指标和维度                    | 已添加 Amazon Machine Learning 指标和维度。                                                                                                                                                                                                       | 2015 年 4 月 9 日  |
| 新功能：Amazon EC2 实例恢复告警操作                              | 更新了警报操作以包含 EC2 实例恢复操作。有关更多信息，请参阅 <a href="#">创建告警以停止、终止、重启或恢复 EC2 实例 (p. 178)</a> 。                                                                                                                                                      | 2015 年 3 月 12 日 |
| 已添加 Amazon CloudFront 和 Amazon CloudSearch 指标和维度     | 已添加 Amazon CloudFront 和 Amazon CloudSearch 指标和维度。                                                                                                                                                                                        | 2015 年 3 月 6 日  |
| 已添加 Amazon Simple Workflow Service 指标和维度             | 已添加 Amazon Simple Workflow Service 指标和维度。                                                                                                                                                                                                | 2014 年 5 月 9 日  |
| 更新了指南以增加对 Amazon CloudTrail 的支持                      | 添加了一个新主题，说明如何在 Amazon CloudWatch 中使用 Amazon CloudTrail 记录活动。有关更多信息，请参阅 <a href="#">使用 Amazon CloudTrail 记录 Amazon CloudWatch API 调用 (p. 897)</a> 。                                                                                       | 2014 年 3 月 4 日  |
| 指南更新为使用新的 Amazon Command Line Interface (Amazon CLI) | Amazon CLI 是一个跨服务 CLI，安装得到简化、具有统一配置且采用一致的命令行语法。Linux/Unix、Windows 和 Mac 均支持 Amazon CLI。本指南中的 CLI 示例已更新为使用新的 Amazon CLI。<br><br>有关如何安装和配置新 Amazon CLI 的信息，请参阅 Amazon Command Line Interface 用户指南中的 <a href="#">使用 Amazon CLI 界面进行设置</a> 。 | 2014 年 2 月 21 日 |
| 已添加 Amazon Redshift、Amazon OpsWorks 指标和维度            | 已添加 Amazon Redshift、Amazon OpsWorks 指标和维度。                                                                                                                                                                                               | 2013 年 7 月 16 日 |



| 更改                         | 描述                                                                                                                                             | 发行日期             |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 已添加 Amazon Route 53 指标和维度  | 已添加 Amazon Route 53 指标和维度。                                                                                                                     | 2013 年 1 月 26 日  |
| 新功能：Amazon CloudWatch 告警操作 | 新增了一个区段用于记录 Amazon CloudWatch 告警操作，您可以使用此区段停止或终止 Amazon Elastic Compute Cloud 实例。有关更多信息，请参阅 <a href="#">创建告警以停止、终止、重启或恢复 EC2 实例 (p. 178)</a> 。 | 2013 年 1 月 8 日   |
| 更新的 EBS 指标                 | 已将 EBS 指标更新至包含配置的 IOPS 卷的两项新指标。                                                                                                                | 2012 年 11 月 20 日 |
| 新账单警报                      | 现在可以使用 Amazon CloudWatch 指标监控您的 Amazon 收费并创建告警使其在超过指定阈值时通知您。有关更多信息，请参阅 <a href="#">创建账单告警以监控 Amazon 估算费用 (p. 182)</a> 。                        | 2012 年 5 月 10 日  |
| 新指标                        | 现在可以访问六种新的 Elastic Load Balancing 指标，这些指标提供多种 HTTP 响应代码计数。                                                                                     | 2011 年 10 月 19 日 |
| 新功能                        | 现在可以从 Amazon EMR 访问指标。                                                                                                                         | 2011 年 6 月 30 日  |
| 新功能                        | 现在，您可以从 Amazon Simple Notification Service 和 Amazon Simple Queue Service 访问指标。                                                                 | 2011 年 7 月 14 日  |
| 新功能                        | 添加了关于使用 PutMetricData API 发布自定义指标的信息。有关更多信息，请参阅 <a href="#">发布自定义指标 (p. 147)</a> 。                                                             | 2011 年 5 月 10 日  |
| 更新的指标保留                    | Amazon CloudWatch 现在对告警历史记录保留时间的保留时间从之前的 6 周缩减到 2 周。这一改变让警报的保留时间和指标数据的保留时间相一致。                                                                 | 2011 年 4 月 7 日   |
| 新功能                        | 添加了当指标越过阈值时发送 Amazon Simple Notification Service 或 Auto Scaling 通知的功能。有关更多信息，请参阅 <a href="#">告警 (p. 7)</a> 。                                   | 2010 年 12 月 2 日  |
| 新功能                        | 多种 CloudWatch 操作现在包括 MaxRecords 和 NextToken 参数，让您可以控制要显示的结果页。                                                                                  | 2010 年 12 月 2 日  |
| 新功能                        | 该服务现已集成 Amazon Identity and Access Management ( IAM ) 。                                                                                        | 2010 年 12 月 2 日  |