Amazon EKS

用户指南

Amazon EKS: 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、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 EKS？ 1
Amazon EKS 控制面板架构 1
Amazon EKS 是如何工作的？ 1
定价 2
部署选项 2
开始使用 Amazon EKS 4
安装 kubectl 4
安装 eksctl 10
创建您的第一个集群 – eksctl 13
先决条件 13
第 1 步：创建集群和节点 13
第 2 步：查看 Kubernetes 资源 14
第 3 步：删除集群和节点 15
后续步骤 15
创建您的第一个集群 – Amazon Web Services Management Console 15
先决条件 15
第 1 步：创建集群 16
第 2 步：配置集群通信 17
第 3 步：创建节点 18
第 4 步：查看资源 19
第 5 步：删除资源 19
后续步骤 20
集群 21
创建集群 21
更新 Kubernetes 版本 28
更新 Amazon EKS 集群的 Kubernetes 版本 28
删除集群 35
配置端点访问 38
修改集群终端节点访问 38
访问私有 API 服务器 41
启用密钥加密 42
配置 日志记录 44
启用和禁用控制层面日志 45
查看集群控制层面日志 47
查看 API 服务器标记 47
启用 Windows 支持 48
启用 Windows 支持 49
删除旧版 Windows 支持 50
禁用 Windows 支持 50
部署 Pod 51
启用旧版 Windows 支持 51
私有集群要求 56
要求 56
注意事项 57
创建容器映像的本地副本 58
用于服务账户 IAM 角色的 Amazon STS 端点 58
Kubernetes 版本 59
可用的 Amazon EKS Kubernetes 版本 59
Kubernetes 1.23 59
Kubernetes 1.22 60
Kubernetes 1.21 62
Kubernetes 1.20 63
Kubernetes 1.19 63
Kubernetes 1.18 65

Amazon EKS Kubernetes 发布日历 65
Amazon EKS 版本支持和常见问题 65
平台版本 67
Kubernetes 版本 1.23 67
Kubernetes 版本 1.22 68
Kubernetes 版本 1.21 68
Kubernetes 版本 1.20 69
Kubernetes 版本 1.19 70
Kubernetes 版本 1.18 71
Autoscaling 72
Cluster Autoscaler 72
Karpenter 82
节点 84
托管节点组 87
托管节点组概念 87
托管节点组容量类型 88
创建托管节点组 90
更新托管节点组 95
托管节点组上的节点污点 99
启动模板支持 100
删除托管节点组 105
自行管理的节点 106
Amazon Linux 107
Bottlerocket 112
Windows 114
更新 119
Amazon Fargate 127
Fargate 注意事项 127
Fargate 入门 128
Fargate 配置文件 131
Fargate pod 配置 135
Fargate pod 修补 136
Fargate 指标 137
Fargate 日志记录 138
实例类型 145
最大 pods 146
Amazon EKS 优化版 AMI 146
Dockershim 弃用 147
Amazon Linux 147
Ubuntu Linux 162
Bottlerocket 162
Windows 169
Amazon Outposts 上的 Amazon EKS 节点 189
先决条件 189
Outpost 考虑因素 189
..... 190
在 Amazon Outposts 上部署包含 Worker 节点的 Amazon EKS 集群 190
存储 194
存储类 194
Amazon EBS CSI 驱动程序 195
创建 IAM policy 和角色 196
管理 Amazon EKS 附加组件 201
部署示例 应用程序 206
CSI 迁移常见问题 208
Amazon EFS CSI 驱动程序 210
创建 IAM policy和角色 210
安装 Amazon EFS 驱动程序 212

创建 Amazon EFS 文件系统 213
部署示例应用程序 215
Amazon FSx for Lustre CSI 驱动程序 220
Amazon FSx for NetApp ONTAP CSI 驱动程序 225
联网 226
VPC 和子网要求 226
VPC 要求和注意事项 226
子网要求和注意事项 227
创建 VPC 228
安全组要求 231
Pod 联网 233
管理 Amazon VPC CNI plugin for Kubernetes 233
为 IAM 账户配置插件 243
使用案例 247
指标帮助程序 283
备选的兼容 CNI 插件 287
安装 Amazon Load Balancer Controller 附加组件 287
管理 CoreDNS 附加组件 294
添加 Amazon EKS 附加组件 295
更新 Amazon EKS 附加组件 296
删除 Amazon EKS 附加组件 298
更新 自行管理的附加组件 299
管理 kube-proxy 附加组件 300
添加 Amazon EKS 附加组件 301
更新 Amazon EKS 附加组件 302
删除 Amazon EKS 附加组件 304
更新 自行管理的附加组件 305
安装 Calico 附加组件 306
安装 Calico 307
Stars 策略演示 309
删除 Calico 314
工作负载 315
示例应用程序部署 315
Vertical Pod Autoscaler 321
部署 Vertical Pod Autoscaler 322
测试 Vertical Pod Autoscaler 安装 323
Horizontal Pod Autoscaler 325
运行 Horizontal Pod Autoscaler 测试应用程序 326
网络负载均衡 327
创建网络负载均衡器 329
（可选）部署示例应用程序 330
应用程序负载均衡 332
（可选）部署示例应用程序 335
限制服务外部 IP 地址分配 337
将镜像复制到存储库 338
Amazon 容器镜像注册表 340
Amazon EKS 附加组件 341
附加组件配置 342
Machine Learning 培训 344
创建节点组 345
（可选）部署示例 EFA 兼容应用程序 349
机器学习推理 350
先决条件 350
创建集群 351
（可选）部署 TensorFlow Serving 应用程序映像 351
（可选）根据 TensorFlow Serving 服务进行预测 353
集群身份验证 355

授予用户和角色访问权限 355
添加用户或角色 356
将 aws-authConfigMap 应用到集群 361
OIDC 身份提供商身份验证 362
关联 OIDC 身份提供商 362
解除 OIDC 身份提供商与集群的关联 364
示例 IAM policy 364
为 Amazon EKS 创建 kubeconfig 365
自动创建 kubeconfig 文件 366
手动创建 kubeconfig 文件 367
安装 aws-iam-authenticator 369
默认 Amazon EKS 角色和用户 373
集群管理 374
教程：部署 Kubernetes 控制面板 374
先决条件 374
步骤 1：部署 Kubernetes 控制面板 374
第 2 步：创建 eks-admin 服务账户和集群角色绑定 376
第 3 步：连接到控制面板 376
成本监控 378
删除 Kubecost 380
Metrics Server 380
Prometheus 指标 381
查看原始指标 381
部署 Prometheus 382
将您的 Prometheus 指标存储在 Amazon Managed Service for Prometheus 384
使用 Helm 384
给您的 资源加标签 385
有关标签的基本知识 386
给您的 资源加标签 386
标签限制 386
标记资源以便于计费 387
通过控制台使用标签 387
通过 CLI、API 或 eksctl 使用标签 388
服务限额 389
服务限额 390
Amazon Fargate 服务配额 390
安全性 392
证书签名 392
CSR 示例 393
Kubernetes 服务账户 394
服务账户令牌 394
集群附加组件 395
服务账户的 IAM 角色 395
身份和访问管理 409
Audience 409
使用身份进行身份验证 409
使用策略管理访问 411
Amazon EKS 如何与 IAM 配合使用 412
基于身份的策略示例 415
使用服务相关角色 418
集群 IAM 角色 425
节点 IAM 角色 427
容器执行 IAM 角色 430
Connector IAM 角色 433
Amazon 托管策略 436
问题排查 451
合规性验证 451

故障恢复能力 451
基础设施安全性 452
配置和漏洞分析 452
安全最佳实践 453
Pod 安全策略 453
Amazon EKS 默认 pod 安全策略 453
删除默认策略 454
安装或恢复原定设置策略 455
管理 Kubernetes 密钥 456
Amazon EKS Connector 注意事项 456
Amazon 责任 457
客户责任 457
查看 Kubernetes 资源 458
所需权限 458
可观察性 463
日志记录和监控 463
Amazon EKS 日志记录和监控工具 464
使用 Amazon CloudTrail 记录 Amazon EKS API 调用 465
CloudTrail 中的 Amazon EKS 信息 465
了解 Amazon EKS 日志文件条目 466
对 ADOT Operator 的 Amazon EKS 附加组件支持 468
ADOT 考虑因素 468
Amazon Distro for OpenTelemetry (ADOT) 的先决条件 468
创建 IAM 角色 469
管理 ADOT Operator 470
部署 ADOT 收集器 472
部署示例应用程序 474
使用其他服务 476
利用 Amazon CloudFormation 创建 Amazon EKS 资源 476
Amazon EKS 和 Amazon CloudFormation 模板 476
了解有关 Amazon CloudFormation 的更多信息 476
Amazon EKS 和 Amazon 本地区域 476
Deep Learning Containers 477
故障排除 478
IAM 487
Amazon EKS Connector 问题排查 489
常见问题 489
常见问题 491
基本问题排查 492
ADOT Amazon EKS 附加组件问题排查 493
常见问题 493
Amazon EKS Connector 495
注意事项 495
所需的 IAM 权限 495
连接集群 496
步骤 1：注册集群 496
步骤 2：应用清单文件 498
向用户授予查看集群上的 Kubernetes 资源的访问权限 498
先决条件 498
注销集群 500
相关项目 502
管理工具 502
eksctl 502
Amazon Controllers for Kubernetes 502
Flux CD 502
CDK for Kubernetes 502
联网 502

Amazon VPC CNI plugin for Kubernetes 503
Amazon Load Balancer Controller for Kubernetes 503
ExternalDNS 503
App Mesh 控制器 503
安全性 503
Amazon IAM 身份验证器 503
Machine Learning 504
Kubeflow 504
Auto Scaling 504
Cluster Autoscaler 504
Escalator 504
监控 504
Prometheus 504
持续集成/持续部署 504
Jenkins X 505
Amazon EKS 新功能和路线图 506
文档历史记录 507

什么是 Amazon EKS？

Amazon Elastic Kubernetes Service (Amazon EKS) 是一项托管服务，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、扩展和管理的自动化的开源系统。Amazon EKS：

- 跨多个 Amazon 可用区运行和缩放 Kubernetes 控制面板，确保高可用性。
- 根据负载自动缩放控制面板实例，检测并替换运行状况不良的控制面板实例，并为其提供自动化的版本更新和修补。
- 与许多Amazon服务集成以便为您的应用程序提供可扩展性和安全性，包括以下能力：
  - 用于容器镜像的 Amazon ECR
  - 用于负载分配的 Elastic Load Balancing
  - 用于身份验证的 IAM
  - 用于隔离的 Amazon VPC
- 运行最新版本的开源 Kubernetes 软件，因此您可以使用 Kubernetes 社群的所有现有插件和工具。在 Amazon EKS 上运行的应用程序与在任何标准 Kubernetes 环境中运行的应用程序完全兼容，无论这些应用程序是在本地数据中心还是在公有云中运行，都能兼容。这意味着，您可以轻松地将任何标准 Kubernetes 应用程序迁移到 Amazon EKS，而无需进行任何代码修改。

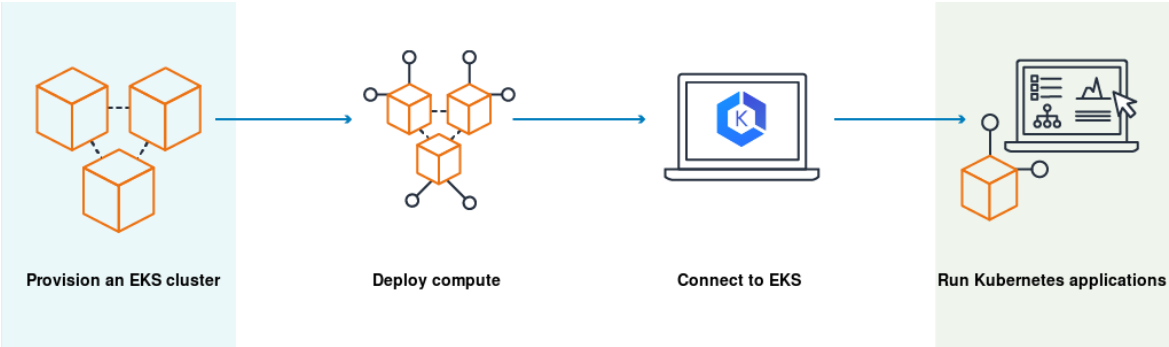
Amazon EKS 控制面板架构

Amazon EKS 为每个集群运行一个单一租户 Kubernetes 控制面板。控制面板基础设施不会在集群或 Amazon 账户之间共享。控制面板包含至少两个 API 服务器实例和三个 `etcd` 实例，这些实例在某个 Amazon Web Services 区域内的三个可用区之间运行。Amazon EKS：

- 主动监控控制面板实例上的负载，并自动缩放实例，确保高性能。
- 自动检测和替换运行状况不佳的控制面板实例，并根据需要跨 Amazon Web Services 区域内的可用区重新启动它们。
- 利用 Amazon Web Services 区域区域的架构以保持高可用性。因此，Amazon EKS 能够提供确保 API 服务器端点可用性的 SLA。

Amazon EKS 使用 Amazon VPC 网络策略将控制面板组件之间的流量限制到一个集群内。除非通过 Kubernetes RBAC 策略授权，否则，集群的控制面板组件无法查看或接收来自其他集群或其他 Amazon 账户的通信。这个安全且高度可用的配置让 Amazon EKS 成为生产工作负载的可靠的建议配置。

Amazon EKS 是如何工作的？



Amazon EKS 入门很简单：

1. 首先，在 Amazon Web Services Management Console 中或使用 Amazon CLI 或Amazon开发工具包之一创建一个 Amazon EKS 集群。
2. 启动托管或自行管理的 Amazon EC2 节点，或将您的工作负载部署到 Amazon Fargate。
3. 在集群准备就绪时，可以配置您常用的 Kubernetes 工具（如 `kubectl`），从而与您的集群通信。
4. 像在任何其他 Kubernetes 环境中一样，在 Amazon EKS 集群上部署和管理工作负载。您还可以使用 Amazon Web Services Management Console 查看与您的工作负载相关的信息。

要创建您的第一个集群及其关联的资源，请参阅 开始使用 Amazon EKS (p. 4)。要了解其他 Kubernetes 部署选项，请参阅 部署选项 (p. 2)。

定价

Amazon EKS 集群由一个控制面板以及您在其中运行 pods 的 Amazon EC2 或 Amazon Fargate 计算资源组成。有关控制面板定价的更多信息，请参阅 Amazon EKS 定价。Amazon EC2 和 Fargate 都提供：

- 按需实例 – 您只需按秒支付使用实例的费用，无需长期订阅或支付预付款。有关更多信息，请参阅 Amazon EC2 按需定价和 Amazon Fargate 定价。

部署选项

您可以将 Amazon EKS 用于以下任意部署选项或全部部署选项：

- Amazon EKS – Amazon Elastic Kubernetes Service (Amazon EKS) 是一项托管服务，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。有关更多信息，请参阅什么是 Amazon EKS？ (p. 1)。
- Amazon EKS on Amazon Outposts：您可以在 Amazon Outposts 上运行 Amazon EKS 节点。Amazon Outposts 在本地设施中支持本地 Amazon 服务、基础设施和操作模型。有关更多信息，请参阅Amazon Outposts 上的 Amazon EKS 节点 (p. 189)。
- Amazon EKS Anywhere – Amazon EKS Anywhere 是 Amazon EKS 的部署选项，使您能够在本地轻松创建和操作 Kubernetes 集群。Amazon EKS 和 Amazon EKS Anywhere 都是基于 Amazon EKS Distro 构建的。要了解有关 Amazon EKS Anywhere 及其与 Amazon EKS 的区别的更多信息，请参阅 Amazon EKS Anywhere 文档中的概览和将 Amazon EKS Anywhere 与 Amazon EKS 比较。
- Amazon EKS Distro – Amazon EKS Distro 是 Amazon EKS 在云中部署的相同开源 Kubernetes 软件和依赖项的发行版。Amazon EKS Distro 遵循与 Amazon EKS 相同的 Kubernetes 版本发布周期，并作为开源项目项目提供。要了解更多信息，请参阅 Amazon EKS Distro。您还可以在 GitHub 上查看和下载 Amazon EKS Distro 的源代码。

在选择用于 Kubernetes 集群的部署选项时，请考虑以下因素：

功能	Amazon EKS	Amazon Outposts 上的 Amazon EKS	Amazon EKS Anywhere	Amazon EKS Distro
Hardware (硬件)	Amazon 提供	Amazon 提供	由您提供	由您提供
部署位置	Amazon 云	您的数据中心	您的数据中心	您的数据中心
Kubernetes 控制面板位置	Amazon 云	Amazon 云	您的数据中心	您的数据中心

功能	Amazon EKS	Amazon Outposts 上的 Amazon EKS	Amazon EKS Anywhere	Amazon EKS Distro
Kubernetes 数据面板位置	Amazon 云	您的数据中心	您的数据中心	您的数据中心
支持	Amazon 支持	Amazon 支持	Amazon 支持	OSS 社群支持

常见问题

- 问：我能否在 Amazon 云中部署 Amazon EKS Anywhere？

  答：Amazon EKS Anywhere 并未设计成在 Amazon 云中运行。它未与适用于 Amazon 的 Kubernetes Cluster API Provider 集成。如果您计划在 Amazon Cloud 中部署 Kubernetes 集群，强烈建议您使用 Amazon EKS。
- 问：我能否在 Amazon Outposts 中部署 Amazon EKS Anywhere？

  答：Amazon EKS Anywhere 并未设计成在 Amazon Outposts 中运行。如果您计划在 Amazon Outposts 中部署 Kubernetes 集群，强烈建议您对 Amazon Outposts 使用 Amazon EKS。

开始使用 Amazon EKS

本用户指南的许多步骤都使用以下命令行工具：

- **kubectl** – 用于与 Kubernetes 集群一起使用的命令行工具。有关更多信息，请参阅安装或更新 kubectl (p. 4)。
- **eksctl** – 用于处理 EKS 集群的命令行工具，该工具可自动执行许多单独任务。有关更多信息，请参阅安装或更新 eksctl (p. 10)。
- Amazon CLI – 与 Amazon 服务一起使用的命令行工具，包括 Amazon EKS。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI。在安装 Amazon CLI 后，建议您还要对其进行配置。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的如何使用 aws configure 快速配置。

有两个入门指南可用于在 Amazon EKS 中使用节点创建新的 Kubernetes 集群：

- 开始使用 Amazon EKS – eksctl (p. 13) – 此入门指南可帮助您使用 eksctl（用于在 Amazon EKS 上创建和管理 Kubernetes 集群的简单命令行实用程序）来安装开始使用 Amazon EKS 所需的所有资源。在教程的最后，将为您提供一个运行中的 Amazon EKS 集群，您可向其部署应用程序。这是开始使用 Amazon EKS 的最快、最简单的方式。
- 开始使用 Amazon EKS – Amazon Web Services Management Console 和 Amazon CLI (p. 15) – 此入门指南可帮助您使用 Amazon Web Services Management Console 和 Amazon CLI 创建开始使用 Amazon EKS 所需的所有资源。在教程的最后，将为您提供一个运行中的 Amazon EKS 集群，您可向其部署应用程序。在本指南中，您将手动创建 Amazon EKS 集群所需的每个资源。这些程序可让您了解每个资源的创建方式以及资源之间的相互交互方式。

安装或更新 kubectl

Kubectl 是一个命令行工具，用于与 Kubernetes API 服务器进行通信。很多操作系统程序包管理器中都提供 kubectl 二进制文件。使用程序包管理器进行安装通常比手动下载并安装这一过程更简单。

本主题将帮助您下载并安装或更新设备上的 kubectl 二进制文件。该二进制文件与上游社区版本相同。该二进制文件不为 Amazon EKS 或 Amazon 所特有。

> Note
>
> 您必须使用与您的 Amazon EKS 集群控制层面不同的一个次要版本内的 kubectl 版本。例如，1.22 kubectl 客户端使用 Kubernetes 1.21、1.22 和 1.23 集群。

要安装或更新 kubectl

1. 确定您是否已将 kubectl 安装在设备上。

```
kubectl version | grep Client | cut -d : -f 5
```

如果您已将 kubectl 安装在设备路径中，示例输出如下所示。如果要使用更高版本更新当前已安装的版本，请完成下一步，确保将新版本安装在当前版本所在的位置。

```
"v1.22.6-eks-7d68063", GitCommit
```

如果您没有收到任何输出，则可能尚未安装 kubectl，或者未安装在设备路径中的位置。

2. 在 macOS、Linux 和 Windows 操作系统上安装或更新 kubectl。

   macOS

   要在 macOS 上安装或更新 kubectl

   1. 从 Amazon S3 为集群的 Kubernetes 版本下载二进制文件。
      - Kubernetes 1.23

        ```
        curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.23.7/2022-06-29/bin/darwin/amd64/kubectl
        ```

      - Kubernetes 1.22

        ```
        curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.22.6/2022-03-09/bin/darwin/amd64/kubectl
        ```

      - Kubernetes 1.21

        ```
        curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.21.2/2021-07-05/bin/darwin/amd64/kubectl
        ```

      - Kubernetes 1.20

        ```
        curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.20.4/2021-04-12/bin/darwin/amd64/kubectl
        ```

      - Kubernetes 1.19

        ```
        curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.19.6/2021-01-05/bin/darwin/amd64/kubectl
        ```

      - Kubernetes 1.18

        ```
        curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.18.9/2020-11-02/bin/darwin/amd64/kubectl
        ```

   2. （可选）使用二进制文件的 SHA-256 总和验证下载的二进制文件。
      a. 下载集群 Kubernetes 版本的 SHA-256 总和。
      - Kubernetes 1.23

        ```
        curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.23.7/2022-06-29/bin/darwin/amd64/kubectl.sha256
        ```

      - Kubernetes 1.22

        ```
        curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.22.6/2022-03-09/bin/darwin/amd64/kubectl.sha256
        ```

      - Kubernetes 1.21

        ```
        curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.21.2/2021-07-05/bin/darwin/amd64/kubectl.sha256
        ```

      - Kubernetes 1.20

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.20.4/2021-04-12/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.19

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.19.6/2021-01-05/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.18

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.18.9/2020-11-02/bin/darwin/amd64/kubectl.sha256
```

b. 检查下载的二进制文件的 SHA-256 总和。

```
openssl sha1 -sha256 kubectl
```

c. 将命令输出中生成的 SHA-256 总和与下载的 SHA-256 文件进行比较。这两者应该匹配。

3. 将执行权限应用于二进制文件。

```
chmod +x ./kubectl
```

4. 将二进制文件复制到您的 PATH 中的文件夹。如果您已经安装了某个版本的 kubectl，建议您创建一个 $HOME/bin/kubectl 并确保 $HOME/bin 先出现在您的 $PATH 中。

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:
$PATH
```

5. (可选) 将 $HOME/bin 路径添加到 shell 初始化文件，以便在打开 shell 时配置此路径。

```
echo 'export PATH=$PATH:$HOME/bin' >> ~/.bash_profile
```

6. 安装 kubectl 后，可以使用以下命令验证其版本：

```
kubectl version --short --client
```

Linux

要在 Linux 上安装或更新 **kubectl**

1. 使用适用于您的设备硬件平台的命令从 Amazon S3 为集群的 Kubernetes 版本下载 kubectl 二进制文件。每个版本的第一个链接对应的是 amd64，第二个链接对应的是 arm64。

- Kubernetes 1.23

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.23.7/2022-06-29/bin/linux/amd64/kubectl
```

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.23.7/2022-06-29/bin/linux/arm64/kubectl
```

- Kubernetes 1.22

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl
```

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.22.6/2022-03-09/bin/linux/arm64/kubectl
```

- Kubernetes `1.21`

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.21.2/2021-07-05/bin/linux/amd64/kubectl
```

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.21.2/2021-07-05/bin/linux/arm64/kubectl
```

- Kubernetes `1.20`

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.20.4/2021-04-12/bin/linux/amd64/kubectl
```

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.20.4/2021-04-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.19`

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.19.6/2021-01-05/bin/linux/amd64/kubectl
```

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.19.6/2021-01-05/bin/linux/arm64/kubectl
```

- Kubernetes `1.18`

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.18.9/2020-11-02/bin/linux/amd64/kubectl
```

```
curl -o kubectl https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.18.9/2020-11-02/bin/linux/arm64/kubectl
```

2. （可选）使用二进制文件的 `SHA-256` 总和验证下载的二进制文件。

   a. 使用适用于您的设备硬件平台的命令从 Amazon S3 为集群的 Kubernetes 版本下载 `SHA-256` 总和。每个版本的第一个链接对应的是 `amd64`，第二个链接对应的是 `arm64`。

   - Kubernetes `1.23`

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.23.7/2022-06-29/bin/linux/amd64/kubectl.sha256
```

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.23.7/2022-06-29/bin/linux/arm64/kubectl.sha256
```

   - Kubernetes `1.22`

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl.sha256
```

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.22.6/2022-03-09/bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.21`

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.21.2/2021-07-05/bin/linux/amd64/kubectl.sha256
```

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.21.2/2021-07-05/bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.20`

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.20.4/2021-04-12/bin/linux/amd64/kubectl.sha256
```

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.20.4/2021-04-12/bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.19`

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.19.6/2021-01-05/bin/linux/amd64/kubectl.sha256
```

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.19.6/2021-01-05/bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.18`

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.18.9/2020-11-02/bin/linux/amd64/kubectl.sha256
```

```
curl -o kubectl.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.18.9/2020-11-02/bin/linux/arm64/kubectl.sha256
```

b. 检查下载的二进制文件的 SHA-256 总和。

```
openssl sha1 -sha256 kubectl
```

c. 将命令输出中生成的 SHA-256 总和与下载的 SHA-256 文件进行比较。这两者应该匹配。

3. 将执行权限应用于二进制文件。

```
chmod +x ./kubectl
```

4. 将二进制文件复制到您的 PATH 中的文件夹。如果您已经安装了某个版本的 kubectl，建议您创建一个 $HOME/bin/kubectl 并确保 $HOME/bin 先出现在您的 $PATH 中。

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/
bin
```

5. (可选) 将 $HOME/bin 路径添加到 shell 初始化文件，以便在打开 shell 时配置此路径。

   Note

   这一步假设您使用 Bash Shell；如果使用其他 Shell，请将命令更改为使用您的特定 Shell 的初始化文件。

```
echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
```

6. 安装 kubectl 后，可以使用以下命令验证其版本：

```
kubectl version --short --client
```

Windows

要在 Windows 上安装或更新 **kubectl**

1. 打开 PowerShell 终端。
2. 从 Amazon S3 为集群的 Kubernetes 版本下载 kubectl 二进制文件。
   - Kubernetes 1.23

```
curl -o kubectl.exe https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.23.7/2022-06-29/bin/windows/amd64/kubectl.exe
```

   - Kubernetes 1.22

```
curl -o kubectl.exe https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.22.6/2022-03-09/bin/windows/amd64/kubectl.exe
```

   - Kubernetes 1.21

```
curl -o kubectl.exe https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.21.2/2021-07-05/bin/windows/amd64/kubectl.exe
```

   - Kubernetes 1.20

```
curl -o kubectl.exe https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.20.4/2021-04-12/bin/windows/amd64/kubectl.exe
```

   - Kubernetes 1.19

```
curl -o kubectl.exe https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.19.6/2021-01-05/bin/windows/amd64/kubectl.exe
```

   - Kubernetes 1.18

```
curl -o kubectl.exe https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.18.9/2020-11-02/bin/windows/amd64/kubectl.exe
```

3. （可选）使用二进制文件的 SHA-256 总和验证下载的二进制文件。
   a. 为 Windows 的集群的 Kubernetes 版本下载 SHA-256 总和。
      - Kubernetes 1.23

```
curl -o kubectl.exe.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.23.7/2022-06-29/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.22

```
curl -o kubectl.exe.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.22.6/2022-03-09/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.21

```
curl -o kubectl.exe.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.21.2/2021-07-05/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.20

```
curl -o kubectl.exe.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.20.4/2021-04-12/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.19

```
curl -o kubectl.exe.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.19.6/2021-01-05/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.18

```
curl -o kubectl.exe.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.18.9/2020-11-02/bin/windows/amd64/kubectl.exe.sha256
```

b. 检查下载的二进制文件的 SHA-256 总和。

```
Get-FileHash kubectl.exe
```

c. 将命令输出中生成的 SHA-256 总和与下载的 SHA-256 文件进行比较。二者应匹配，不过 PowerShell 输出是大写。

4. 将二进制文件复制到您的 PATH 中的文件夹。如果 PATH 中有现有目录可用于命令行实用程序，请将二进制文件复制到该目录。否则，请完成以下步骤。
   a. 为命令行二进制文件创建一个新目录，例如 C:\bin。
   b. 将 kubectl.exe 二进制文件复制到新目录。
   c. 编辑用户或系统 PATH 环境变量，将新目录添加到 PATH 中。
   d. 关闭 PowerShell 终端并打开一个新的终端来选取新的 PATH 变量。
5. 安装 kubectl 后，可以使用以下命令验证其版本：

```
kubectl version --short --client
```

安装或更新 eksctl

本主题帮助您安装或更新 eksctl，一种用于创建和管理 Amazon EKS 上 Kubernetes 集群的简单命令行工具。Eksctl 提供使用节点为 Amazon EKS 创建新集群的最快、最简单的方式。有关更多信息以及查看官方文档，请参阅 https://eksctl.io/。

先决条件

您的设备或 Amazon CloudShell 上安装了 kubectl 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 1.22，则可以将 kubectl 版本 1.21、1.22 或 1.23 用于它。要安装或升级 kubectl，请参阅 安装或更新 kubectl (p. 4)。

要安装或更新 eksctl

1. 确定您是否已将 eksctl 安装在设备上。

```
eksctl version
```

如果您已将 eksctl 安装在设备路径中，示例输出如下所示。如果要使用更高版本更新当前已安装的版本，请完成下一步，确保将新版本安装在当前版本所在的位置。

```
0.110.0
```

如果您没有收到任何输出，则可能尚未安装 eksctl，或者未安装在设备路径中的位置。

2. 您可以在 macOS、Linux 或 Windows 上安装 eksctl

macOS

要在 macOS 上安装或更新 eksctl

开始使用 Amazon EKS 和 macOS 的最简单方式是使用 Homebrew 安装 eksctl，可以按照以下说明安装此开源工具。eksctl Homebrew 配方安装 Amazon EKS 所需的 eksctl 和任何其他依赖项，如 kubectl。配方还会安装 aws-iam-authenticator (p. 369)，如果您没有安装 Amazon CLI 版本 1.16.156 或更高版本，则这是必需的。

1. 如果您尚未在 macOS 上安装 Homebrew，请使用以下命令安装它。

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
master/install.sh)"
```

2. 安装 Weaveworks Homebrew tap。

```
brew tap weaveworks/tap
```

3. 安装或升级 eksctl。
   - 使用以下命令安装 eksctl：

```
brew install weaveworks/tap/eksctl
```

   - 如果已安装 eksctl，请运行以下命令进行升级：

```
brew upgrade eksctl && brew link --overwrite eksctl
```

4. 使用以下命令测试您的安装是否成功。

```
eksctl version
```

Note

GitTag 版本应至少为 0.110.0。否则，请检查您的终端输出是否有任何安装或升级错误，或从以下位置手动下载该版本的归档：https://github.com/weaveworks/eksctl/releases/download/v0.110.0/eksctl_Darwin_amd64.tar.gz，提取并运行 eksctl。

Linux

要在 Linux 上安装或更新 **eksctl**

1. 使用以下命令下载并提取最新版本的 eksctl。

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

2. 将提取的二进制文件移至 /usr/local/bin。

```
sudo mv /tmp/eksctl /usr/local/bin
```

3. 使用以下命令测试您的安装是否成功。

```
eksctl version
```

> **Note**
>
> GitTag 版本应至少为 0.110.0。如果不是，请查看终端输出中的任何安装或升级错误，或将步骤 1 中的地址替换为 https://github.com/weaveworks/eksctl/releases/download/v0.110.0/eksctl_Linux_amd64.tar.gz 并再次完成步骤 1-3。

Windows

要在 Windows 上安装或更新 **eksctl**

1. 如果您尚未在 Windows 系统上安装 Chocolatey，请参阅安装 Chocolatey。
2. 安装或升级 eksctl。
   - 使用以下命令安装二进制文件：

```
choco install -y eksctl
```

   - 如果已安装，请运行以下命令进行升级：

```
choco upgrade -y eksctl
```

3. 使用以下命令测试您的安装是否成功。

```
eksctl version
```

> **Note**
>
> GitTag 版本应至少为 0.110.0。否则，请检查您的终端输出是否有任何安装或升级错误，或从以下位置手动下载该版本的存档：https://github.com/weaveworks/eksctl/releases/download/v0.110.0/eksctl_Windows_amd64.zip，提取并运行 eksctl。

开始使用 Amazon EKS – eksctl

此指南可帮助您使用 eksctl（用于在 Amazon EKS 上创建和管理 Kubernetes 集群的简单命令行实用程序）来安装开始使用 Amazon Elastic Kubernetes Service (Amazon EKS) 所需的所有资源。在本教程的最后，将为您提供一个运行中的 Amazon EKS 集群，您可向其部署应用程序。

本指南中的程序可自动为您创建在您使用 Amazon Web Services Management Console 创建集群时必须手动创建的一些资源。如果您希望手动创建大多数资源并更好地了解它们之间的交互方式，请使用 Amazon Web Services Management Console 创建集群和执行计算。有关更多信息，请参阅开始使用 Amazon EKS – Amazon Web Services Management Console 和 Amazon CLI (p. 15)。

先决条件

在开始使用本教程之前，您必须安装并配置创建和管理 Amazon EKS 集群所需的以下工具和资源。

- **kubectl** – 用于与 Kubernetes 集群一起使用的命令行工具。本指南要求您使用 1.23 版或更高版本。有关更多信息，请参阅安装或更新 kubectl (p. 4)。
- **eksctl** – 用于处理 EKS 集群的命令行工具，该工具可自动执行许多单独任务。本指南要求您使用 0.110.0 版或更高版本。有关更多信息，请参阅安装或更新 eksctl (p. 10)。
- 所需的 IAM 权限 – 您正在使用的 IAM 安全主体必须具有使用 Amazon EKS IAM 角色和服务相关角色的权限、Amazon CloudFormation 以及 VPC 和相关资源。有关更多信息，请参阅 IAM 用户指南中的用于 Amazon Elastic Container Service for Kubernetes 的操作、资源和条件键和使用服务相关角色。您必须以同一用户身份完成本指南中的所有步骤。

第 1 步：创建 Amazon EKS 集群和节点

> **Important**
>
> 为了尽可能简单快速地入门，本主题包括创建具有原定设置的集群和节点的步骤。在创建用于生产使用的集群和节点之前，我们建议您熟悉所有设置，并使用符合您要求的设置部署集群和节点。有关更多信息，请参阅 创建 Amazon EKS 集群 (p. 21) 和 Amazon EKS 节点 (p. 84)。一些设置仅在创建集群和节点时可以启用。

使用以下命令创建 Amazon EKS 集群。您可以将 *my-cluster* 替换为自己的值。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。将 *region-code* 替换为 Amazon EKS 支持的任一 Amazon Web Services 区域。有关 Amazon Web Services 区域列表，请参阅 Amazon 一般参考中的 Amazon EKS 端点和配额。

Managed nodes – Linux

```
eksctl create cluster --name my-cluster --region region-code
```

创建集群需要几分钟时间。在创建过程中，您将看到几行输出。输出的最后一行类似于以下示例行。

```
...
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

eksctl 在 ~/.kube 中创建了一个 kubectl config 文件，或在计算机上的 ~/.kube 中的现有 config 文件添加了新集群的配置。

完成集群创建后，在 Amazon CloudFormation 控制台 https://console.aws.amazon.com/cloudformation 中查看名为 eksctl-*my-cluster*-cluster 的 Amazon CloudFormation 堆栈，查看创建的所有资源。

第 2 步：查看 Kubernetes 资源

1. 查看您的集群节点。

```
kubectl get nodes -o wide
```

输出示例如下。

Managed nodes – Linux

```
NAME                                              STATUS   ROLES    AGE    VERSION
           INTERNAL-IP      EXTERNAL-IP     OS-IMAGE          KERNEL-VERSION
          CONTAINER-RUNTIME
ip-192-168-12-49.region-code.compute.internal   Ready    <none>   6m7s
  v1.23.7-eks-d1db3c   192.168.12.49    52.35.116.65    Amazon Linux 2
  5.4.156-83.273.amzn2.x86_64   docker://20.10.7
ip-192-168-72-129.region-code.compute.internal  Ready    <none>   6m4s
  v1.23.7-eks-d1db3c   192.168.72.129   44.242.140.21   Amazon Linux 2
  5.4.156-83.273.amzn2.x86_64   docker://20.10.7
```

如需详细了解输出中显示的内容，请参阅 查看 Kubernetes 资源 (p. 458)。

2. 查看在集群上运行的工作负载。

```
kubectl get pods -A -o wide
```

输出示例如下。

Managed nodes – Linux

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
              NODE                                                 NOMINATED NODE
 READINESS GATES
kube-system   aws-node-6ctpm                1/1     Running   0          7m43s
  192.168.72.129   ip-192-168-72-129.region-code.compute.internal   <none>
  <none>
kube-system   aws-node-cbntg                1/1     Running   0          7m46s
  192.168.12.49    ip-192-168-12-49.region-code.compute.internal    <none>
  <none>
kube-system   coredns-559b5db75d-26t47      1/1     Running   0          14m
  192.168.78.81    ip-192-168-72-129.region-code.compute.internal   <none>
  <none>
kube-system   coredns-559b5db75d-9rvnk      1/1     Running   0          14m
  192.168.29.248   ip-192-168-12-49.region-code.compute.internal    <none>
  <none>
kube-system   kube-proxy-l8pbd              1/1     Running   0          7m46s
  192.168.12.49    ip-192-168-12-49.region-code.compute.internal    <none>
  <none>
kube-system   kube-proxy-zh85h              1/1     Running   0          7m43s
  192.168.72.129   ip-192-168-72-129.region-code.compute.internal   <none>
  <none>
```

如需详细了解输出中显示的内容，请参阅 查看 Kubernetes 资源 (p. 458)。

第 3 步：删除集群和节点

在使用完成针对本教程而创建的集群和节点后，应使用下面的命令删除这些集群和节点，从而将它们清除。如果要在清除该集群前对其执行更多操作，请参阅 后续步骤 (p. 15)。

```
eksctl delete cluster --name my-cluster --region region-code
```

后续步骤

以下文档主题可帮助您扩展集群的此功能。

- 将示例应用程序 (p. 315)部署到您的集群。
- 创建集群的 IAM 实体（用户或角色）是唯一可以使用 `kubectl` 或 Amazon Web Services Management Console 调用 Kubernetes API 服务器的 IAM 用户。如果您希望其他 IAM 用户或角色拥有访问您集群的权限，您需要添加它们。有关更多信息，请参阅 让 IAM 用户和角色有权访问您的集群 (p. 355) 和 所需权限 (p. 458)。
- 在部署集群用于生产用途前，我们建议您熟悉集群 (p. 21)和节点 (p. 84)的所有设置。创建集群时必须进行一些设置（例如启用 SSH 访问 Amazon EC2 节点）。
- 为了提高集群的安全性，将 Amazon VPC Container Networking Interface 插件配置为使用 IAM 角色作为服务账户 (p. 243).

开始使用 Amazon EKS – Amazon Web Services Management Console 和 Amazon CLI

本指南帮助您创建通过 Amazon Web Services Management Console 和 Amazon CLI 开始使用 Amazon Elastic Kubernetes Service (Amazon EKS) 所需的所有资源。在本指南中，您将手动创建每个资源。在本教程的最后，将为您提供一个运行中的 Amazon EKS 集群，您可向其部署应用程序。

本指南中的步骤可让您完全了解每个资源的创建方式以及资源之间如何相互交互。如果您希望系统自动为您创建大部分资源，请使用 `eksctl` CLI 创建集群和节点。有关更多信息，请参阅开始使用 Amazon EKS – `eksctl` (p. 13)。

先决条件

在开始使用本教程之前，您必须安装并配置创建和管理 Amazon EKS 集群所需的以下工具和资源。

- Amazon CLI – 与Amazon服务（包括 Amazon EKS）一起使用的命令行工具。本指南要求您使用 `2.7.21` 版或更高版本，或者 `1.25.46` 版或更高版本。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI。在安装 Amazon CLI 后，建议您还要对其进行配置。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的如何使用 `aws configure` 快速配置。
- **`kubectl`** – 用于与 Kubernetes 集群一起使用的命令行工具。本指南要求您使用 `1.23` 版或更高版本。有关更多信息，请参阅安装或更新 `kubectl` (p. 4)。
- 所需的 IAM 权限 – 您正在使用的 IAM 安全主体必须具有使用 Amazon EKS IAM 角色和服务相关角色的权限、Amazon CloudFormation 以及 VPC 和相关资源。有关更多信息，请参阅 IAM 用户指南中的用于 Amazon Elastic Kubernetes Service 的操作、资源和条件密钥和使用服务相关角色。您必须以同一用户身份完成本指南中的所有步骤。

第 1 步：创建 Amazon EKS 集群

Important

为了尽可能简单快速地入门，本主题包括创建具有原定设置的集群的步骤。创建用于生产用途的集群前，我们建议您熟悉所有设置，并使用符合您要求的设置部署集群。有关更多信息，请参阅创建 Amazon EKS 集群 (p. 21)。一些设置仅在创建集群时可以启用。

创建集群的步骤

1. 创建具有公有和私有子网且符合 Amazon EKS 要求的 Amazon VPC。将 *region-code* 替换为 Amazon EKS 支持的任何 Amazon Web Services 区域。有关 Amazon Web Services 区域 列表，请参阅 Amazon 一般参考指南中的 Amazon EKS 端点和配额。您可以将 *my-eks-vpc-stack* 替换为您选择的任何名称。

```
aws cloudformation create-stack \
  --region region-code \
  --stack-name my-eks-vpc-stack \
  --template-url https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

   Tip

   如需查看上一个命令创建的所有资源的列表，请前往 https://console.aws.amazon.com/cloudformation 打开 Amazon CloudFormation 控制台。选择 *my-eks-vpc-stack* 堆栈，然后选择 Resources（资源）选项卡。

2. 创建集群 IAM 角色并向其附加所需的 Amazon EKS IAM 托管策略。Amazon EKS 托管的 Kubernetes 集群会代表您调用其他 Amazon 服务，以管理您用于该服务的资源。

   a. 将以下内容复制到名为 *eks-cluster-role-trust-policy.json* 的文件中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   b. 创建角色。

```
aws iam create-role \
  --role-name myAmazonEKSClusterRole \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

   c. 将所需的 Amazon EKS 托管 IAM policy 附加到角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name myAmazonEKSClusterRole
```

3. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。

确保控制台右上角显示的 Amazon Web Services 区域 是您要在其中创建集群的Amazon Web Services 区域。如果不是，请选择 Amazon Web Services 区域名称旁边的下拉菜单，然后选择要使用的 Amazon Web Services 区域。

4. 请选择 Add cluster（添加集群），然后选择 Create（创建）。如果没有看到此选项，请先选择左侧导航面板中的 Clusters（集群）。
5. 在 Configure cluster（配置集群）页面上，请执行以下操作：
   a. 输入集群的 Name（名称），例如 **my-cluster**。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。
   b. 对于 Cluster Service Role（集群服务角色），请选择 *myAmazonEKSClusterRole*。
   c. 其余设置保留为默认值，然后选择 Next（下一步）。
6. 在 Specify networking（指定联网）页面中，请执行以下操作：
   a. 请从 VPC 下拉列表中选择在上一步创建的 VPC ID。此 ID 形如 *vpc-00x0000x000x0x000* | *my-eks-vpc-stack-VPC*。
   b. 其余设置保留为默认值，然后选择 Next（下一步）。
7. 请在 Configure logging（配置日志记录）页面上，选择 Next（下一步）。
8. 请在 Review and create（审核和创建）页面上，选择 Create（创建）。

集群名称右侧的集群状态会保持为 Creating（正在创建）几分钟，直至集群调配过程完成。在该状态变为 Active（有效）之前，请勿继续执行下一步。

Note

您可能会收到一个错误，指示请求中的可用区之一没有足够容量来创建 Amazon EKS 集群。如果发生这种情况，错误输出将包含可支持新集群的可用区。再次尝试使用至少两个位于您账户中支持的可用区的子网创建集群。有关更多信息，请参阅容量不足 (p. 478)。

第 2 步：将计算机配置为与您的集群通信

在本部分中，您将为集群创建一个 `kubeconfig` 文件。此文件中的设置会启用 `kubectl` CLI 与您的集群进行通信。

将计算机配置为与您的集群通信的步骤

1. 为集群创建或更新 `kubeconfig` 文件。将 *region-code* 替换为您要在其中创建集群的 Amazon Web Services 区域。将 *my-cluster* 替换为您的集群名称。

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

预设情况下，`config` 文件创建在 `~/.kube` 中或者新集群的配置已添加到 `~/.kube` 的现有 `config` 文件中。

2. 测试配置。

```
kubectl get svc
```

Note

如果您收到任何授权或资源类型错误，请参阅故障排除部分中的未经授权或访问被拒绝 (kubectl) (p. 478)。

输出示例如下。

```
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

第 3 步：创建节点

Important

为了尽可能简单快速地入门，本主题包括创建具有原定设置的节点的步骤。在创建用于生产用途的节点之前，我们建议您熟悉所有设置，并使用符合您要求的设置部署节点。有关更多信息，请参阅Amazon EKS 节点 (p. 84)。一些设置仅在创建节点时可以启用。

您可以使用下列节点类型创建集群。要了解有关节点类型的更多信息，请参阅 Amazon EKS 节点 (p. 84)。

- 托管节点 – Linux – 此节点类型允许您在 Amazon EC2 实例上运行 Amazon Linux 应用程序。

Managed nodes – Linux

创建托管节点组，指定您在前面的步骤中创建的子网和节点 IAM 角色。

要创建 Amazon EC2 Linux 托管节点组

1. 创建节点 IAM 角色并向其附加所需的 Amazon EKS IAM 托管策略。Amazon EKS 节点 `kubelet` 守护进程代表您调用 Amazon API。节点通过 IAM 实例配置文件和关联的策略获得这些 API 调用的权限。

   a. 将以下内容复制到名为 *node-role-trust-policy.json* 的文件中。

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "ec2.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   b. 创建节点 IAM 角色。

   ```
   aws iam create-role \
     --role-name myAmazonEKSNodeRole \
     --assume-role-policy-document file://"node-role-trust-policy.json"
   ```

   c. 将所需的托管 IAM policy 附加到角色。

   ```
   aws iam attach-role-policy \
     --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
     --role-name myAmazonEKSNodeRole
   aws iam attach-role-policy \
     --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
     --role-name myAmazonEKSNodeRole
   aws iam attach-role-policy \
     --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
   ```

```
--role-name myAmazonEKSNodeRole
```

2. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
3. 选择您在 第 1 步：创建 Amazon EKS 集群 (p. 16) 中创建的集群的名称，例如 *my-cluster*。
4. 在 ***my-cluster*** 页面上，请执行以下操作：
   a. 请选择 Compute（计算）选项卡。
   b. 请选择 Add Node Group（添加节点组）。
5. 在 Configure Node Group（配置节点组）页面上，请执行以下操作：
   a. 对于 Name（名称），输入托管节点组的唯一名称，例如 ***my-nodegroup***。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。
   b. 对于 Node IAM role name（节点 IAM 角色名称），请选择您在上一步中创建的 *myAmazonEKSNodeRole* 角色。我们建议让每个节点组使用各自的唯一 IAM 角色。
   c. 选择 Next (下一步)。
6. 请在 Set compute and scaling configuration（设置计算和扩缩配置）页面上，接受默认值并选择 Next（下一步）。
7. 请在 Specify networking（指定联网）页面，接受默认值，选择 Next（下一步）。
8. 在 Review and create (审核并创建) 页面上，审核托管节点组配置并选择 Create (创建)。
9. 几分钟后，Node Group configuration（节点组配置）部分中的 Status（状态）将从 Creating（正在创建）变为 Active（有效）。在该状态变为 Active（有效）之前，请勿继续执行下一步。

第 4 步：查看资源

您可以查看节点和 Kubernetes 工作负载。

查看节点和工作负载

1. 在左侧导航窗格中，选择集群。在 Clusters（集群）列表中，选择您创建的集群名称，例如 *my-cluster*。
2. 在 ***my-cluster*** 页面上，选择以下选项：
   a. Compute（计算）选项卡– 您可以看到为集群部署的节点列表。您可以选择节点的名称以查看有关该节点的详细信息。
   b. Resources（资源）选项卡 – 您将看到默认情况下部署到 Amazon EKS 集群的所有 Kubernetes 资源。在控制台中选择任何资源类型以了解有关它的更多信息。

第 5 步：删除资源

在使用完成针对本教程而创建的集群和节点后，应删除这些资源。如果要在删除资源前对此集群执行更多操作，请参阅 后续步骤 (p. 20)。

删除您按照本指南中的说明创建的资源

1. 请删除您创建的任何节点组。
   a. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
   b. 在左侧导航窗格中，选择集群。请在集群列表中，选择 *my-cluster*。
   c. 请选择 Compute（计算）选项卡。
   d. 选择 *my-nodegroup* 节点组，然后选择 Delete（删除）。输入 ***my-nodegroup***，然后选择删除。
   e. 在节点组删除后再继续操作。

2. 请删除集群。

   a. 在左侧导航窗格中，选择集群。请在集群列表中，选择 *my-cluster*。
   b. 选择 Delete Cluster (删除集群)。
   c. 输入 ***my-cluster***，然后选择 Delete（删除）。请在集群删除后再继续操作。
3. 删除创建的 VPC Amazon CloudFormation 堆栈。

   a. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
   b. 选择 *my-eks-vpc-stack* 堆栈，然后选择 Delete（删除）。
   c. 请在 Delete ***my-eks-vpc-stack***（删除 my-eks-vpc-stack）确认对话框中，选择 Delete stack（删除堆栈）。
4. 删除您创建的 IAM 角色。

   a. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
   b. 在左侧导航窗格中，选择 Roles（角色）。
   c. 请从列表中选择您创建的每个角色（***myAmazonEKSClusterRole***，以及 *myAmazonEKSNodeRole*）。请选择 Delete（删除），输入请求的确认文本，然后选择 Delete（删除）。

后续步骤

以下文档主题可帮助您扩展集群的此功能。

- 创建集群的 IAM 实体（用户或角色）是唯一可以使用 `kubectl` 或 Amazon Web Services Management Console 调用 Kubernetes API 服务器的 IAM 用户。如果您希望其他 IAM 用户或角色拥有访问您集群的权限，您需要添加它们。有关更多信息，请参阅 让 IAM 用户和角色有权访问您的集群 (p. 355) 和 所需权限 (p. 458)。
- 将示例应用程序 (p. 315)部署到您的集群。
- 在部署集群用于生产用途前，我们建议您熟悉集群 (p. 21)和节点 (p. 84)的所有设置。创建集群时必须进行一些设置（例如启用 SSH 访问 Amazon EC2 节点）。
- 为了提高集群的安全性，将 Amazon VPC Container Networking Interface 插件配置为使用 IAM 角色作为服务账户 (p. 243).

Amazon EKS 集群

一个 Amazon EKS 集群包含两个主要组件：

- Amazon EKS 控制层面
- 向控制层面注册的 Amazon EKS 节点

Amazon EKS 控制面板由运行 Kubernetes 软件（如 `etcd`）的控制面板节点和 Kubernetes API 服务器组成。控制面板在由 Amazon 管理的账户中运行，并且 Kubernetes API 将通过与集群关联的 Amazon EKS 端点公开。每个 Amazon EKS 集群控制层面都是单一租户而且是唯一的，它们在其自己的一组 Amazon EC2 实例上运行。

`etcd` 节点和关联 Amazon EBS 卷存储的所有数据均使用 Amazon KMS 加密。集群控制面板已跨多个可用区进行预置，其前面配置了一个 Elastic Load Balancing Network Load Balancer。Amazon EKS 还在 VPC 子网中预置了弹性网络接口，以便在控制面板实例与节点之间建立连接（例如，为了支持 `kubectl exec logs proxy` 数据流）。

> **Important**
>
> 在 Amazon EKS 环境中，根据上游指导，`etcd` 存储限制为 8GB。您可以监控当前数据库大小的 `etcd_db_total_size_in_bytes` 指标。

Amazon EKS 节点在Amazon账户中运行并通过 API 服务器端点和为集群创建的证书文件连接到集群的控制层面。

> **Note**
>
> - 您可以了解 Amazon EKS 的不同组件在 Amazon EKS 联网 (p. 226) 中的工作方式。
> - 有关互联的集群，请参阅 Amazon EKS Connector (p. 495)。

创建 Amazon EKS 集群

本主题概述了可用选项，并介绍了创建 Amazon EKS 集群时需要考虑的内容。如果您是首次创建 Amazon EKS 集群，我们建议您按照我们的 开始使用 Amazon EKS (p. 4) 指南之一操作。这些指南可帮助您创建一个简单的默认集群，而无需扩展到所有可用选项。

先决条件

- 满足 Amazon EKS 要求 (p. 226)的现有 VPC 和子网。在部署集群用于生产用途前，我们建议您彻底了解 VPC 和子网要求。如果您没有 VPC 和子网，则可以使用 Amazon EKS 提供的 Amazon CloudFormation 模板 (p. 228)创建它们。
- 您的设备或 Amazon CloudShell 上安装了 `kubectl` 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 `1.22`，则可以将 `kubectl` 版本 `1.21`、`1.22` 或 `1.23` 用于它。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl` (p. 4)。
- 您的设备或 Amazon CloudShell 上安装并配置了 `2.7.21` 版或更高版本，或 `1.25.46` 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 `aws configure` 进行快速配置。

- 具有 `create` 和 `describe` Amazon EKS 集群权限的 IAM 用户或角色。有关更多信息，请参阅 the section called “创建 Kubernetes 集群” (p. 417) 和 the section called “列出或描述所有集群” (p. 418)。

创建 Amazon EKS 集群后，创建集群的 IAM 实体（用户或角色）将作为管理员永久添加到 Kubernetes RBAC 授权表中。该实体具有 `system:masters` 权限。此实体的身份在您的集群配置中不可见。因此，重要的是要注意创建集群的实体并确保永远不会删除它。最初，仅创建服务器的 IAM 实体可以使用 `kubectl` 调用 Kubernetes API 服务器。如果使用控制台创建集群，则必须确保在集群上运行 `kubectl` 命令时，相同的 IAM 凭证位于 Amazon 开发工具包凭证链中。创建集群后，您可以向其他 IAM 实体授予对集群的访问权限。

创建 Amazon EKS 集群

1. 如果您已经拥有集群 IAM 角色，或者您将使用 `eksctl` 创建集群，则可以跳过此步骤。默认情况下，`eksctl` 会为您创建角色。

   #### 创建 Amazon EKS 集群 IAM 角色

   1. 运行以下命令以创建 IAM 信任策略 JSON 文件。

      ```
      cat >eks-cluster-role-trust-policy.json <<EOF
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "eks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      }
      EOF
      ```

   2. 创建 Amazon EKS 集群 IAM 角色。如有必要，使用您在上一步中将文件写入到的计算机上的路径为 *eks-cluster-role-trust-policy.json* 添加前言。该命令将您在上一步中创建的信任策略与角色关联。要创建 IAM 角色，必须为正在创建角色的 IAM 实体（用户或角色）分配以下 IAM 操作（权限）：`iam:CreateRole`。

      ```
      aws iam create-role --role-name myAmazonEKSClusterRole --assume-role-policy-document
       file://"eks-cluster-role-trust-policy.json"
      ```

   3. 将名为 `AmazonEKSClusterPolicy` 的 Amazon EKS 托管 IAM policy 附加到角色。要将 IAM policy 附加到 IAM 实体（用户或角色），必须为附加该策略的 IAM 实体分配以下 IAM 操作（权限）之一：`iam:AttachUserPolicy` 或 `iam:AttachRolePolicy`。

      ```
      aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
      AmazonEKSClusterPolicy --role-name myAmazonEKSClusterRole
      ```

2. 创建 Amazon EKS 集群。

   您可以使用 `eksctl`、Amazon Web Services Management Console 或 Amazon CLI 创建集群。

   eksctl

   先决条件

   您的设备或 Amazon CloudShell 上安装了 `0.110.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 安装或更新 `eksctl` (p. 10)。

创建集群的步骤

在您的默认 Amazon Web Services 区域 中，使用 Amazon EKS 最新 Kubernetes 版本创建 Amazon EKS IPv4 集群。在运行命令之前，进行以下替换：

- 将 *region-code* 替换为您要在其中创建集群的 Amazon Web Services 区域。
- 将 *my-cluster* 替换为您的集群名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。对于您在其中创建集群的 Amazon Web Services 区域 和 Amazon Web Services 账户，该名称必须在其内具有唯一性。
- 将 *1.23* 替换为任何 Amazon EKS 支持的版本 (p. 59)。
- 更改 vpc-private-subnets 的值以满足您的要求。您还可以添加其他 ID。您必须指定至少两个子网 ID。如果您宁愿指定公有子网，您可以将 --vpc-private-subnets 更改为 --vpc-public-subnets。公有子网有一个与互联网网关的路由相关联的路由，但私有子网没有关联的路由表。我们建议尽可能使用私有子网。

  您选择的子网必须符合 Amazon EKS 子网要求 (p. 227)。在选择子网之前，我们建议您熟悉所有的 Amazon EKS VPC 以及子网要求和考虑因素 (p. 226)。集群创建后，您无法更改要使用的子网。

```
eksctl create cluster --name my-cluster --region region-code --version 1.23 --vpc-private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
```

集群预配置需要几分钟时间。在创建集群时，将显示几行输出。输出的最后一行类似于以下示例行。

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> 要查看在使用 eksctl 创建集群时可指定的大多数选项，请使用 **eksctl create cluster --help** 命令。要查看所有可用的选项，请使用 config 文件。有关更多信息，请参阅 eksctl 文档中的使用配置文件和配置文件架构。您可以在 GitHub 上查找配置文件示例。

可选设置

以下是可选设置，如果需要，必须将这些设置添加到上一个命令中。您只能在创建集群时启用这些选项，而不能在创建集群后启用。如果您需要指定这些选项，则必须使用 eksctl 配置文件创建集群，然后指定设置，而不是使用上一个命令。

- 如果您想指定 Amazon EKS 分配给它创建的网络接口的一个或多个安全组，请指定 securityGroup 选项。

  无论您是否选择任何安全组，Amazon EKS 都会创建一个安全组，以实现集群和 VPC 之间的通信。Amazon EKS 将此安全组以及您选择的任何安全组与它创建的网络接口关联起来。有关 Amazon EKS 创建的集群安全组的更多信息，请参阅 the section called “安全组要求” (p. 231)。您可以修改 Amazon EKS 创建的集群安全组中的规则。如果您选择添加自己的安全组，则无法更改集群创建后选择的安全组。
- 如果您想指定 Kubernetes 从中分配服务 ID 地址的 IPv4 无类别域间路由块，请指定 serviceIPv4CIDR 选项。

  指定自己的范围有助于防止 Kubernetes 服务与对等或连接到您的 VPC 的其他网络之间的冲突。使用 CIDR 表示法输入范围。例如：10.2.0.0/16。

此 CIDR 块必须满足以下要求：

- 处于以下范围之一：10.0.0.0/8、172.16.0.0/12 或 192.168.0.0/16。
- 具有最小大小 /24 和最大大小 /12。
- 与您的 Amazon EKS 资源的 VPC 范围不重叠。

您只能在使用 IPv4 地址系列和创建集群时指定此选项。如果您没有指定此选项，Kubernetes 从 10.100.0.0/16 或 172.20.0.0/16 CIDR 块分配服务 IP 地址。

Amazon Web Services Management Console

创建集群的步骤

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 请选择 Add cluster（添加集群），然后选择 Create（创建）。
3. 在 Configure cluster（配置集群）页面上，输入以下字段：
   - Name（名称）– 集群的名称。它必须在您的 Amazon Web Services 账户 中具有唯一性。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。对于您在其中创建集群的 Amazon Web Services 区域 和 Amazon Web Services 账户，该名称必须在其内具有唯一性。
   - Kubernetes 版本 – 要用于集群的 Kubernetes 版本。
   - Cluster Service Role（集群服务角色）– 选择您创建的 Amazon EKS 集群 IAM 角色以允许 Kubernetes 控制面板来代表您管理 Amazon 资源。
   - Secrets encryption（密钥加密）–（可选）选择此选项以使用 KMS 密钥启用 Kubernetes 密钥的密钥加密。您也可以在创建集群后启用此功能。在启用此功能之前，请确保您熟悉 the section called “启用密钥加密” (p. 42) 中的信息。
   - Tags（标签）–（可选）向集群添加任何标签。有关更多信息，请参阅为您的 Amazon EKS 资源添加标签 (p. 385)。
4. 选择 Next（下一步）。
5. 在 Specify networking (指定联网) 页面上，为以下字段选择值：
   - VPC – 选择符合 Amazon EKS VPC 要求 (p. 226)的现有 VPC 以在其中创建集群。在选择子网之前，我们建议您熟悉 the section called “VPC 和子网要求” (p. 226) 中的所有要求和考虑因素。集群创建后，您无法更改要使用的 VPC。如果没有列出任何 VPC，则需要先创建一个。有关更多信息，请参阅the section called “创建 VPC” (p. 228)。
   - Subnets（子网）– 预设情况下，已预先选中在之前字段中指定的 VPC 中的所有可用子网。您必须至少选择两个子网。

     您选择的子网必须符合 Amazon EKS 子网要求 (p. 227)。在选择子网之前，我们建议您熟悉所有的 Amazon EKS VPC 以及子网要求和考虑因素 (p. 226)。集群创建后，您无法更改要使用的子网。

     Security groups（安全组）–（可选）指定您希望 Amazon EKS 将之与其创建的网络接口关联的一个或多个安全组。

     无论您是否选择任何安全组，Amazon EKS 都会创建一个安全组，以实现集群和 VPC 之间的通信。Amazon EKS 将此安全组以及您选择的任何安全组与它创建的网络接口关联起来。有关 Amazon EKS 创建的集群安全组的更多信息，请参阅 the section called “安全组要求” (p. 231)。您可以修改 Amazon EKS 创建的集群安全组中的规则。如果您选择添加自己的安全组，则无法更改集群创建后选择的安全组。
   - Choose cluster IP address family（选择集群 IP 地址系列）- 如果您选择的集群版本为 1.20 版或更早版本，则仅 IPv4 选项可用。如果您选择了 1.21 或更高版本，则 IPv4 和 IPv6 可用。

默认情况下，Kubernetes 将 `IPv4` 地址分配给容器组（pod）和服务。在决定使用 `IPv6` 系列前，请确保您熟悉 the section called “VPC 要求和注意事项” (p. 226)、the section called “子网要求和注意事项” (p. 227)、the section called “安全组要求” (p. 231) 和 the section called “IPv6” (p. 248) 主题中的所有考虑因素和要求。如果您选择 `IPv6` 系列，您无法指定从中分配 `IPv6` 服务地址的 Kubernetes 的地址范围，就像您可以为 `IPv4` 系列进行的那样。Kubernetes 从唯一的本地地址范围（`fc00::/7`）中分配服务地址。

- （可选）选择 Configure Kubernetes Service IP address range（配置服务 IP 地址范围），然后指定 Service **`IPv4`** range（服务范围）。

  指定自己的范围有助于防止 Kubernetes 服务与对等或连接到您的 VPC 的其他网络之间的冲突。使用 CIDR 表示法输入范围。例如：`10.2.0.0/16`。

  此 CIDR 块必须满足以下要求：

  - 处于以下范围之一：`10.0.0.0/8`、`172.16.0.0/12` 或 `192.168.0.0/16`。
  - 具有最小大小 `/24` 和最大大小 `/12`。
  - 与您的 Amazon EKS 资源的 VPC 范围不重叠。

  您只能在使用 `IPv4` 地址系列和创建集群时指定此选项。如果您没有指定此选项，Kubernetes 从 `10.100.0.0/16` 或 `172.20.0.0/16` CIDR 块分配服务 IP 地址。

- 对于集群端点访问中，选择一个选项。创建集群后，您可以更改此选项。在选择非默认选项之前，请务必熟悉这些选项及其影响。有关更多信息，请参阅the section called “配置端点访问” (p. 38)。

6. 您可以接受 Networking add-ons（联网附加组件）部分的默认值，安装默认版本的 Amazon VPC CNI plugin for Kubernetes (p. 233)、CoreDNS (p. 294) 和 `kube-proxy` (p. 300) Amazon EKS 附加组件。或者，您也可以选择其他版本。如果您不需要任何附加组件的功能，则可以在创建集群后将其删除。如果您需要自行管理任何这些附加组件的 Amazon EKS 管理设置，则在集群创建后删除附加组件的 Amazon EKS 管理。有关更多信息，请参阅Amazon EKS 附加组件 (p. 341)。
7. 选择 Next（下一步）。
8. 在 Configure logging (配置日志记录) 页面上，您可以选择要启用的日志类型，此操作是可选的。预设情况下，每个日志类型均为已禁用。在选择其他选项前，请自行熟悉 Amazon EKS 控制面板日志记录 (p. 44) 中的信息。创建集群后，您可以更改此选项。
9. 选择 Next（下一步）。
10. 在 Review and create (审核和创建) 页面上，审核您在之前页面输入或选择的信息。如果需要进行更改，请选择 Edit（编辑）。在您感到满意后，选择 Create（创建）。Status（状态）字段在预置集群时显示 CREATING（正在创建）。

    Note

    您可能会收到一个错误，指示请求中的可用区之一没有足够容量来创建 Amazon EKS 集群。如果发生这种情况，错误输出将包含可支持新集群的可用区。再次尝试使用至少两个位于您账户中支持的可用区的子网创建集群。有关更多信息，请参阅容量不足 (p. 478)。

集群预配置需要几分钟时间。

Amazon CLI

创建集群的步骤

1. 使用以下命令创建集群。在运行命令之前，进行以下替换：
   - 将 *`region-code`* 替换为您要在其中创建集群的 Amazon Web Services 区域。

- 将 *my-cluster* 替换为您的集群名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。对于您在其中创建集群的 Amazon Web Services 区域 和 Amazon Web Services 账户，该名称必须在其内具有唯一性。
- 将 *1.23* 替换为 Amazon EKS 支持的版本 (p. 59)。
- 将 *111122223333* 替换为您的账户 ID，并将 *myAmazonEKSClusterRole* 替换为您的集群 IAM 角色名称。
- 将 `subnetIds` 的值替换为您自己的值。您还可以添加其他 ID。您必须指定至少两个子网 ID。

  您选择的子网必须符合 Amazon EKS 子网要求 (p. 227)。在选择子网之前，我们建议您熟悉所有的 Amazon EKS VPC 以及子网要求和考虑因素 (p. 226)。集群创建后，您无法更改要使用的子网。
- 如果您不想指定安全组 ID，请从命令中删除 `,securityGroupIds=sg-`*ExampleID1*。如果您想指定一个或多个安全组 ID，请将 `securityGroupIds` 的值替换为您自己的值。您还可以添加其他 ID。

  无论您是否选择任何安全组，Amazon EKS 都会创建一个安全组，以实现集群和 VPC 之间的通信。Amazon EKS 将此安全组以及您选择的任何安全组与它创建的网络接口关联起来。有关 Amazon EKS 创建的集群安全组的更多信息，请参阅 the section called “安全组要求” (p. 231)。您可以修改 Amazon EKS 创建的集群安全组中的规则。如果您选择添加自己的安全组，则无法更改集群创建后选择的安全组。

```
aws eks create-cluster --region region-code --name my-cluster --kubernetes-
version 1.23 \
   --role-arn arn:aws:iam::111122223333:role/myAmazonEKSClusterRole \
   --resources-vpc-config
 subnetIds=subnet-ExampleID1,subnet-ExampleID2,securityGroupIds=sg-ExampleID1
```

  Note

  您可能会收到一个错误，指示请求中的可用区之一没有足够容量来创建 Amazon EKS 集群。如果发生这种情况，错误输出将包含可支持新集群的可用区。再次尝试使用至少两个位于您账户中支持的可用区的子网创建集群。有关更多信息，请参阅容量不足 (p. 478)。

  可选设置

  以下是可选设置，如果需要，必须将这些设置添加到上一个命令中。您只能在创建集群时启用这些选项，而不能在创建集群后启用。

- 如果您想指定 Kubernetes 从中分配服务 ID 地址的 `IPv4` 无类别域间路由块，您必须通过将 **`--kubernetes-network-config serviceIpv4Cidr=`*`CIDR block`* 添加到以下命令中来指定它。

  指定自己的范围有助于防止 Kubernetes 服务与对等或连接到您的 VPC 的其他网络之间的冲突。使用 CIDR 表示法输入范围。例如：`10.2.0.0/16`。

  此 CIDR 块必须满足以下要求：

  - 处于以下范围之一：`10.0.0.0/8`、`172.16.0.0/12` 或 `192.168.0.0/16`。
  - 具有最小大小 `/24` 和最大大小 `/12`。
  - 与您的 Amazon EKS 资源的 VPC 范围不重叠。

  您只能在使用 `IPv4` 地址系列和创建集群时指定此选项。如果您没有指定此选项，Kubernetes 从 `10.100.0.0/16` 或 `172.20.0.0/16` CIDR 块分配服务 IP 地址。

2. 预置集群需要几分钟时间。可使用以下命令查询集群的状态。

```
aws eks describe-cluster \
    --region region-code \
    --name my-cluster \
    --query "cluster.status"
```

在返回的输出为 `ACTIVE` 之前，请勿继续执行下一步。

3. 如果您使用 `eksctl` 创建了集群，则可以跳过此步骤。这是因为 `eksctl` 已经为您完成了此步骤。通过向 `kubectl config` 文件添加新上下文来启用 `kubectl` 与您的集群通信。有关如何创建和更新文件的更多信息，请参阅 the section called “为 Amazon EKS 创建 kubeconfig” (p. 365)。

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

输出示例如下。

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

4. 通过运行以下命令以确认与集群的通信。

```
kubectl get svc
```

输出示例如下。

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

5. （建议）要使用某些 Amazon EKS 附加组件，或启用个别 Kubernetes 工作负载以具有特定 Amazon Identity and Access Management（IAM）权限，请为集群创建 IAM OpenID Connect（OIDC）提供商。有关如何为集群创建 IAM OIDC 提供商的说明，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。您只需为集群创建一次 IAM OIDC 提供商。要了解有关 Amazon EKS 附加组件的详情，请参阅 Amazon EKS 附加组件 (p. 341)。要了解有关将特定 IAM 权限分配给工作负载的详情，请参阅 the section called “服务账户的 IAM 角色” (p. 395)。
6. （推荐）在将 Amazon EC2 节点部署到集群之前为 Amazon VPC CNI plugin for Kubernetes 插件配置集群。默认情况下，该插件随集群一起安装。当您将 Amazon EC2 节点添加到集群时，插件将自动部署到您添加的每个 Amazon EC2 节点上。该插件要求您将以下 IAM policy 之一附加到 IAM 角色：

   - **`AmazonEKS_CNI_Policy`** 托管 IAM policy – 如果您的集群使用 `IPv4` 系列
   - 您创建的 IAM policy (p. 246) – 如果您的集群使用 `IPv6` 系列

   您将策略附加到的 IAM 角色可以是节点 IAM 角色，也可以是仅用于插件的专用角色。我们建议将策略附加到此角色。有关创建角色的更多信息，请参阅 the section called “为 IAM 账户配置插件” (p. 243) 或 the section called “节点 IAM 角色” (p. 427)。
7. 如果您使用 Amazon Web Services Management Console 部署了集群，则可以跳过此步骤。Amazon Web Services Management Console在默认情况下会部署 Amazon VPC CNI plugin for Kubernetes、CoreDNS、`kube-proxy` Amazon EKS 附加组件。

   （可选）如果您使用 `eksctl` 或 Amazon CLI 部署集群，则 Amazon VPC CNI plugin for Kubernetes、CoreDNS 和 `kube-proxy` 自行管理的附加组件将会被部署。您可以将使用您的集群部署的 Amazon VPC CNI plugin for Kubernetes、CoreDNS 和 `kube-proxy` 自行管理的附加组件迁移到 Amazon EKS 附加组件中。有关更多信息，请参阅Amazon EKS 附加组件 (p. 341)。

向您建议的后续步骤：

- 创建集群的 IAM 用户或角色是唯一可以访问集群的 IAM 实体。向其他 IAM 用户或角色授予权限 (p. 355)，以便它们可以访问您的集群。
- 如果创建集群的 IAM 用户或角色仅具有先决条件 (p. 21)中引用的最低 IAM 权限，则您可能想要为该 IAM 实体添加额外的 Amazon EKS 权限。有关向 IAM 实体授予 Amazon EKS 权限的更多信息，请参阅 the section called “身份和访问管理” (p. 409)。
- 如果您希望创建集群的 IAM 实体或任何其他 IAM 实体在 Amazon EKS 控制台中查看 Kubernetes 资源，请向实体授予 the section called “所需权限” (p. 458)。
- 如果您希望节点和用户从 VPC 内访问您的集群，请为集群启用私有端点。默认情况下，将启用公有端点。如需要，可以在启用私有端点后禁用公有端点。有关更多信息，请参阅the section called “配置端点访问” (p. 38)。
- 为集群启用密钥加密 (p. 42)。
- 为集群配置日志记录 (p. 44)。
- 将节点添加到集群 (p. 84)。
- 如果您计划将工作负载部署到使用 Amazon EBS 卷的集群，并且您创建了 `1.23` 或更高版本的集群，则在部署工作负载之前，您必须将 the section called “Amazon EBS CSI 驱动程序” (p. 195) 安装到您的集群。

更新 Amazon EKS 集群 Kubernetes 版本

当 Amazon EKS 中有新的 Kubernetes 版本可用时，您可以将 Amazon EKS 集群更新到最新版本。

> Important
>
> 我们建议您在更新到新的 Kubernetes 版本之前，先查看 Amazon EKS Kubernetes 版本 (p. 59) 中的信息，同时查看此主题中的更新步骤。如果您要更新到版本 `1.22`，您必须对集群进行 the section called “Kubernetes 版本 `1.22` 的先决条件” (p. 32) 中列出的更改，然后再更新它。

新的 Kubernetes 版本有时会引入重大更改。因此，我们建议您先针对新的 Kubernetes 版本测试您应用程序的行为，然后再更新生产集群。您可以通过构建持续集成工作流来测试应用程序行为，然后再移至新的 Kubernetes 版本来执行此操作。

更新过程包含 Amazon EKS 随 Kubernetes 的更新版本推出新的 API 服务器节点，以取代现有此类节点。Amazon EKS 对这些新节点上的网络流量执行标准基础设施和就绪运行状况检查，以确认它们是否按预期工作。如果任意一项检查失败，Amazon EKS 都将恢复基础设施部署，且您的集群仍为先前的 Kubernetes 版本。正在运行的应用程序不会受影响，并且您的集群绝不会处于不确定性或不可恢复的状态。Amazon EKS 会定期备份所有托管的集群，并且具有在必要时恢复集群的机制。我们会不断评估和改进我们的 Kubernetes 基础设施管理流程。

为了升级集群，Amazon EKS 需要您在创建集群时指定的子网中的最多 5 个空闲 IP 地址。Amazon EKS 在您指定的任何子网中创建新的集群弹性网络接口（网络接口）。网络接口可能在不同于现有网络接口所在的子网中创建，因此请确保您的安全组规则允许您对创建集群时指定的任何子网进行所需的集群通信 (p. 231)。如果您在创建集群时指定的任何子网不存在，没有足够的可用 IP 地址，或者没有允许必要的集群通信的安全组规则，则更新可能会失败。

> Note
>
> 尽管 Amazon EKS 运行高度可用的控制层面，但您可能在更新期间遇到次要服务中断。例如，假设您在终止 API 服务器并使用运行新版 Kubernetes 的新 API 服务器替换此服务器时，尝试连接到此服务器。您可能会遇到 API 调用错误或连接问题。如果出现这种情况，请重试您的 API 操作直至成功。

更新 Amazon EKS 集群的 Kubernetes 版本

更新集群的 Kubernetes 版本的步骤

1. 比较集群控制面板的 Kubernetes 版本与节点的 Kubernetes 版本。

- 使用 **`kubectl version --short`** 命令获取集群控制面板的 Kubernetes 版本。

```
kubectl version --short
```

- 使用 **`kubectl get nodes`** 命令获取节点的 Kubernetes 版本。此命令会返回所有自我管理和托管的 Amazon EC2 和 Fargate 节点。每个 Fargate pod 都作为其自身的节点列出。

```
kubectl get nodes
```

将控制面板升级到新的 Kubernetes 版本之前，确保集群中的托管节点和 Fargate 节点的 Kubernetes 次要版本与控制面板的版本相同。例如，如果您的控制面板正在运行版本 `1.22` 且您的其中一个节点正在运行版本 `1.21`，则您必须将节点更新为版本 `1.22`。我们还建议您，在更新控制层面之前将自我管理节点更新为与控制层面相同的版本。有关更多信息，请参阅 更新托管节点组 (p. 95) 和 自行管理节点的更新 (p. 119)。要更新 Fargate 节点的版本，请先删除节点所表示的 pod。然后更新您的控制面板。任何剩余的 pods 都将在重新部署之后更新到新版本。

2. 默认情况下，Amazon EKS 集群上会启用 pod 安全策略准入控制器。在升级集群前，请确保已实施适当的 pod 安全策略。这是为了避免潜在的安全问题。您可以使用 **`kubectl get psp eks.privileged`** 命令检查默认策略。

```
kubectl get psp eks.privileged
```

如果您收到以下错误，请参阅默认 pod 安全策略 (p. 453)，然后再继续操作。

```
Error from server (NotFound): podsecuritypolicies.extensions "eks.privileged" not found
```

3. 如果您最初部署集群所用的 Kubernetes 版本是 Kubernetes `1.18` 或更高版本，请跳过此步骤。

您可能需要从您的 CoreDNS 清单中删除已停用的术语。

a. 检查您的 CoreDNS 清单是否有一行仅包含词语 `upstream`。

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' | grep
 upstream
```

如果未返回任何输出，则说明您的清单没有此类行。如果是这样，请跳至下一步。如果返回了词语 `upstream`，则删除该行。

b. 删除文件顶部附近在 configmap 文件中仅包含词语 `upstream` 的行。不要更改文件中的任何其他内容。删除该行后，保存更改。

```
kubectl edit configmap coredns -n kube-system -o yaml
```

4. 使用 `eksctl`、Amazon Web Services Management Console 或 Amazon CLI 更新您的集群。

Important

- 如果您要更新到版本 `1.22`，您必须对集群进行 the section called “Kubernetes 版本 1.22 的先决条件” (p. 32) 中列出的更改，然后再更新它。
- 如果您要在集群中更新到版本 `1.23` 并使用 Amazon EBS 卷，请在将集群更新为版本 `1.23` 之前，在集群中安装 Amazon EBS CSI 驱动程序，避免中断工作负载。有关更多信息，请参阅 the section called “Kubernetes 1.23” (p. 59) 和 the section called “Amazon EBS CSI 驱动程序” (p. 195)。

- 由于 Amazon EKS 运行高度可用的控制层面，您可以一次只更新一个次要版本。有关此要求的更多信息，请参阅 Kubernetes 版本和版本偏差支持策略。假设集群的当前版本为 `1.21`，而且您想更新到 `1.23`，则必须先将集群更新为 `1.22`，再将集群从 `1.22` 更新到 `1.23`。
- 在更新之前，请确保您的托管节点和 Fargate 节点上的 `kubelet` 与控制面板的 Kubernetes 版本相同。我们建议您将自行管理节点保持在与控制面板相同的版本。它们最多只能比控制面板的当前版本落后一个版本。
- 如果您的集群配置了早于 `1.8.0` 的 Amazon VPC CNI 插件版本，那么我们建议您将插件更新为版本 `1.11.3`，然后再将集群更新为版本 `1.21` 或更高版本。有关更多信息，请参阅 更新 Amazon VPC CNI plugin for Kubernetes 附加组件 (p. 236) 或 更新 Amazon VPC CNI plugin for Kubernetes 自行管理的附加组件 (p. 238)。

eksctl

此过程需要 `eksctl` 版本 `0.110.0` 或更高版本。可以使用以下命令来查看您的版本：

```
eksctl version
```

有关如何安装或更新 `eksctl` 的说明，请参阅 安装或更新 `eksctl` (p. 10)。

使用以下命令将 Amazon EKS 控制面板的 Kubernetes 版本更新到比当前版本高的一个次要版本。将 *my-cluster* 替换为您的集群名称。

```
eksctl upgrade cluster --name my-cluster --approve
```

更新过程可能需要几分钟才能完成。

Amazon Web Services Management Console

a. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
b. 选择要更新的 Amazon EKS 集群的名称，然后选择 Update cluster version (更新集群版本)。
c. 对于 Kubernetes version（版本），选择集群要更新到的版本，然后选择 Update（更新）。
d. 对于 Cluster name（集群名称），输入您的集群的名称并选择 Confirm（确认）。

更新过程可能需要几分钟才能完成。

Amazon CLI

a. 使用以下 Amazon CLI 命令更新 Amazon EKS 集群。将 *example values* 替换为您自己的值。

```
aws eks update-cluster-version \
 --region region-code \
 --name my-cluster \
 --kubernetes-version 1.23
```

输出示例如下。

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
```

```
                "value": "1.23"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
...
        "errors": []
    }
}
```

b. 使用以下命令监控集群更新的状态。使用上一命令返回的集群名称和更新 ID。当 `Successful` 状态显示时，更新完成。更新过程可能需要几分钟才能完成。

```
aws eks describe-update \
  --region region-code \
  --name my-cluster \
  --update-id b5f0ba18-9a87-4450-b5a0-825e6e84496f
```

输出示例如下。

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "Successful",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.23"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
...
        "errors": []
    }
}
```

5. 集群更新完成后，将节点更新为与已更新的集群相同的 Kubernetes 次要版本。有关更多信息，请参阅自行管理节点的更新 (p. 119) 和 更新托管节点组 (p. 95)。在 Fargate 上启动的任何新 pods 都具有与您的集群版本匹配的 `kubelet` 版本。不会更改现有 Fargate pods。
6. （可选）如果您在更新集群之前将 Kubernetes Cluster Autoscaler 部署到了集群，请将 Cluster Autoscaler 更新为与您升级后的 Kubernetes 主版本和次要版本匹配的最新版本。

   a. 在 Web 浏览器中打开 Cluster Autoscaler 版本页面，找到与您集群的 Kubernetes 主版本和次要版本相匹配的 Cluster Autoscaler 最新版本。例如，如果您集群的 Kubernetes 版本是 `1.23`，则查找以 `1.23` 开头的 Cluster Autoscaler 最新版本。记录该版本的语义版本号 (`<1.23.n>`) 以在下一步中使用。

   b. 使用以下命令，将 Cluster Autoscaler 映像标签设置为您在上一步中记录的版本。如有必要，将 *1.23.n* 替换为您自己的值。

```
kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-
autoscaler=public.ecr.aws/bitnami/cluster-autoscaler:v1.23.n
```

7. （仅限于具有 GPU 节点的集群）如果您的集群具有带 GPU 支持的节点组（例如，`p3.2xlarge`），则您必须使用以下命令在您的集群上更新适用于 Kubernetes 的 NVIDIA 设备插件 DaemonSet。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/
nvidia-device-plugin.yml
```

8. 更新 Amazon VPC CNI plugin for Kubernetes、CoreDNS 和 kube-proxy 附加组件。如果您将集群更新为版本 1.21 或更新版本，我们建议将附加组件更新为服务账户令牌 (p. 395)中列出的最低版本。

   - 如果您将集群更新为版本 1.18，则可以添加 Amazon EKS 附加组件。有关说明，请参阅 添加 Amazon VPC CNI Amazon EKS 附加组件 (p. 234)、添加 CoreDNS Amazon EKS 附加组件 (p. 295) 和 添加 kube-proxy Amazon EKS 附加组件 (p. 301)。要了解有关 Amazon EKS 附加组件的详情，请参阅 Amazon EKS 附加组件 (p. 341)。
   - 如果您已更新到 1.19 或更高版本，并且正在使用 Amazon EKS 附加组件，请在 Amazon EKS 控制台中选择 Clusters（集群），然后在左侧导航窗格中选择更新的集群的名称。控制台中显示通知。这些通知将告知您有一个新版本可用于每个具有可用更新的附加组件。要更新附加组件，请选择 Add-ons（附加组件）选项卡。在具有可用更新的附加组件的其中一个框中，选择 Update now（立即更新），选择一个可用版本，然后选择 Update（更新）。
   - 此外，您也可以使用 Amazon CLI 或 eksctl 更新 Amazon VPC CNI plugin for Kubernetes (p. 238)、CoreDNS (p. 296) 和 kube-proxy (p. 302) Amazon EKS 附加组件。

Kubernetes 版本 1.22 的先决条件

许多已弃用的测试版 API（v1beta1）已在版本 1.22 中删除，以支持这些相同 API 的 GA（v1）版本。正如 Kubernetes 版本 1.22 API 和功能删除博客以及已弃用的 API 迁移指南中所提到的，在将集群更新为版本 1.22 之前，需要对以下部署的资源进行 API 更改。

在将集群更新为 Kubernetes 版本 1.22 之前，请确保执行以下操作：

- 更改您的 YAML 清单文件和客户端以引用新 API。
- 更新自定义集成和控制器以调用新的 API。
- 确保使用任何第三方工具的更新版本。这些工具包括入口控制器、服务网格控制器、持续交付系统以及调用新 API 的其他工具。要检查集群中已停用的 API 使用情况，启用审计控制面板日志记录并将 v1beta 指定为事件筛选条件。Kubernetes 中提供了多个版本的替换 API。
- 如果您目前有部署到集群的 Amazon 负载均衡器控制器，您必须将其更新为版本 2.4.1，然后再将您的集群更新到 Kubernetes 版本 1.22。

> **Important**
>
> 当您将集群更新为版本 1.22 时，可以使用新 API 访问现有的持久化对象。但是，您必须迁移清单并更新客户端才能使用这些新 API。更新集群可防止潜在的工作负载故障。

Kubernetes 版本 1.22 从以下测试版 API 中删除了支持。根据以下信息迁移清单和 API 客户端：

资源	测试版本	目标版本	注意
ValidatingWebhookConfiguration MutatingWebhookConfiguration	admissionregistration.k8s.io/v1beta1	admissionregistration.k8s.io/v1	• v1 的 webhooks[*].failurePolicy 默认设置已从 Ignore 更改为 Fail。 • v1 的 webhooks[*].matchPolicy 默认设置已从 Exact 更改为 Equivalent。 • v1 的 webhooks[*].timeoutSeconds 默认设置已从 30s 更改为 10s。

资源	测试版本	目标版本	注意
			• webhooks[*].sideEffects 默认值被删除，字段变为必填字段，且仅 v1 只允许 None 和 NoneOnDryRun。 • webhooks[*].admissionReviewVersions 默认值被删除，而且字段对于 v1 为必填字段（AdmissionReview 的支持的版本为 v1 和 v1beta1）。 • 对于通过 admissionregistration.k8s.io/v1 创建的对象，webhooks[*].name 在列表中必须是唯一的。
CustomResourceDefinition	apiextensions.k8s.io/v1beta1	apiextensions.k8s.io/v1	• spec.scope 不再默认设置为 Namespaced 并且必须明确指定。 • spec.version 已在 v1 中删除，改用 spec.versions • spec.validation 已在 v1 中删除，改用 spec.versions[*].schema。 • spec.subresources 已在 v1 中删除，改用 spec.versions[*].subresources。 • spec.additionalPrinterColumns已在 v1 中删除；改用 spec.versions[*].additionalPrinterColumns。 • spec.conversion.webhookClientConfig 已在 v1 中迁移至 spec.conversion.webhook.clientConfig。 • spec.conversion.conversionReviewVersion 已在 v1 中迁移至 spec.conversion.webhook.conversionRevie • spec.versions[*].schema.openAPIV3Schema 现在在创建 v1 CustomResourceDefinition 对象时是必填项，并且必须是结构化模式。 • 创建 v1 CustomResourceDefinition 时不允许 spec.preserveUnknownFields: true；必须在模式定义中将其指定为 x-kubernetes-preserve-unknown-fields: true。 • 在 additionalPrinterColumns 项目中，JSONPath 字段在 v1 中被重命名为 jsonPath（修复 #66531）。
APIService	apiregistration.k8s.io/v1beta1	apiregistration.k8s.io/v1	None（无）

资源	测试版本	目标版本	注意
TokenReview	authentication.k8s.io/v1beta1	authentication.k8s.io/v1	None（无）
SubjectAccessReview LocalSubjectAccessReview SelfSubjectAccessReview	authorization.k8s.io/v1beta1	authorization.k8s.io/v1	spec.group 已重命名为 spec.groups
CertificateSigningRequest	certificates.k8s.io/v1beta1	certificates.k8s.io/v1	对于请求证书的 API 客户端： • spec.signerName 现在是必需的（请参阅已知的 Kubernetes 签名者），且不允许通过 certificates.k8s.io/v1 API 创建 kubernetes.io/legacy-unknown 的请求 • spec.usages 现在是必需的，可能不包含重复值，并且必须只包含已知的用法 • 对于批准或签名证书的 API 客户端： • status.conditions 可能不包含重复的类型 • 现在需要使用 status.conditions[*].status • status.certificate 必须是 PEM 编码的，并且只包含 CERTIFICATE 块
Lease	coordination.k8s.io/v1beta1	coordination.k8s.io/v1	None（无）
Ingress	• extensions/v1beta1 • networking.k8s.io/v1beta1	networking.k8s.io/v1	• spec.backend 已重命名为 spec.defaultBackend • serviceName 字段被重命名为 service.name • 数字后端 servicePort 字段被重命名为 service.port.number • 字符串后端 servicePort 字段被重命名为 service.port.name • pathType 现在是每个指定路径的必填项。可选值有 Prefix、Exact 和 ImplementationSpecific。要匹配未定义的 v1beta1 行为，请使用 ImplementationSpecific
IngressClass	networking.k8s.io/v1beta1	networking.k8s.io/v1	None（无）
RBAC	rbac.authorization.k8s.io/v1beta1	rbac.authorization.k8s.io/v1	无
PriorityClass	scheduling.k8s.io/v1beta1	scheduling.k8s.io/v1	无

资源	测试版本	目标版本	注意
CSIDriver CSINode StorageClass VolumeAttachment	storage.k8s.io/v1beta1	storage.k8s.io/v1	None (无)

要了解有关 API 移除的更多信息，请参阅已弃用的 API 迁移指南。

删除 Amazon EKS 集群

使用完 Amazon EKS 集群后，应删除与其关联的资源，这样便不会产生任何不必要的费用。

要删除连接的集群，请参阅 注销集群 (p. 500)

Important

- 如果集群中具有与负载均衡器关联的有效服务，则必须先删除这些服务，然后再删除集群，以便正确删除负载均衡器。否则，VPC 中可能有阻止您删除 VPC 的孤立资源。
- 如果您因为已删除集群创建者而收到错误，请参阅这篇文章解决。

您可以使用 eksctl、Amazon Web Services Management Console 或 Amazon CLI 删除集群。选择带有要用于删除集群的工具的名称的选项卡。

eksctl

使用 **eksctl** 删除 Amazon EKS 集群和节点

此过程需要 eksctl 版本 0.110.0 或更高版本。可以使用以下命令来查看您的版本：

```
eksctl version
```

有关如何安装或升级 eksctl 的说明，请参阅 安装或更新 eksctl (p. 10)。

1. 列出集群中运行的所有服务。

   ```
   kubectl get svc --all-namespaces
   ```

2. 删除具有关联的 EXTERNAL-IP 值的任何服务。这些服务的前面配置了一个 Elastic Load Balancing 负载均衡器，您必须从 Kubernetes 中将其删除才能释放负载均衡器和关联资源。

   ```
   kubectl delete svc service-name
   ```

3. 使用以下命令删除集群及其关联的节点，同时将 *prod* 替换为您的集群名称。

   ```
   eksctl delete cluster --name prod
   ```

   输出：

   ```
   [#]  using region region-code
   [#]  deleting EKS cluster "prod"
   [#]  will delete stack "eksctl-prod-nodegroup-standard-nodes"
   [#]  waiting for stack "eksctl-prod-nodegroup-standard-nodes" to get deleted
   ```

```
[#]  will delete stack "eksctl-prod-cluster"
[#]  the following EKS cluster resource(s) for "prod" will be deleted: cluster. If
 in doubt, check CloudFormation console
```

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 删除 Amazon EKS 集群

1. 列出集群中运行的所有服务。

```
kubectl get svc --all-namespaces
```

2. 删除具有关联的 EXTERNAL-IP 值的任何服务。这些服务的前面配置了一个 Elastic Load Balancing 负载均衡器，您必须从 Kubernetes 中将其删除才能释放负载均衡器和关联资源。

```
kubectl delete svc service-name
```

3. 删除所有节点组和 Fargate 配置文件。
   a. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
   b. 请在左侧导航窗格中，选择 Amazon EKS Clusters（集群），然后在集群的选项卡列表中，选择要删除的集群的名称。
   c. 选择 Compute（计算）选项卡，然后选择要删除的节点组。选择 Delete（删除），输入节点组的名称，然后选择 Delete（删除）。删除集群中的所有节点组。

      Note

      只会列出托管节点组 (p. 87)。

   d. 选择要删除的 Fargate Profile（Fargate 配置文件），选择 Delete（删除），输入配置文件的名称，然后选择 Delete（删除）。删除集群中的所有 Fargate 配置文件。
4. 删除所有自行管理的节点 Amazon CloudFormation 堆栈。
   a. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
   b. 请选择要删除的节点堆栈，然后选择 Delete（删除）。
   c. 在 Delete stack（删除堆栈）确认对话框中，请选择 Delete stack（删除堆栈）。删除集群中的所有自行管理的节点堆栈。
5. 请删除集群。
   a. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
   b. 选择要删除的集群并选择 Delete（删除）。
   c. 在删除集群确认屏幕上，选择 Delete (删除)。
6. （可选）删除 VPC Amazon CloudFormation 堆栈。
   a. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
   b. 请选择要删除的 VPC 堆栈，然后选择 Delete（删除）。
   c. 在 Delete stack（删除堆栈）确认对话框中，请选择 Delete stack（删除堆栈）。

Amazon CLI

使用 Amazon CLI 删除 Amazon EKS 集群

1. 列出集群中运行的所有服务。

```
kubectl get svc --all-namespaces
```

2. 删除具有关联的 EXTERNAL-IP 值的任何服务。这些服务的前面配置了一个 Elastic Load Balancing 负载均衡器，您必须从 Kubernetes 中将其删除才能释放负载均衡器和关联资源。

```
kubectl delete svc service-name
```

3. 删除所有节点组和 Fargate 配置文件。

   a. 使用以下命令列出集群中的节点组。

```
aws eks list-nodegroups --cluster-name my-cluster
```

   Note

   只会列出托管节点组 (p. 87)。

   b. 使用以下命令删除每个节点组。删除集群中的所有节点组。

```
aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
```

   c. 使用以下命令列出集群中的 Fargate 配置文件。

```
aws eks list-fargate-profiles --cluster-name my-cluster
```

   d. 使用以下命令删除每个 Fargate 配置文件。删除集群中的所有 Fargate 配置文件。

```
aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --cluster-name my-cluster
```

4. 删除所有自行管理的节点 Amazon CloudFormation 堆栈。

   a. 使用以下命令列出您的可用 Amazon CloudFormation 堆栈。在生成的输出中查找节点模板名称。

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

   b. 使用以下命令删除每个节点堆栈，同时将 *node-stack* 替换为您的节点堆栈名称。删除集群中的所有自行管理的节点堆栈。

```
aws cloudformation delete-stack --stack-name node-stack
```

5. 使用以下命令删除集群，同时将 *my-cluster* 替换为您的集群名称。

```
aws eks delete-cluster --name my-cluster
```

6. （可选）删除 VPC Amazon CloudFormation 堆栈。

   a. 使用以下命令列出您的可用 Amazon CloudFormation 堆栈。在生成的输出中查找 VPC 模板名称。

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

   b. 使用以下命令删除 VPC 堆栈，同时将 *my-vpc-stack* 替换为您的 VPC 堆栈名称。

```
aws cloudformation delete-stack --stack-name my-vpc-stack
```

Amazon EKS 集群端点访问控制

本主题可帮助您为 Amazon EKS 集群的 Kubernetes API 服务器端点启用私有访问，并限制或完全禁用通过 Internet 进行的公有访问。

在创建新集群时，Amazon EKS 将为您用于与集群进行通信的托管 Kubernetes API 服务器（使用 Kubernetes 管理工具，如 `kubectl`）创建端点。预设情况下，此 API 服务器端点对于 Internet 是公有的，对 API 服务器的访问将使用 Amazon Identity and Access Management（IAM）与本机 Kubernetes 基于角色的访问控制（RBAC）相结合的方式加以保护。

您可以启用对 Kubernetes API 服务器的私有访问，以便节点与 API 服务器之间的所有通信都在 VPC 内。您可以限制从 Internet 访问 API 服务器的 IP 地址，或完全禁用对 API 服务器的 Internet 访问。

> Note
>
> 由于此端点用于 Kubernetes API 服务器，而不是用于与 Amazon API 通信的传统 Amazon PrivateLink 端点，所以它不会在 Amazon VPC 控制台中显示为端点。

当您为集群启用端点私有访问时，Amazon EKS 将代表您创建一个 Route 53 私有托管区域，并将它与您的集群的 VPC 关联。这个私有托管区域由 Amazon EKS 管理，它不会出现在您的账户的 Route 53 资源中。为了使私有托管区域正确地将流量路由到您的 API 服务器，您的 VPC 必须将 `enableDnsHostnames` 和 `enableDnsSupport` 设置为 `true`，而且为 VPC 设置的 DHCP 选项必须在其域名服务器列表中包含 `AmazonProvidedDNS`。有关更多信息，请参阅 Amazon VPC 用户指南中的更新 VPC 的 DNS 支持。

您可以在创建新集群时定义 API 服务器终端节点的访问要求，并且可以随时更新集群的 API 服务器终端节点访问。

修改集群终端节点访问

使用本节中的过程来修改现有集群的终端节点访问。下表显示了受支持的 API 服务器终端节点访问组合及其关联的行为。

API 服务器终端节点访问选项

终端节点公有访问	终端节点私有访问	行为
Enabled	已禁用	• 这是新 Amazon EKS 集群的默认行为。 • 源自集群的 VPC 内的 Kubernetes API 请求（如控制面板通信的节点）离开 VPC 但不离开 Amazon 网络。 • 集群 API 服务器可从 Internet 访问。您可以选择性地限制可访问公有终端节点的 CIDR 块。如果限制对特定 CIDR 块的访问，则建议您还启用私有端点，或者确保指定的 CIDR 块包括节点和 Fargate pods（如果您使用这些 Pod）用于访问公有端点的地址。

终端节点公有访问	终端节点私有访问	行为
Enabled	Enabled	• 集群的 VPC 内的 Kubernetes API 请求（如控制面板通信的节点）使用私有 VPC 端点。 • 集群 API 服务器可从 Internet 访问。您可以选择性地限制可访问公有终端节点的 CIDR 块。
已禁用	Enabled	• 传输到集群 API 服务器的所有流量都必须来自您的集群的 VPC 或连接的网络中。 • 没有来自 Internet 的对 API 服务器的公有访问。所有 `kubectl` 命令都必须来自 VPC 或连接的网络中。对于连接选项，请参阅 访问私有 API 服务器 (p. 41)。 • 公有 DNS 服务器将集群的 API 服务器终端节点解析为 VPC 中的私有 IP 地址。过去，终端节点只能在 VPC 内部解析。 如果您的终端节点未解析为现有集群的 VPC 内的私有 IP 地址，您可以： • 启用公有访问，然后重新禁用。您只需为集群执行一次该操作，终端节点就将从该时间点开始解析为私有 IP 地址。 • 更新 (p. 28)您的集群。

您可以使用 Amazon Web Services Management Console或 Amazon CLI 修改集群 API 服务器终端节点访问。选择带有要用于修改端点访问的工具名称的选项卡。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 修改集群 API 服务器端点访问

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 选择集群的名称可以显示集群信息。
3. 选择 Networking（联网）选项卡，然后选择 Update（更新）。
4. 对于 Private access（私有访问），选择是启用还是禁用集群的 Kubernetes API 服务器端点的私有访问。如果启用私有访问，源自集群的 VPC 内的 Kubernetes API 请求将使用私有 VPC 端点。您必须启用私有访问以禁用公有访问。
5. 对于 Public access（公有访问），选择是启用还是禁用集群的 Kubernetes API 服务器端点的公有访问。如果禁用公有访问，集群的 Kubernetes API 服务器只能接收来自集群 VPC 内的请求。
6. （可选）如果您已启用 Public access (公有访问)，则可以指定 Internet 中的哪些地址可以与公有端点通信。选择 Advanced Settings (高级设置)。输入 CIDR 区块，例如 *`203.0.113.5/32`*。该块不能包含预留地址。您可以通过选择 Add Source (添加源) 来输入其他块。您可以指定的 CIDR 块存在最大数量限制。有关更多信息，请参阅Amazon EKS 服务配额 (p. 389)。如果未指定任何块，则公有 API 服务器端点将接收来自所有（`0.0.0.0/0`）IP 地址的请求。如果您使用 CIDR 块

限制对公有端点的访问，建议您还启用私有端点访问，以便节点和 Fargate pods（如果您使用这些 Pod）可以与集群进行通信。在未启用私有终端节点的情况下，您的公有访问终端节点 CIDR 源必须包含来自 VPC 的出口源。例如，如果您在私有子网中有一个节点，该节点通过 NAT 网关与 Internet 通信，则您需要将 NAT 网关的出站 IP 地址添加作为公有端点上的容许 CIDR 块的一部分。

7. 选择 Update (更新) 完成操作。

Amazon CLI

使用 Amazon CLI 修改集群 API 服务器端点访问

使用 Amazon CLI 版本 `1.25.46` 或更高版本完成以下步骤。您可以使用 `aws --version` 检查您的当前版本。要安装或升级 Amazon CLI，请参阅安装 Amazon CLI。

1. 使用下面的 Amazon CLI 命令更新集群 API 服务器终端节点访问。替换您的集群名称和所需的终端节点访问值。如果设置了 `endpointPublicAccess=true`，则可以（可选）输入单个 CIDR 块，或者输入 `publicAccessCidrs` 的用逗号分隔的 CIDR 块列表。块不能包含预留地址。如果您指定 CIDR 块，则公有 API 服务器终端节点将只接收来自列出的块的请求。您可以指定的 CIDR 块存在最大数量限制。有关更多信息，请参阅Amazon EKS 服务配额 (p. 389)。如果您使用 CIDR 块限制对公有端点的访问，建议您还启用私有端点访问，以便节点和 Fargate pods（如果您使用这些 Pod）可以与集群进行通信。在未启用私有终端节点的情况下，您的公有访问终端节点 CIDR 源必须包含来自 VPC 的出口源。例如，如果您在私有子网中有一个节点，该节点通过 NAT 网关与 Internet 通信，则您需要将 NAT 网关的出站 IP 地址添加作为公有端点上的容许 CIDR 块的一部分。如果未指定任何 CIDR 块，则公有 API 服务器终端节点将接收来自所有 (0.0.0.0/0) IP 地址的请求。

   Note

   以下命令为 API 服务器终端节点启用来自单个 IP 地址的私有访问和公有访问。将 *203.0.113.5/32* 替换为要限制网络访问的单个 CIDR 块或用逗号分隔的 CIDR 块列表。

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --resources-vpc-config
 endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

   输出示例如下。

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
```

```
    }
}
```

2. 使用以下命令通过上一命令返回的集群名称和更新 ID 监控您的终端节点访问更新的状态。当状态显示为 Successful 时，您的更新将完成。

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000
```

输出示例如下。

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "Successful",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

访问私有 API 服务器

如果您已禁用集群的 Kubernetes API 服务器端点的公有访问，则只能从 VPC 或连接的网络内访问 API 服务器。以下是访问 Kubernetes API 服务器端点的部分可行方法：

- 连接的网络 – 使用 Amazon Transit Gateway 或其他连接选项将您的网络连接到 VPC，然后使用所连接的网络中的计算机。必须确保您的 Amazon EKS 控制层面安全组规则允许来自您的连接网络的端口 443 上的入口流量。
- Amazon EC2 堡垒主机 – 您可以在集群的 VPC 中将 Amazon EC2 实例启动到公有子网中，然后通过 SSH 登录到该实例来运行 kubectl 命令。有关更多信息，请参阅 Amazon 上的 Linux 堡垒机主机。必须确保您的 Amazon EKS 控制层面安全组规则允许来自您的堡垒主机的端口 443 上的入口流量。有关更多信息，请参阅Amazon EKS 安全组要求和注意事项 (p. 231)。

  在为堡垒主机配置 kubectl 时，请确保使用已映射到集群的 RBAC 配置的Amazon凭证，或在删除端点公有访问之前添加您的堡垒将用于 RBAC 配置的 IAM 用户或角色。有关更多信息，请参阅 让 IAM 用户和角色有权访问您的集群 (p. 355) 和 未经授权或访问被拒绝 (kubectl) (p. 478)。
- Amazon Cloud9 IDE – Amazon Cloud9 是一种基于云的集成开发环境 (IDE)，您只需要一个浏览器即可编写、运行和调试代码。您可以在集群的 VPC 中创建 Amazon Cloud9 IDE，然后使用 IDE 来与集群进行通信。有关更多信息，请参阅在 Amazon Cloud9 中创建环境。您必须确保 Amazon EKS 控制层面安全组包含允许来自 IDE 安全组的端口 443 上的入口流量的规则。有关更多信息，请参阅Amazon EKS 安全组要求和注意事项 (p. 231)。

在为 Amazon Cloud9 IDE 配置 `kubectl` 时，请确保使用已映射到集群的 RBAC 配置的Amazon凭证，或在删除端点公有访问之前添加您的 IDE 将用于 RBAC 配置的 IAM 用户或角色。有关更多信息，请参阅 让 IAM 用户和角色有权访问您的集群 (p. 355) 和 未经授权或访问被拒绝 (kubectl) (p. 478)。

在现有集群中启用密钥加密

如果启用密钥加密，将使用您选择的 Amazon KMS key 对 Kubernetes 密钥加密。KMS 密钥必须符合以下条件：

- 对称
- 可以加密和解密数据
- 在与集群相同的 Amazon Web Services 区域 中创建
- 如果 KMS 密钥是在其他账户中创建的，则用户必须拥有对 KMS 密钥的访问权限。

有关更多信息，请参阅 Amazon Key Management Service 开发人员指南中的允许其他账户中的用户使用 KMS 密钥。

> **Warning**
>
> 密钥加密启用后将无法禁用。此操作不可逆。

eksctl

可以通过下列两种方式启用加密：

- 使用单个命令将加密添加到您的集群。

  要自动重新加密密钥，请运行以下命令。

```
eksctl utils enable-secrets-encryption \

    --cluster my-cluster \
    --key-arn arn:aws:kms:region-code:account:key/key
```

  要选择退出自动重新加密密钥，请运行以下命令。

```
eksctl utils enable-secrets-encryption
    --cluster my-cluster \
    --key-arn arn:aws:kms:region-code:account:key/key \
    --encrypt-existing-secrets=false
```

- 使用 `.yaml` 文件向集群添加加密。

```
# cluster.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

secretsEncryption:
  keyARN: arn:aws:kms:region-code:account:key/key
```

要自动重新加密密钥，请运行以下命令。

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

要选择退出自动重新加密密钥，请运行以下命令。

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-
secrets=false
```

Amazon Web Services Management Console

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择要向其添加 KMS 加密的集群。
3. 选择 Overview（概述）选项卡（默认处于选中状态）。
4. 向下滚动到 Secrets encryption（密钥加密）部分，然后选择 Enable（启用）按钮。
5. 从下拉列表中选择一个密钥，然后选择 Enable（启用）按钮。如果未列出任何密钥，则必须先创建一个密钥。有关更多信息，请参阅创建密钥。
6. 选择 Confirm（确认）按钮以使用选定的密钥。

Amazon CLI

1. 使用以下 Amazon CLI 命令将密钥加密配置与您的集群相关联。将 *example values* 替换为您自己的值。

```
aws eks associate-encryption-config \
    --cluster-name my-cluster \
    --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws:kms:region-code:account:key/key"}}]'
```

输出如下所示。

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734,
    "errors": []
  }
}
```

2. 您可以使用以下命令监控加密更新的状态。使用上一个输出中返回的特定 `cluster name` 和 `update ID`。当 `Successful` 状态显示时，更新完成。

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
```

```
    --update-id 3141b835-8103-423a-8e68-12c2521ffa4d
```

输出如下所示。

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734>,
    "errors": []
  }
}
```

3. 要验证集群已启用加密，请运行 `describe-cluster` 命令。响应包含 `EncryptionConfig` 字符串。

```
aws eks describe-cluster --region region-code --name my-cluster
```

在集群上启用加密后，您必须使用新密钥加密所有现有密钥：

Note

如果您使用 `eksctl`，只有在选择不自动重新加密密钥时才需要运行以下命令。

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-
encryption-timestamp="time value"
```

Warning

如果您为现有集群启用密钥加密，并且您使用的 KMS 密钥已被删除，那么将无法恢复集群。如果您删除 KMS 密钥，会将集群永久性置于降级状态。有关更多信息，请参阅删除 Amazon KMS 密钥.。

Note

预设情况下，`create-key` 命令会创建一个具有密钥策略的对称加密 KMS 密钥，该密钥策略向账户的根管理员授予对 Amazon KMS 操作和资源的访问权限。如果要缩小权限的范围，请确保允许对将调用 `create-cluster` API 的主体的策略执行 `kms:DescribeKey` 和 `kms:CreateGrant` 操作。
对于使用 KMS 信封加密的集群，需要具有 `kms:CreateGrant` 权限。CreateCluster 操作不支持条件 `kms:GrantIsForAWSResource`，也不应在 KMS 策略中用于控制执行 CreateCluster 的用户的 `kms:CreateGrant` 权限。

Amazon EKS 控制面板日志记录

Amazon EKS 控制层面日志记录将审计和诊断日志直接从 Amazon EKS 控制层面提供到您账户中的 CloudWatch Logs。这些日志可让您轻松地保护和运行您的集群。您可以选择您需要的确切的日志类型，日志将作为日志流发送到 CloudWatch 中每个 Amazon EKS 集群的组。

您可以通过选择您想要为每个新的或现有 Amazon EKS 集群启用的日志类型来开始使用 Amazon EKS 控制层面日志记录。您可以使用 Amazon Web Services Management Console、Amazon CLI（版本 1.16.139 或更高版本）或通过 Amazon EKS API 按集群启用或禁用每种日志类型。启用后，日志将从 Amazon EKS 集群自动发送到同一账户中的 CloudWatch Logs。

使用 Amazon EKS 控制层面日志记录时，您运行的每个集群将按标准 Amazon EKS 定价收费。对于从集群发送到 CloudWatch Logs 的任何日志，将向您收取标准 CloudWatch Logs 数据提取和存储费用。您还需要为预置为集群一部分的任何Amazon资源付费，如 Amazon EC2 实例或 Amazon EBS 卷。

以下集群控制层面日志类型可用。每个日志类型对应一个 Kubernetes 控制面板组件。要了解有关这些组件的更多信息，请参阅 Kubernetes 文档中的 Kubernetes 组件。

- Kubernetes API 服务器组件日志 (**api**) – 集群的 API 服务器是公开 Kubernetes API 的控制面板组件。有关更多信息，请参阅 Kubernetes 文档中的 kube-apiserver 和审计策略。
- 审计 (**audit**) – Kubernetes 审计日志提供单个用户、管理员或影响集群的系统组件的记录。有关更多信息，请参阅 Kubernetes 文档中的审计。
- 身份验证器 (**authenticator**) – 身份验证器日志对于 Amazon EKS 是唯一的。这些日志代表 Amazon EKS 使用 IAM 凭证用于 Kubernetes 基于角色的访问控制 (RBAC, Role based access control) 身份验证的控制面板组件。有关更多信息，请参阅集群管理 (p. 374)。
- 控制器管理器 (**controllerManager**) – 控制器管理器管理 Kubernetes 附带的核心控制环。有关更多信息，请参阅 Kubernetes 文档中的 kube-controller-manager。
- 计划程序 (**scheduler**) – 计划程序组件管理在集群中运行 pods 的时间和位置。有关更多信息，请参阅 Kubernetes 文档中的 kube-scheduler。

启用和禁用控制层面日志

预设情况下，集群控制层面日志不会发送到 CloudWatch Logs 中。您必须单独启用每个日志类型来为集群发送日志。CloudWatch Logs 摄取、归档存储和数据扫描速率适用于已启用的控制层面日志。有关更多信息，请参阅 CloudWatch 定价。

启用日志类型后，发送的日志具有详细程度级别 2。

使用控制台启用或禁用控制层面日志

1. 打开 Amazon EKS console（Amazon EKS 控制台）。
2. 选择集群的名称可以显示集群信息。
3. 在 Logging（日志记录）选项卡上，选择 Manage logging（管理日志记录）。
4. 对于每个日志类型，选择日志类型应为 Enabled (已启用) 还是 Disabled (已禁用)。预设情况下，每个日志类型均为已禁用。
5. 选择 Save changes（保存更改）以完成操作。

使用 Amazon CLI 启用或禁用控制层面日志

1. 使用以下命令查看您的 Amazon CLI 版本。

```
aws --version
```

   如果您的 Amazon CLI 版本低于 1.16.139，则必须先更新到最新版本。要安装或升级 Amazon CLI，请参阅 Amazon Command Line Interface 用户指南中的安装 Amazon Command Line Interface。
2. 使用下面的 Amazon CLI 命令更新集群的控制层面日志导出配置。将 *my-cluster* 替换为您的集群名称并指定所需的端点访问值。

Note

以下命令将所有可用日志类型发送到 CloudWatch Logs。

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

输出示例如下。

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

3. 使用以下命令通过上一命令返回的集群名称和更新 ID 监控您的日志配置更新的状态。当状态显示为 `Successful` 时，您的更新将完成。

```
aws eks describe-update \
    --region region-code\
    --name my-cluster \
    --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9
```

输出示例如下。

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "Successful",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

查看集群控制层面日志

为您的 Amazon EKS 集群启用任何控制层面日志类型后，您可以在 CloudWatch 控制台上进行查看。

要了解有关在 CloudWatch 中查看、分析和管理日志的详情，请参阅 Amazon CloudWatch Logs 用户指南。

在 CloudWatch 控制台上查看集群控制层面日志

1. 打开 CloudWatch console（CloudWatch 控制台）。此链接将打开控制台并显示您当前可用的日志组，并使用 `/aws/eks` 前缀筛选它们。
2. 选择您要查看日志的集群。日志组名称格式为 `/aws/eks/my-cluster/cluster`。
3. 选择要查看的日志流。以下列表介绍了每个日志类型的日志流名称格式。

   Note

   随着日志流数据增加，日志流名称将轮换。当某个日志类型存在多个日志流时，您可以通过查找包含最新的上次事件时间的日志流名称来查看最新的日志流。

   - Kubernetes API 服务器组件日志 (**api**) – `kube-apiserver-<nnn...>`
   - 审计 (**audit**) – `kube-apiserver-audit-<nnn...>`
   - 身份验证器 (**authenticator**) – `authenticator-<nnn...>`
   - 控制器管理器 (**controllerManager**) – `kube-controller-manager-<nnn...>`
   - 调度器 (**scheduler**) – `kube-scheduler-<nnn...>`

在 Amazon CloudWatch 控制台查看 API 服务器标志

您可以使用 Amazon EKS 集群的控制层面日志记录功能查看创建集群时启用的 API 服务器标记。有关更多信息，请参阅Amazon EKS 控制面板日志记录 (p. 44)。本主题介绍如何在 Amazon CloudWatch 控制台中查看 Amazon EKS 集群的 API 服务器标记。

首次创建集群时，初始 API 服务器日志包含用于启动 API 服务器的标记。如果您在启动集群时或在之后不久启用 API 服务器日志，这些日志将被发送到 CloudWatch Logs，您可以在那里进行查看。

查看集群的 API 服务器标记

1. 为 Amazon EKS 集群启用 API 服务器标记（如果您尚未启用）。
   a. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
   b. 选择集群的名称可以显示集群信息。
   c. 在 Logging（日志记录）选项卡上，选择 Manage logging（管理日志记录）。
   d. 对于 API server (API 服务器)，确保日志类型为 Enabled (启用)。
   e. 选择 Save changes（保存更改）以完成操作。
2. 打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/
3. 选择 Logs（日志），然后选择侧面菜单中的 Log groups（日志组）。选择要查看其日志的集群，然后选择 Log streams（日志流）选项卡。
4. 在日志流列表中，找到 `kube-apiserver-example-ID-288ec988b77a59d70ec77` 日志流的最早版本。使用 Last Event Time (上次事件时间) 列确定日志流的期限。
5. 向上滚动到最早的事件（日志流的开头）。您应该会看到集群的初始 API 服务器标记。

   Note

   如果您在日志流的开头未看到 API 服务器日志，有可能是因为在您在服务器上启用 API 服务器日志记录之前，API 服务器日志文件已在服务器上轮换。在启用 API 服务器日志记录之前轮换的任何日志文件都无法导出到 CloudWatch。

不过，您可以使用相同的 Kubernetes 版本创建新集群，并在创建集群时启用 API 服务器日志记录。具有相同平台版本的集群会启用相同的标记，因此您的标记应该会与新集群的标记匹配。在 CloudWatch 中查看新集群的标记后，您可以删除新集群。

为 Amazon EKS 集群启用 Windows 支持

在部署 Windows 节点之前，请了解以下注意事项。

注意事项

- Windows 工作负载不支持 Amazon EC2 实例类型 `C3`、`C4`、`D2`、`I2`、`M4`（不包括 `m4.16xlarge`）、`M6a.x` 和 `R3` 实例。
- Windows 工作负载不支持主机联网模式。
- Amazon EKS 集群必须包含一个或多个 Linux 或 Fargate 节点，才能运行仅在 Linux 上运行的核心系统 pods，如 CoreDNS。
- `kubelet` 和 `kube-proxy` 事件日志将重定向到 `EKS` Windows 事件日志，并设置为 200 MB 限制。
- 您不能将 pods 的安全组 (p. 273) 用于在 Windows 节点上运行的 pods。
- 您不能将自定义联网 (p. 258)用于 Windows 节点。
- 您不能将 IP 前缀 (p. 270)用于 Windows 节点。这是使用 IPv6 (p. 248) 的要求，因此您也不能将 `IPv6` 用于 Windows 节点。
- Windows 节点支持每个节点一个弹性网络接口。每个 Windows 节点可以运行的 pods 数等于每个弹性网络接口可用于节点实例类型的 IP 地址数减 1。有关更多信息，请参阅适用于 Windows 实例的 Amazon EC2 用户指南中的每个实例类型的每个网络接口的 IP 地址。
- 在 Amazon EKS 集群中，采用负载均衡器的单个服务最多可支持 1024 个后端 pods。每个 pod 都有自己的唯一 IP 地址。对于从 操作系统构建 17763.2746 开始的 Windows 服务器更新，之前的 64 个 pods 限制已经没有了。
- 您不能部署 Windows 托管式节点。您只能创建自行管理的 Windows 节点。有关更多信息，请参阅启动自行管理的 Windows 节点 (p. 114)。
- Fargate 上的 Amazon EKS 容器组（pod）不支持 Windows 容器。
- 无法从 `vpc-resource-controller` Pod 检索日志。以前在将此控制器部署到数据面板时则可以检索。
- 在将 `IPv4` 地址分配给新容器组（pod）之前，会有一段“冷却”时间。这样可以防止流量因过时的 `kube-proxy` 规则而流向具有相同 `IPv4` 地址的旧容器组（pod）。
- 该控制器的源代码在 GitHub 上进行管理。要为该控制器贡献代码或提交针对该控制器的问题，请访问 GitHub 上的相应项目。

先决条件

- 现有集群。该集群必须运行下表中列出的 Kubernetes 版本和平台版本之一。所有比所列版本更高的 Kubernetes 和平台版本也受支持。如果您的集群或平台版本早于以下版本之一，则需要在集群的数据面板上启用旧版 Windows 支持 (p. 51)。如果您的集群为以下 Kubernetes 和平台版本之一或更高版本，您可以在控制面板上删除旧版 Windows 支持 (p. 50)并启用 Windows 支持 (p. 49)。

Kubernetes 版本	平台版本
1.22	eks.1
1.21	eks.3
1.20	eks.3
1.19	eks.7

Kubernetes 版本	平台版本
1.18	eks.9

您的集群必须至少有一个（我们建议至少有两个）Linux 节点或 Fargate pod 才能运行 CoreDNS。如果您启用旧版 Windows 支持，则必须使用 Linux 节点（不能使用 Fargate pod）来运行 CoreDNS。

- 已有一个 Amazon EKS 集群 IAM 角色 (p. 425)。

启用 Windows 支持

如果您的集群不是先决条件 (p. 48)部分中列出的 Kubernetes 和平台版本之一或更高版本，则您必须改而启用旧版 Windows 支持。有关更多信息，请参阅启用旧版 Windows 支持 (p. 51)。

如果从未在集群上启用 Windows 支持，请跳到下一步。

如果您在早于先决条件 (p. 48)列出的 Kubernetes 或平台版本的集群上启用了 Windows 支持，则必须先从数据面板中删除 vpc-resource-controller 和 vpc-admission-webhook (p. 50)。这两者已被弃用，现已不再需要。

为集群启用 Windows 支持

1. 如果您的集群中没有 Amazon Linux 节点并且对 pods 使用了安全组，请跳至下一步。否则，请确认 AmazonEKSVPCResourceController 托管策略是否已附加到您的集群角色 (p. 425)。将 *eksClusterRole* 替换为您的集群角色名称。

   ```
   aws iam list-attached-role-policies --role-name eksClusterRole
   ```

   输出示例如下。

   ```
   {
       "AttachedPolicies": [
           {
               "PolicyName": "AmazonEKSClusterPolicy",
               "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
           },
           {
               "PolicyName": "AmazonEKSVPCResourceController",
               "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"
           }
       ]
   }
   ```

   如果像前面的输出中那样已经附加了策略，请跳过下一步。

2. 将 AmazonEKSVPCResourceController 托管策略附加到您的 Amazon EKS 集群 IAM 角色 (p. 425)。将 *eksClusterRole* 替换为您的集群角色名称。

   ```
   aws iam attach-role-policy \
     --role-name eksClusterRole \
     --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
   ```

3. 使用以下内容创建名为 *vpc-resource-controller-configmap.yaml* 的文件。

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
   ```

```
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-windows-ipam: "true"
```

4. 将 ConfigMap 应用到集群。

```
kubectl apply -f vpc-resource-controller-configmap.yaml
```

从数据面板中删除旧版 Windows 支持

如果您在早于先决条件 (p. 48)列出的 Kubernetes 或平台版本的集群上启用了 Windows 支持，则必须先从数据面板中删除 vpc-resource-controller 和 vpc-admission-webhook。它们已被弃用，现已不再需要，因为它们提供的功能现已在控制面板上启用。

1. 可以使用以下命令卸载 vpc-resource-controller。无论您原来使用哪种工具安装它，现在都请使用此命令。将 *region-code* （仅 /manifests/ 之后的文本实例）替换为集群所在的 Amazon Web Services 区域。

```
kubectl delete -f https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-controller.yaml
```

2. 卸载 vpc-admission-webhook，卸载时请按照您之前安装它时所用工具的说明操作。

   eksctl

   运行以下命令。

   ```
   kubectl delete deployment -n kube-system vpc-admission-webhook
   kubectl delete service -n kube-system vpc-admission-webhook
   kubectl delete mutatingwebhookconfigurations.admissionregistration.k8s.io vpc-admission-webhook-cfg
   ```

   kubectl on macOS or Windows

   运行以下命令。将 *region-code* （仅 /manifests/ 之后的文本实例）替换为集群所在的 Amazon Web Services 区域。

   ```
   kubectl delete -f https://s3.cn-north-1.amazonaws.com.cn/amazon-eks//manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-deployment.yaml
   ```

3. 在控制面板上为集群启用 Windows 支持 (p. 49)。

禁用 Windows 支持

在集群上禁用 Windows 支持

1. 如果您的集群包含 Amazon Linux 节点，并且您对这些节点使用了 pods 安全组 (p. 273)，请跳过此步骤。

   从您的集群角色 (p. 425)中删除 AmazonVPCResourceController 这项托管的 IAM policy。将 *eksClusterRole* 替换为集群角色的名称，并将 *111122223333* 替换为您的账户 ID。

```
aws iam detach-role-policy \
```

```
    --role-name eksClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
```

2. 在 amazon-vpc-cni ConfigMap 中禁用 Windows IPAM。

```
kubectl patch configmap/amazon-vpc-cni \-n kube-system \--type merge \-p '{"data":
{"enable-windows-ipam":"false"}}'
```

部署 Pod

将 Pod 部署到集群时，如果您运行的是多种不同类型的节点，则需要指定这些 Pod 所用的操作系统。

对于 Linux pods，请使用清单中的以下节点选择器文本。

```
nodeSelector:
        kubernetes.io/os: linux
        kubernetes.io/arch: amd64
```

对于 Windows pods，请使用清单中的以下节点选择器文本。

```
nodeSelector:
        kubernetes.io/os: windows
        kubernetes.io/arch: amd64
```

可以部署一款示例应用程序 (p. 315)，以查看正在使用的节点选择器。

启用旧版 Windows 支持

如果您的集群是先决条件 (p. 48)部分中列出的 Kubernetes 和平台版本之一或更高版本，我们建议在控制面板上启用 Windows 支持。有关更多信息，请参阅启用 Windows 支持 (p. 49)。

如果您的集群或平台版本早于先决条件 (p. 48)部分中列出的版本，则按照以下步骤操作有助于您为 Amazon EKS 集群的数据面板启用旧版 Windows 支持。如果您的集群和平台版本是先决条件 (p. 48)部分中列出的版本或更高版本，我们建议您删除旧版 Windows 支持 (p. 50)并为控制面板启用新版支持 (p. 49)。

您可以使用 eksctl、Windows 客户端或者 macOS 或 Linux 客户端为集群启用旧版 Windows 支持。

eksctl

使用 **eksctl** 为您的集群启用旧版 Windows 支持

先决条件

此过程需要 eksctl 版本 0.110.0 或更高版本。可以使用以下命令来查看您的版本。

```
eksctl version
```

有关安装或升级 eksctl 的更多信息，请参阅安装或更新 eksctl (p. 10)。

1. 使用以下 eksctl 命令为您的 Amazon EKS 集群启用 Windows 支持。将 *my-cluster* 替换为您的集群名称。此命令会部署 Amazon EKS 集群上运行 Windows 工作负载所需的 VPC 资源控制器和 VPC 准入控制器 Webhook。

```
eksctl utils install-vpc-controllers --cluster my-cluster --approve
```

Important

VPC 准入控制器 Webhook 使用证书签名，该证书在签发后具有一年有效期。为避免停机，请确保在该证书到期之前更新证书。有关更多信息，请参阅更新 VPC 准入 Webhook 证书 (p. 55)。

2. 启用 Windows 支持后，您可以在集群中启动 Windows 节点组。有关更多信息，请参阅启动自行管理的 Windows 节点 (p. 114)。

Windows

使用 Windows 客户端为您的集群启用旧版 Windows 支持

在以下步骤中，将 *region-code* 替换为您的集群所在的 Amazon Web Services 区域。

1. 将 VPC 资源控制器部署到您的集群。

```
kubectl apply -f https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-
controller.yaml
```

2. 将 VPC 准入控制器 Webhook 部署到您的集群。

   a. 下载所需的脚本和部署文件。

```
curl -o vpc-admission-webhook-deployment.yaml https://s3.cn-
north-1.amazonaws.com.cn/amazon-eks/manifests/region-code/vpc-admission-
webhook/latest/vpc-admission-webhook-deployment.yaml;
curl -o Setup-VPCAdmissionWebhook.ps1 https://s3.cn-north-1.amazonaws.com.cn/
amazon-eks/manifests/region-code/vpc-admission-webhook/latest/Setup-
VPCAdmissionWebhook.ps1;
curl -o webhook-create-signed-cert.ps1 https://s3.cn-north-1.amazonaws.com.cn/
amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-
signed-cert.ps1;
curl -o webhook-patch-ca-bundle.ps1 https://s3.cn-north-1.amazonaws.com.cn/
amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-patch-ca-
bundle.ps1;
```

   b. 安装 OpenSSL 和 jq。

   c. 设置和部署 VPC 准入 Webhook。

```
./Setup-VPCAdmissionWebhook.ps1 -DeploymentTemplate ".\vpc-admission-webhook-
deployment.yaml"
```

Important

VPC 准入控制器 Webhook 使用证书签名，该证书在签发后具有一年有效期。为避免停机，请确保在该证书到期之前更新证书。有关更多信息，请参阅更新 VPC 准入 Webhook 证书 (p. 55)。

3. 确定您的集群是否具有所需的集群角色绑定。

```
kubectl get clusterrolebinding eks:kube-proxy-windows
```

如果返回的输出类似于以下示例输出，则集群具有必要的角色绑定。

```
NAME                       AGE
eks:kube-proxy-windows     10d
```

如果输出包含 `Error from server (NotFound)`，则集群没有必要的集群角色绑定。通过创建一个名为 *eks-kube-proxy-windows-crb.yaml* 且包含以下内容的文件来添加绑定。

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: eks:kube-proxy-windows
  labels:
    k8s-app: kube-proxy
    eks.amazonaws.com/component: kube-proxy
subjects:
  - kind: Group
    name: "eks:kube-proxy-windows"
roleRef:
  kind: ClusterRole
  name: system:node-proxier
  apiGroup: rbac.authorization.k8s.io
```

将配置应用于集群。

```
kubectl apply -f eks-kube-proxy-windows-crb.yaml
```

4. 启用 Windows 支持后，您可以在集群中启动 Windows 节点组。有关更多信息，请参阅启动自行管理的 Windows 节点 (p. 114)。

macOS and Linux

使用 macOS 或 Linux 客户端为您的集群启用旧版 Windows 支持

此过程要求在客户端系统上安装 `openssl` 库和 `jq` JSON 处理器。

在以下步骤中，将 *region-code* 替换为您的集群所在的 Amazon Web Services 区域。

1. 将 VPC 资源控制器部署到您的集群。

```
kubectl apply -f https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-
controller.yaml
```

2. 为您的集群创建 VPC 准入控制器 Webhook 清单。

   a. 下载所需的脚本和部署文件。

```
curl -o webhook-create-signed-cert.sh https://s3.cn-north-1.amazonaws.com.cn/
amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-
signed-cert.sh
curl -o webhook-patch-ca-bundle.sh https://s3.cn-north-1.amazonaws.com.cn/
amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-patch-ca-
bundle.sh
curl -o vpc-admission-webhook-deployment.yaml https://s3.cn-
north-1.amazonaws.com.cn/amazon-eks/manifests/region-code/vpc-admission-
webhook/latest/vpc-admission-webhook-deployment.yaml
```

   b. 向 Shell 脚本添加权限，以便它们能够运行。

```
chmod +x webhook-create-signed-cert.sh webhook-patch-ca-bundle.sh
```

   c. 创建安全通信的密钥。

```
./webhook-create-signed-cert.sh
```

d. 验证此密钥。

```
kubectl get secret -n kube-system vpc-admission-webhook-certs
```

e. 配置 Webhook 并创建部署文件。

```
cat ./vpc-admission-webhook-deployment.yaml | ./webhook-patch-ca-bundle.sh >
 vpc-admission-webhook.yaml
```

3. 部署 VPC 准入 Webhook。

```
kubectl apply -f vpc-admission-webhook.yaml
```

Important

VPC 准入控制器 Webhook 使用证书签名，该证书在签发后具有一年有效期。为避免停机，请确保在该证书到期之前更新证书。有关更多信息，请参阅更新 VPC 准入 Webhook 证书 (p. 55)。

4. 确定您的集群是否具有所需的集群角色绑定。

```
kubectl get clusterrolebinding eks:kube-proxy-windows
```

如果返回的输出类似于以下示例输出，则集群具有必要的角色绑定。

```
NAME                     ROLE                                AGE
eks:kube-proxy-windows   ClusterRole/system:node-proxier   19h
```

如果输出包含 `Error from server (NotFound)`，则集群没有必要的集群角色绑定。通过创建一个名为 *eks-kube-proxy-windows-crb.yaml* 且包含以下内容的文件来添加绑定。

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: eks:kube-proxy-windows
  labels:
    k8s-app: kube-proxy
    eks.amazonaws.com/component: kube-proxy
subjects:
  - kind: Group
    name: "eks:kube-proxy-windows"
roleRef:
  kind: ClusterRole
  name: system:node-proxier
  apiGroup: rbac.authorization.k8s.io
```

将配置应用于集群。

```
kubectl apply -f eks-kube-proxy-windows-crb.yaml
```

5. 启用 Windows 支持后，您可以在集群中启动 Windows 节点组。有关更多信息，请参阅启动自行管理的 Windows 节点 (p. 114)。

更新 VPC 准入 Webhook 证书

VPC 准入 Webhook 使用的证书将在签发一年后过期。为了避免停机，请务必在证书过期之前更新该证书。您可以使用以下命令检查当前证书的到期日期。

```
kubectl get secret \
    -n kube-system \
    vpc-admission-webhook-certs -o json | \
    jq -r '.data."cert.pem"' | \
    base64 -decode | \
    openssl x509 \
    -noout \
    -enddate | \
    cut -d= -f2
```

输出示例如下。

```
May 28 14:23:00 2022 GMT
```

您可以使用 `eksctl` 或者 Windows 或 Linux/macOS 计算机更新证书。按照您最初用于安装 VPC 准入 Webhook 的工具的说明进行操作。例如，如果您最初安装 VPC 准入 Webhook 使用的是 `eksctl`，那么您应该根据 `eksctl` 选项卡上的说明来更新证书。

eksctl

1. 重新安装证书。将 *my-cluster* 替换为您的集群名称。

   ```
   eksctl utils install-vpc-controllers -cluster my-cluster -approve
   ```

2. 验证您是否收到了以下输出。

   ```
   2021/05/28 05:24:59 [INFO] generate received request
   2021/05/28 05:24:59 [INFO] received CSR
   2021/05/28 05:24:59 [INFO] generating key: rsa-2048
   2021/05/28 05:24:59 [INFO] encoded CSR
   ```

3. 重新启动 Webhook 部署。

   ```
   kubectl rollout restart deployment -n kube-system vpc-admission-webhook
   ```

4. 如果您更新的证书已过期，并且有 Windows pods 卡在 `Container creating` 状态，则必须删除这些 pods 并重新部署。

Windows

1. 获取用于生成新证书的脚本。

   ```
   curl -o webhook-create-signed-cert.ps1 https://s3.cn-north-1.amazonaws.com.cn/
   amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-
   cert.ps1;
   ```

2. 为脚本准备参数。

   ```
   ./webhook-create-signed-cert.ps1 -ServiceName vpc-admission-webhook-svc -SecretName
    vpc-admission-webhook-certs -Namespace kube-system
   ```

3. 重新启动 Webhook 部署。

```
kubectl rollout restart deployment -n kube-system vpc-admission-webhook-deployment
```

4. 如果您更新的证书已过期，并且有 Windows pods 卡在 `Container creating` 状态，则必须删除这些 pods 并重新部署。

Linux and macOS

先决条件

您的计算机上必须已安装了 OpenSSL 和 jq。

1. 获取用于生成新证书的脚本。

```
curl -o webhook-create-signed-cert.sh \
    https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.sh
```

2. 更改权限。

```
chmod +x webhook-create-signed-cert.sh
```

3. 运行脚本。

```
./webhook-create-signed-cert.sh
```

4. 重启 Webhook。

```
kubectl rollout restart deployment -n kube-system vpc-admission-webhook-deployment
```

5. 如果您更新的证书已过期，并且有 Windows pods 卡在 `Container creating` 状态，则必须删除这些 pods 并重新部署。

私有集群要求

本主题介绍如何在没有出站 Internet 访问的情况下部署 Amazon EKS 私有集群。如果您不熟悉 Amazon EKS 联网，请参阅 Amazon EKS 工作线程节点的集群联网解密。

要求

必须满足以下要求才能在无出站 Internet 访问的私有集群中运行 Amazon EKS。

- 容器镜像必须位于或者复制到 Amazon Elastic Container Registry (Amazon ECR) 中，或复制到要拉取的 VPC 内部的注册表中。有关更多信息，请参阅创建容器映像的本地副本 (p. 58)。
- 节点需要端点私有访问权限才能注册到集群端点。终端节点公有访问权限是可选的。有关更多信息，请参阅Amazon EKS 集群端点访问控制 (p. 38)。
- 对于 Linux 和 Windows 节点，启动自行管理的节点时，必须包含引导参数。此文本绕过 Amazon EKS 自检，不需要从 VPC 内访问 Amazon EKS API。将 *api-server-endpoint* 和 *certificate-authority* 替换为 Amazon EKS 集群中的值。
  - 对于 Linux 节点：

```
--apiserver-endpoint api-server-endpoint --b64-cluster-ca certificate-authority
```

有关更多参数，请参阅 GitHub 上的引导脚本。

- 对于 Windows 节点：

```
-APIServerEndpoint api-server-endpoint -Base64ClusterCA certificate-authority
```

有关更多参数，请参阅 Amazon EKS 优化版 Windows AMI (p. 188)。

- 必须从 VPC 内创建 `aws-auth` ConfigMap。有关创建 `aws-auth` ConfigMap 的更多信息，请参阅让 IAM 用户和角色有权访问您的集群 (p. 355)。

注意事项

以下是在没有出站 Internet 访问的私有集群中运行 Amazon EKS 时需要考虑的一些事项。

- 许多 Amazon 服务都支持私有集群，但您必须使用 VPC 终端节点。有关更多信息，请参阅 VPC 终端节点。一些常用的服务和终端节点包括：

服务	端点
Amazon Elastic Container Registry	`cn.com.amazonaws.region-code.ecr.api` 和 `cn.com.amazonaws.region-code.ecr.dkr` 以及 Amazon S3 网关端点
Application Load Balancer 和 Network Load Balancer	`cn.com.amazonaws.region-code.elasticloadbalancing`
Amazon X-Ray	`cn.com.amazonaws.region-code.xray`
Amazon CloudWatch Logs	`cn.com.amazonaws.region-code.logs`
服务账户的 IAM 角色 (p. 395)	`cn.com.amazonaws.region-code.sts`
App Mesh • 支持适用于 Kubernetes 的 App Mesh 附加注入器。有关更多信息，请参阅 GitHub 上的 App Mesh 附加注入器。 • 不支持适用于 Kubernetes 的 App Mesh 控制器。有关更多信息，请参阅 GitHub 上的 App Mesh 控制器。 • 支持Cluster Autoscaler (p. 72)。在部署 Cluster Autoscaler pods 时，请确保命令行包括 `--aws-use-static-instance-list=true`。有关更多信息，请参阅 GitHub 上的使用静态实例列表。Worker 节点 VPC 还必须包括 STS VPC 终端节点和弹性伸缩 VPC 终端节点。	`cn.com.amazonaws.region-code.appmesh-envoy-management`

- 在部署 Amazon EFS CSI 驱动程序 (p. 210) 之前，必须更改 kustomization.yaml 文件，以将容器镜像设置为使用与 Amazon EKS 集群相同的 Amazon Web Services 区域。
- 自行管理和托管的节点 (p. 106)受支持。节点的实例必须具有对 VPC 端点的访问权限。如果您创建托管节点组，则 VPC 端点安全组必须允许子网的 CIDR，或者您必须将创建的节点安全组添加到 VPC 端点安全组。
- Amazon FSx for Lustre CSI 驱动程序 (p. 220) 不受支持。

- 一些容器软件产品使用 API 调用来访问监控使用情况的 Amazon Marketplace Metering 计量服务。私有集群不允许这些调用，因此这些容器类型不能用于私有集群。

创建容器映像的本地副本

由于私有集群没有出站 Internet 访问，因此无法从 Docker Hub 等外部源提取容器映像。相反，容器镜像必须本地复制到 Amazon ECR 或者复制到在 VPC 中可访问的备用注册表。容器镜像可以从私有 VPC 外部复制到 Amazon ECR。私有集群使用 Amazon ECR VPC 端点访问 Amazon ECR 存储库。您必须在用于创建本地副本的工作站上安装 Docker 和 Amazon CLI。

创建容器映像的本地副本

1. 创建 Amazon ECR 存储库。有关更多信息，请参阅创建存储库。
2. 使用 `docker pull` 从外部注册表中提取容器映像。
3. 通过 `docker tag`，使用 Amazon ECR 注册表、存储库和可选镜像标签名称组合标记您的镜像。
4. 对注册表进行身份验证。有关更多信息，请参阅注册表身份验证。
5. 使用 `docker push` 将镜像推送到 Amazon ECR。

   Note

   请确保更新资源配置以使用新映像位置。

   以下示例展示使用标签 `v1.3.1-linux-amd64` 从 Docker Hub 拉取 amazon/aws-node-termination-handler 映像以及在 Amazon ECR 中创建本地副本。

```
aws ecr create-repository --repository-name amazon/aws-node-termination-handler
docker pull amazon/aws-node-termination-handler:v1.3.1-linux-amd64
docker tag amazon/aws-node-termination-handler 111122223333.dkr.ecr.region-
code.amazonaws.com.cn/amazon/aws-node-termination-handler:v1.3.1-linux-amd64
aws ecr get-login-password --region region-code | docker login --username Amazon --
password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com.cn
docker push 111122223333.dkr.ecr.region-code.amazonaws.com.cn/amazon/aws-node-
termination-handler:v1.3.1-linux-amd64
```

用于服务账户 IAM 角色的 Amazon STS 端点

配置了服务账户 IAM 角色 (p. 395)的 Pod 通过 Amazon Security Token Service (Amazon STS) API 调用获取凭证。如果没有出站 Internet 访问，则您必须在 VPC 中创建并使用 Amazon STS VPC 终端节点。预设情况下，大多数 Amazon v1 开发工具包均使用全局 Amazon STS 端点（`sts.amazonaws.com`），而此端点不使用 Amazon STS VPC 端点。要使用 Amazon STS VPC 终端节点，您可能需要将此 SDK 配置为使用区域 Amazon STS 端点 (`sts.region-code.amazonaws.com`)。为此，可以设置值为 `regional` 的 `AWS_STS_REGIONAL_ENDPOINTS` 环境变量，以及 Amazon Web Services 区域。

例如，在 pod 规范中：

```
...
containers:
- env:
- name: AWS_REGION
value: region-code
    - name: AWS_STS_REGIONAL_ENDPOINTS
    value: regional
    ...
    ```
```
```

将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

Amazon EKS Kubernetes 版本

该 Kubernetes 项目持续整合新功能、设计更新和错误修复。社区发布了新的 Kubernetes 次要版本，例如 1.23。平均每三个月提供一次新版本更新。每个次要版本在首次发布后约 12 个月内受支持。

可用的 Amazon EKS Kubernetes 版本

以下 Kubernetes 版本目前可用于新的 Amazon EKS 集群：

- 1.23.7
- 1.22.10
- 1.21.13
- 1.20.15
- 1.19.16

如果您的应用程序需要一个特定版本的 Kubernetes，我们建议您为您的集群使用 Amazon EKS 支持的最新可用 Kubernetes 版本。随着新 Kubernetes 版本在 Amazon EKS 中提供，我们建议您主动将集群更新为使用最新的可用版本。有关如何更新集群的说明，请参阅 更新 Amazon EKS 集群 Kubernetes 版本 (p. 28)。有关 Kubernetes 版本的更多信息，请参阅 Amazon EKS Kubernetes 发布日历 (p. 65) 和 Amazon EKS 版本支持和常见问题 (p. 65)。

Note

从 Kubernetes 版本 1.24 开始，正式发布的 Amazon EKS AMI 将包括 containerd 作为唯一的运行时。Kubernetes 版本 1.18–1.23 将 Docker 用作默认运行时。但是，这些版本有一个引导标志选项，您可以使用该选项在任何受支持的集群上通过 containerd 测试工作负载。有关更多信息，请参阅Amazon EKS 即将结束对 Dockershim 的支持 (p. 147)。

Kubernetes 1.23

Kubernetes 1.23 现已在 Amazon EKS 中推出。有关 Kubernetes 1.23 的更多信息，请参阅官方发布公告。

- Important

  Kubernetes 树内插件至 Container Storage Interface (CSI) 卷的迁移功能已启用。此功能支持使用相应的 Amazon EBS CSI 驱动程序替换适用于 Amazon EBS 的现有 Kubernetes 树内存储插件。有关更多信息，请参阅 Kubernetes 博客上的 Kubernetes 1.17 Feature: Kubernetes In-Tree to CSI Volume Migration Moves to Beta（1.17 功能：树内插件至 CSI 卷的迁移功能推出测试版）。
  该功能将树内 API 转换为等效的 CSI API，并将操作委派给替换 CSI 驱动程序。借助此功能，如果您使用的是属于这些工作负载的现有 StorageClass、PersistentVolume 和 PersistentVolumeClaim 对象，则可能不会有任何明显变化。该功能支持 Kubernetes 将树内插件的所有存储管理操作委派给 CSI 驱动程序。如果您在现有集群中使用 Amazon EBS 卷，请在将集群更新为版本 **1.23** 之前，在集群中安装 Amazon EBS CSI 驱动程序。如果未在更新现有集群之前安装驱动程序，则可能会中断您的工作负载。如果您计划在新 1.23 集群中部署使用 Amazon EBS 卷的工作负载，请在集群中部署工作负载之前，在集群中安装 Amazon EBS CSI 驱动程序。有关如何在集群上安装 Amazon EBS CSI 驱动程序的说明，请参阅 the section called “Amazon EBS CSI 驱动程序” (p. 195)。有关迁移功能的常见问题，请参阅 the section called “CSI 迁移常见问题” (p. 208)。

- Kubernetes 已停止支持版本 1.20 中的 dockershim，并已删除版本 1.24 中的 dockershim。有关更多信息，请参阅 Kubernetes 博客上的 Kubernetes is Moving on From: Commitments and Next Steps（Kubernetes 正在舍弃 Dockershim：承诺和后续步骤）。从 Amazon EKS 版本 1.24 开始，Amazon EKS 将终止对 dockershim 的支持。从 Amazon EKS 版本 containerd 开始，Amazon EKS 官方 AMI 仅会将 1.24 作为唯一运行时。

  尽管 Amazon EKS 版本 1.23 继续支持 dockershim，我们仍建议您立即开始测试自己的应用程序，以识别和删除任何 Docker 依赖项。如此，您就可以做好将集群更新到版本 1.24 的准备。有关删除 dockershim 的更多信息，请参阅 the section called “Dockershim 弃用” (p. 147)。
- Kubernetes 已将 IPv4/IPv6 双堆栈联网功能升级到正式版本，可供 pods、服务和节点使用。但是，Amazon EKS 和 Amazon VPC CNI plugin for Kubernetes 目前不支持双堆栈联网。您的集群可以将 IPv4 或 IPv6 地址分配给 pods 和服务，但不能同时分配两种地址类型。
- Kubernetes 已将容器组（pod）安全准入功能升级到测试版。该功能已默认启用。有关更多信息，请参阅 Kubernetes 文档中的 Pod Security Admission [容器组（pod）安全准入]。容器组（pod）安全准入将取代容器组（pod）安全策略 (PSP) 准入控制器。PSP 准入控制器不受支持，计划在 Kubernetes 版本 1.25 中删除。

  PSP 准入控制器根据设置强制级别的特定命名空间标签，在命名空间中的 pod 上强制实施 pods 安全标准。有关更多信息，请参阅 Amazon EKS 最佳实践指南中的 Pod Security Standards (PSS) and Pod Security Admission (PSA) [容器组（pod）安全标准（PSS）和容器组（pod）安全准入（PSA）]。
- 使用集群部署的 kube-proxy 映像现在是 Amazon EKS Distro（EKS-D）维护的最低要求基本映像。该映像包含最低要求的程序包，并且没有 shell 或程序包管理器。
- Kubernetes 已将临时容器升级到测试版。临时容器是在与现有 pod 相同的命名空间中运行的暂时性容器。您可以用它们来观察 pods 和容器的状态，以便排查问题并进行调试。在 kubectl exec 不足时，这种容器对于交互式排查问题尤其有用，因为容器已崩溃或容器映像不包含调试实用程序。包含调试实用程序的示例容器是 Distroless 映像。有关更多信息，请参阅 Kubernetes 文档中的 Debugging with an ephemeral debug container（使用临时调试容器进行调试）。
- Kubernetes 已将 HorizontalPodAutoscaler autoscaling/v2 稳定 API 升级到正式版本。不再支持 HorizontalPodAutoscaler autoscaling/v2beta2 API。

有关完整 Kubernetes 1.23 的更改日志，请参阅 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.23.md#changelog-since-v1220。

Kubernetes 1.22

Kubernetes 1.22 现已在 Amazon EKS 中推出。有关 Kubernetes 1.22 的更多信息，请参阅官方发布公告。

- Kubernetes 1.22 将删除许多不再可用的 API。在升级到 Amazon EKS 版本 1.22 之前，您可能需要更改应用程序。更新集群前请严格按照 Kubernetes 版本 1.22 的先决条件 (p. 32) 执行。
- Important

  BoundServiceAccountTokenVolume 已升级到稳定版本，且默认情况下在 Kubernetes 版本 1.22 中启用。此功能提高了服务账户令牌的安全性。它允许正在 Kubernetes 上运行的工作负载请求与受众、时间和密钥绑定的 JSON Web 令牌。服务账户令牌现在的有效期为 1 小时。在以前的 Kubernetes 版本中，它们没有有效期。这意味着依赖这些令牌的客户端必须在一小时内刷新令牌。以下 Kubernetes 客户端开发工具包会在要求的时间范围内自动刷新令牌：

  - Go 版本 0.15.7 和更高版本
  - Python 版本 12.0.0 和更高版本
  - Java 版本 9.0.0 和更高版本
  - JavaScript 版本 0.10.3 和更高版本
  - Ruby master 分支
  - Haskell 版本 0.3.0.0

  - C# 版本 7.0.5 和更高版本

    如果您的工作负载使用的是旧版本的客户端，则必须予以更新。为了使客户顺利迁移到更新的有时限的服务账户令牌，Kubernetes 版本 1.22 在默认一小时内向服务账户令牌添加延长的到期期限。对于 Amazon EKS 集群，延长到期期限为 90 天。Amazon EKS 集群的 Kubernetes API 服务器拒绝令牌超过 90 天的请求。我们建议您检查应用程序及其依赖项，以确保 Kubernetes 客户端开发工具包版本等于或高于之前列出的版本。有关如何识别使用过时令牌的 pods 的说明，请参阅 Kubernetes 服务账户 (p. 395)。
- 入口 API 版本 extensions/v1beta1 和 networking.k8s.io/v1beta1 已在 Kubernetes 1.22 中删除。如果您使用的是 Amazon Load Balancer Controller，您必须至少升级到版本 2.4.1才能将 Amazon EKS 集群升级到 1.22 版本。此外，您必须修改入口清单，才能使用 apiVersion networking.k8s.io/v1。此选项在 Kubernetes 版本 1.19 之后可用。有关入口 v1beta1 和 v1 之间的更改的更多信息，请参阅 Kubernetes 文档。Amazon Load Balancer Controller 控制器样本清单使用 v1 规格。
- Amazon EKS 旧版 Windows 支持控制器使用已在 Kubernetes 1.22 中删除的 admissionregistration.k8s.io/v1beta1 API。如果您运行的是 Windows 工作负载，您必须删除旧版 Windows 支持并启用 Windows 支持才能升级到 Amazon EKS 版本 1.22。
- CertificateSigningRequest（CSR）API 版本 certificates.k8s.io/v1beta1 已在 Kubernetes 版本 1.22 中删除。您必须迁移清单和 API 客户端才能使用 certificates.k8s.io/v1 CSR API。此 API 在版本 1.19 之后可用。有关如何在 Amazon EKS 中使用 CSR 的说明，请参阅证书签名 (p. 392)。
- CustomResourceDefinition API 版本 apiextensions.k8s.io/v1beta1 已在 Kubernetes 1.22 中删除。确保集群中的所有自定义资源定义都更新为 v1。需要 API 版本 v1 自定义资源定义才能定义 Open API v3 架构验证。有关更多信息，请参阅 Kubernetes 文档。
- 如果您使用的是 App Mesh，则必须至少升级到 App Mesh 控制器 v1.4.3 或更高版本才能升级到 Amazon EKS 版本 1.22。旧版本的 App Mesh 控制器使用 v1beta1 CustomResourceDefinition API 版本且不兼容 Kubernetes 版本 1.22 和更高版本。
- Amazon EKS 版本 1.22 在默认情况下启用 EndpointSliceTerminatingCondition 功能，其中包括 EndpointSlices 中处于终止状态的 pods。如果您在 Amazon Load Balancer Controller 中将 enableEndpointSlices 设置为 True（默认为禁用），您必须至少升级到 Amazon Load Balancer Controller 版本 2.4.1+ 后才能升级到 Amazon EKS 版本 1.22。
- 从 Amazon EKS 版本 1.22 开始，默认配置了 kube-proxy 来公开 pod 之外的 Prometheus 指标。此行为更改解决了容器路线图问题 #657 中提出的请求。
- Amazon EKS 版本 1.22 的初次发布使用 etcd 版本 3.4 作为后端，不受 etcd 版本 3.5 中存在的数据损坏的可能性的影响。
- 从 Amazon EKS 1.22 开始，Amazon EKS 从核心控制面板代码中解耦特定于 Amazon 云的控制逻辑到树外 Amazon Kubernetes 云控制器管理器。这与上游 Kubernetes 推荐一致。通过解耦 Kubernetes 与底层云基础设施之间的互操作性逻辑，cloud-controller-manager 组件使云提供商能够以与主要 Kubernetes 项目相比不同的速度发布功能。这种变化是透明的，无需采取任何行动。但是，当启用时，名为 cloud-controller-manager 的新日志流现在显示在 ControllerManager 日志类型下。有关更多信息，请参阅 Amazon EKS 控制面板日志记录。
- 从 Amazon EKS 1.22 开始，Amazon EKS 将服务账户的 IAM 角色（IRSA）使用的默认 Amazon Security Token Service 端点更改为区域性端点（而不是全局端点），以减少延迟和提高可靠性。您可以选择将 IRSA 配置为在 the section called “配置 Amazon STS 端点” (p. 405) 中使用全局端点。

在 Kubernetes 1.22 Amazon EKS 集群中，现支持以下 Kubernetes 功能：

- 服务器端应用功能升级到 GA - 服务器端应用功能帮助用户和控制器通过声明式配置管理其资源。它允许他们通过发送完全指定的目的以声明方式创建或修改对象。在几个版本中进入测试版后，服务器端应用功能现已正式发布。
- 使用不受支持的 API 的警告机制 – 使用不受支持的 API 会产生对 API 使用者可见的警告以及对集群管理员可见的指标。

有关完整 Kubernetes 1.22 的更改日志，请参阅 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.22.md#changelog-since-v1210。

Kubernetes 1.21

Kubernetes `1.21` 现已在 Amazon EKS 中推出。有关 Kubernetes `1.21` 的更多信息，请参阅官方发布公告。

- Important

  `BoundServiceAccountTokenVolume` 已升级到测试版，且默认在 Kubernetes 版本 `1.21` 中启用。此功能通过允许在 Kubernetes 上运行工作负载来提高服务账户令牌的安全性，以请求与受众、时间和密钥绑定的 JSON Web 令牌。服务账户令牌现在的有效期为 1 小时。在以前的 Kubernetes 版本中，它们没有有效期。这意味着依赖这些令牌的客户端必须在一小时内刷新令牌。以下 Kubernetes 客户端开发工具包会在要求的时间范围内自动刷新令牌：

  - Go 版本 `0.15.7` 和更高版本
  - Python 版本 `12.0.0` 和更高版本
  - Java 版本 `9.0.0` 和更高版本
  - JavaScript 版本 `0.10.3` 和更高版本
  - Ruby `master` 分支
  - Haskell 版本 `0.3.0.0`
  - C# 版本 `7.0.5` 和更高版本

  如果您的工作负载使用的是旧版本的客户端，则必须予以更新。为了使客户顺利迁移到更新的有时限的服务账户令牌，Kubernetes 版本 `1.21` 在默认一小时内向服务账户令牌添加延长的到期期限。对于 Amazon EKS 集群，延长到期期限为 90 天。Amazon EKS 集群的 Kubernetes API 服务器拒绝令牌超过 90 天的请求。我们建议您检查应用程序及其依赖项，以确保 Kubernetes 客户端开发工具包版本等于或高于之前列出的版本。有关如何识别使用过时令牌的 pods 的说明，请参阅 Kubernetes 服务账户 (p. 395)。

- pods、服务和节点上的双堆栈联网支持（`IPv4` 和 `IPv6` 地址）已达到测试版状态。但是，Amazon EKS 和 Amazon VPC CNI plugin for Kubernetes 目前不支持双堆栈联网。
- Amazon EKS 优化版 Amazon Linux 2 AMI 现在包含一个引导启动标志，以启用 `containerd` 运行时作为 Docker 替代方案。此标志允许为在下一个 Kubernetes 版本中删除作为受支持运行时的 Docker 做准备。有关更多信息，请参阅启用 `containerd` 运行时间引导标记 (p. 157)。这可以通过 Github 上的容器路线图进行跟踪。
- 托管节点组支持 Cluster Autoscaler 优先级扩展器。

  Amazon EKS 版本 `1.21` 集群上新创建的托管节点组对基础自动扩缩组名称使用以下格式：

  `eks-`*`managed-node-group-name`*`-`*`uuid`*

  这样就可以根据用户定义的优先级使用 Cluster Autoscaler 的优先级扩展器功能。一个常见的使用案例是，将 Spot 节点组而不是按需节点组作为扩缩首选。此行为更改解决了 #1304 容器路线图问题。

在 Amazon EKS `1.21` 集群中，现支持以下 Kubernetes 功能：

- CronJobs（前称为 ScheduledJobs）现已升级到稳定状态。此项更改后，用户定期执行计划的操作，例如备份和报告生成。
- 不可改变的密钥和 ConfigMaps 现已升级到稳定状态。已向这些对象添加了一个不可改变的新字段以拒绝更改。此拒绝可保护集群免受会意外中断应用程序的更新。由于这些资源是不可改变的，因此 `kubelet` 不会关注或者轮询更改。这将减少 `kube-apiserver` 负载并提高可扩展性和性能。
- 正常节点关机功能现已升级到测试版状态。此更新允许 `kubelet` 知道节点关闭，并能够正常终止该节点的 pods。在此更新之前，当节点关闭时，它的 pods 没有遵循预期的终止生命周期。这导致了工作负载问题。现在，`kubelet` 可以通过 `systemd` 检测即将发生的系统关闭，并通知正在运行的 pods，以便它们正常终止。
- 具有多个容器的容器组（pod）现在可以使用 `kubectl.kubernetes.io/default-container` 注释为 `kubectl` 命令预先选择容器。

- `PodSecurityPolicy` 正被逐步淘汰。根据 `PodSecurityPolicy` 弃用指南，Kubernetes 仍可用于多个版本。要了解详情，请参阅 PodSecurityPolicy 弃用：过去、现在和将来和 Amazon 博客。

有关完整 Kubernetes `1.21` 的更改日志，请参阅 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.21.md。

Kubernetes 1.20

有关 Kubernetes `1.20` 的更多信息，请参阅官方发布公告。

- `1.20` 版本带来了新的原定设置角色和用户。您可以在原定设置 EKS Kubernetes 角色和用户中了解更多信息。请确保您使用的是支持的 cert-manager 版本。

在 Kubernetes `1.20` Amazon EKS 集群中，现支持以下 Kubernetes 功能：

- API 优先级和公平性功能已达到测试状态，并且在预设情况下是处于已启用状态。此功能允许 `kube-apiserver` 按优先级对传入请求进行分类。
- RuntimeClass 功能已达到稳定状态。`RuntimeClass` 资源提供了一种用于支持集群中多个运行时间的机制，并将有关该容器运行时的信息显示到控制层面。
- 进程 ID 限制功能现已升级到正式发布版本。
- kubectl debug 已达到测试版状态。`kubectl debug` 直接支持来自 `kubectl` 的常见调试工作流。
- Docker 容器运行时已逐步停止。Kubernetes 社群已经就这一点撰写了一篇详细的博客文章并提供了专门的常见问题页面。Docker 生成的镜像可以继续使用，并且会一如既往地发挥功能。您可以安全地忽略 `kubelet` 启动日志中打印的 `dockershim` 弃用警告消息。Amazon EKS 最终将迁移到 `containerd`，后者将作为 Amazon EKS 优化版 Amazon Linux 2 AMI 的运行时间。您可以参阅容器路线图问题，了解更多详细信息。
- Pod Hostname as FQDN 功能已升级到测试版状态。此功能允许将 pod 的主机名设置为其完全限定域名（FQDN），从而能够将内核的主机名字段设置为 pod 的 FQDN。
- client-go 凭证插件现在可以通过 `KUBERNETES_EXEC_INFO` 环境变量在当前集群信息中传递。此增强功能允许 Go 客户端使用密钥管理系统（KMS）等外部凭证提供程序进行身份验证。

有关完整 Kubernetes `1.20` 的更改日志，请参阅 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.20.md。

Kubernetes 1.19

有关 Kubernetes `1.19` 的更多信息，请参阅官方发布公告。

- 从 `1.19` 版本开始，Amazon EKS 不再向在集群创建时传入的子网添加 `kubernetes.io/cluster/`*`my-cluster`* 标签。仅当您想影响 Kubernetes 服务控制器或 Amazon Load Balancer Controller 放置 Elastic Load Balancer 的位置时，才需要此子网标签。有关在集群创建过程中传递给 Amazon EKS 的子网的要求的更多信息，请参阅 the section called “VPC 和子网要求” (p. 226)。
  - 在更新为 `1.19` 的现有集群上，子网标签未修改。
  - Amazon Load Balancer Controller 版本以及 `2.1.1` 和更早版本需要 *`my-cluster`* 子网标签。在版本 `2.1.2` 和更高版本中，您可以指定标签以优化子网发现，但这不是必需操作。有关 Amazon Load Balancer Controller 的更多信息，请参阅 安装 Amazon Load Balancer Controller 附加组件 (p. 287)。有关使用负载均衡器时子网标签的更多信息，请参阅 Amazon EKS 上的应用程序负载均衡 (p. 332) 和 Amazon EKS 上的网络负载均衡 (p. 327)。
- 您不再需要为必须访问 Web 身份令牌文件以便与服务账户的 IAM 角色一起使用的无根容器提供安全上下文。有关更多信息，请参阅 GitHub 上的 服务账户的 IAM 角色 (p. 395) 和预计服务账户卷中的文件权限处理建议。

- pod 身份 Webhook 已更新，以解决缺少启动探测器的 GitHub 问题。Webhook 现在还支持控制令牌过期的注释。有关更多信息，请参阅 GitHub 拉取请求。
- CoreDNS 版本 `1.8.0` 是推荐用于 Amazon EKS `1.19` 集群的版本。预设情况下，此版本已安装在新的 Amazon EKS `1.19` 集群中。有关更多信息，请参阅管理 CoreDNS 附加组件 (p. 294)。
- Amazon EKS 优化版 Amazon Linux 2 AMI 包括适用于 Kubernetes 版本 `1.19` 的 Linux 内核版本 `5.4`。有关更多信息，请参阅 GitHub 上的更改日志。有关更多信息，请参阅 ???。
- `CertificateSigningRequest API` 已被提升为稳定 `certificates.k8s.io/v1` 并附带以下更改：
  - 现在需要使用 `spec.signerName`。您无法为具有 `certificates.k8s.io/v1` API 的 `kubernetes.io/legacy-unknown` 创建请求。
  - 您可以继续使用 `certificates.k8s.io/v1beta1` API 创建具有 `kubernetes.io/legacy-unknown` 签名者名称的 CSR。
  - 您可以继续请求，为非节点服务器证书 webhook（例如 `certificates.k8s.io/v1beta1` API）签署 CSR。这些 CSR 不是自动批准的。
  - 要批准证书，特权用户需要 `kubectl 1.18.8` 或更高版本。

  有关证书 `v1` API 的更多详细信息，请参阅 Kubernetes 文档中的证书签名请求。

以下 Amazon EKS Kubernetes 资源对于 Kubernetes 控制面板执行工作至关重要。建议您不要删除或编辑它们。

权限	Kind	命名空间	Reason
`eks:certificate-controller`	`Rolebinding`	`kube-system`	影响控制层面中的签名者和批准者功能。
`eks:certificate-controller`	`Role`	`kube-system`	影响控制层面中的签名者和批准者功能。
`eks:certificate-controller`	`ClusterRolebinding`	全部	影响 `kubelet` 请求服务器证书的能力，这会影响某些集群功能，如 `kubectl exec` 和 `kubectl logs`。

在 Kubernetes `1.19` Amazon EKS 集群中，现支持以下 Kubernetes 功能：

- `ExtendedResourceToleration` 准入控制器已启用。此准入控制器会自动向请求扩展资源（例如 GPU）的 pods 添加污点容忍度。这样，您无需手动添加容忍度。有关更多信息，请参阅 Kubernetes 文档中的 ExtendedResourceToleration。
- 由树内 Kubernetes 服务控制器预置的 Elastic Load Balancers（CLB 和 NLB）支持筛选作为实例目标而纳入的节点。这有助于防止大型集群中达到目标组限制。有关更多信息，请参阅相关 GitHub 问题和 Kubernetes 文档中其他 ELB 注释下的 `service.beta.kubernetes.io/aws-load-balancer-target-node-labels` 注释。
- Pod 拓扑分布已达到稳定状态。您可以使用拓扑分布约束来控制 pods 在集群内故障域（如区域、可用区、节点和其他用户定义的拓扑域）间的分布方式。这有助于实现高可用性以及高效的资源利用率。有关更多信息，请参阅 Kubernetes 文档中的容器组（pod）拓扑分布约束。
- Ingress API 已达到正式发布版本。有关更多信息，请参阅 Kubernetes 文档中的入口。
- 预设情况下，EndpointSlices 为已启用状态。EndpointSlices 是一个新的 API，它为 Endpoints API 提供了更加可扩展和可扩充的替代方案，用于跟踪为服务提供支持的 Pod 的 IP 地址、端口、就绪性和拓扑信息。有关更多信息，请参阅 Kubernetes 博客中的使用 EndpointSlices 扩展 Kubernetes 联网。
- Secret 和 ConfigMap 卷现在可以标记为不可改变。如果集群中存在许多 Secret 卷和 ConfigMap 卷，这将显著减少 API 服务器上的负载。有关更多信息，请参阅 Kubernetes 文档中的 ConfigMap 和密钥。

有关完整 Kubernetes `1.19` 的更改日志，请参阅 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.19.md。

Kubernetes 1.18

有关 Kubernetes `1.18` 的更多信息，请参阅官方发布公告。

在 Kubernetes `1.18` Amazon EKS 集群中，现支持以下 Kubernetes 功能：

- 拓扑管理器现已达到测试版本状态。此功能允许 CPU 和设备管理器协调资源分配决策，通过机器学习和分析工作负载进行优化，从而实现低延迟。有关更多信息，请参阅 Kubernetes 文档中的控制节点上的拓扑管理策略。
- 服务器端应用功能将更新为新的测试版。此功能可以跟踪和管理对所有新 Kubernetes 对象的字段的更改。这有助于您知道是什么更改了您的资源以及更改时间。有关更多信息，请参阅 Kubernetes 文档中的什么是服务器端应用功能？。
- 入口规范中添加了一个新的 `pathType` 字段和新的 `IngressClass` 资源。这些功能使自定义入口配置变得更加简单，并且受 Amazon Load Balancer Controller (p. 332)（以前称为 ALB 入口控制器）支持。有关更多信息，请参阅 Kubernetes 文档中 Kubernetes`1.18` 中的入口 API 的改进。
- 可配置的水平 pod 自动缩放行为。有关更多信息，请参阅 Kubernetes 文档中的可配置的扩展行为支持。
- 在对服务账户使用 IAM 角色时，无论是使用变异 Webhook 还是手动配置环境变量，您都不再需要将 `AWS_DEFAULT_REGION=`*`region-code`* 环境变量纳入 pods。
- 新集群包含 `externalTrafficPolicy` 中更新的默认值。`HealthyThresholdCount` 和 `UnhealthyThresholdCount` 每个都为 2，而且 `HealthCheckIntervalSeconds` 减少到 10 秒。在旧版本中创建并升级的集群将保留旧的值。

有关完整 Kubernetes `1.18` 的更改日志，请参阅 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.18.md。

Amazon EKS Kubernetes 发布日历

Note

只有月份和年份的发布日期均为大概日期，这些日期会在确切日期明确后进行更新。

Kubernetes 版本	上游版本	Amazon EKS 版本	Amazon EKS 支持终止
`1.24`	2022 年 5 月 3 日	2022 年 11 月	2024 年 1 月
`1.23`	2021 年 12 月 7 日	2022 年 8 月 11 日	2023 年 10 月
`1.22`	2021 年 8 月 4 日	2022 年 4 月 22 日	2023 年 5 月
`1.21`	2021 年 4 月 8 日	2021 年 7 月 19 日	2023 年 2 月
`1.20`	2020 年 12 月 8 日	2021 年 5 月 18 日	2022 年 11 月 1 日
`1.19`	2020 年 8 月 26 日	2021 年 2 月 16 日	2022 年 8 月 1 日
`1.18`	2020 年 3 月 23 日	2020 年 10 月 13 日	2022 年 3 月 31 日

Amazon EKS 版本支持和常见问题

为了与 Kubernetes 社群对 Kubernetes 版本的支持保持一致，Amazon EKS 致力于在任何特定时间支持至少四个生产就绪的 Kubernetes 版本。我们会在支持结束之日前至少 60 天公告特定 Kubernetes 次要版

本的支持结束日期。鉴于新 Kubernetes 版本的 Amazon EKS 鉴定和发布流程，Amazon EKS 上对某个 Kubernetes 版本的支持的结束日期将会是 Kubernetes 项目停止支持上游版本之日或之后。

常见问题

问：Amazon EKS 对某个 Kubernetes 版本的支持持续时间是多久？

答：一个 Kubernetes 版本首次在 Amazon EKS 上发布后在 14 个月内受支持。即使上游 Kubernetes 不再支持 Amazon EKS 上可用的版本，也是如此。我们向后移植适用于 Amazon EKS 上支持的 Kubernetes 版本的安全补丁。

问：当对 Amazon EKS 上的 Kubernetes 版本的支持结束时，我是否会收到通知？

答：是的，如果您账户中的任何集群正在运行支持即将结束的版本，Amazon EKS 会在 Kubernetes 版本在 Amazon EKS 上发布大约 12 个月后通过 Amazon Health Dashboard 发布通知。该通知包括支持结束日期。为从通知发出之日起至少 60 天。

问：支持结束之日会发生什么？

答：在支持结束之日，您将无法再使用不受支持的版本创建新的 Amazon EKS 集群。Amazon EKS 会在支持结束日期后通过逐步部署流程，自动将现有控制层面更新为最早的受支持版本。控制面板自动更新后，确保手动更新集群附加组件和 Amazon EC2 节点。有关更多信息，请参阅the section called “更新 Amazon EKS 集群的 Kubernetes 版本 ” (p. 28)。

问：在支持结束日期后，自动更新控制面板的确切时间是何时？

答：Amazon EKS 无法给出具体的时间范围。自动更新可以在支持结束日期后的任何时间进行。在更新之前，您不会收到任何通知。我们建议您主动更新您的控制面板，而不要依赖 Amazon EKS 自动更新流程。有关更多信息，请参阅the section called “更新 Kubernetes 版本” (p. 28)。

问：我能否无限期地将控制面板留在某个 Kubernetes 版本上？

答：不能，Amazon 的云安全性具有最高优先级。过去一段时间（通常为 1 年），Kubernetes 社群停止发布常见漏洞和风险（CVE）补丁程序，也不鼓励提交不受支持版本的 CVE。这意味着旧版本 Kubernetes 的特定漏洞甚至可能没有报告。这会在出现漏洞的情况下，集群被暴露且不会进行通知，也没有修复选项。鉴于此，Amazon EKS 不允许控制面板停留在支持已经结束的版本上。

问：Amazon EKS 支持哪些 Kubernetes 功能？

答：Amazon EKS 支持 Kubernetes API 的所有正式发布功能。它还支持所有测试版功能，这些功能在默认情况下启用。不支持 Alpha 功能。

问：Amazon EKS 托管节点组是否会随集群控制层面版本一起自动更新？

答：否，托管节点组在您的账户中创建 Amazon EC2 实例。当您或 Amazon EKS 更新控制层面时，这些实例不会自动升级。假设 Amazon EKS 会自动更新您的控制面板。您的托管节点组上的 Kubernetes 版本可能会比控制面板早一个版本以上。然后，假设托管节点组包含运行比控制面板早一个版本以上的 Kubernetes 版本的实例。节点组在控制台中集群的 Compute（计算）选项卡的 Node Groups（节点组）部分中存在运行状况问题。最后，如果节点组具有可用版本更新，则控制台中节点组旁边会显示 Update now（立即更新）。有关更多信息，请参阅the section called “更新托管节点组” (p. 95)。我们建议您的控制面板和节点上保持相同的 Kubernetes 版本。

问：自行管理的节点组是否会随集群控制层面版本一起自动更新？

答：否，自行管理的节点组包括您的账户中的 Amazon EC2 实例。当您或 Amazon EKS 代表您更新控制面板版本时，这些实例不会自动升级。对于自行管理的节点组，控制台中没有任何其需要更新的标示。您可以在集群的 Overview（概览）选项卡上的 Nodes（节点）列表中选择节点查看其上安装的 `kubelet` 版本，确定哪些节点需要更新。您必须手动更新节点。有关更多信息，请参阅the section called “更新” (p. 119)。

Kubernetes 项目最多测试两个次要版本的控制面板和节点之间的兼容性。例如，当 1.21 节点由 1.23 控制面板编排时，节点会继续运行。但是，不建议运行具有版本持续落后控制面板两个次要版本的节点的集群。有关更多信息，请参阅 Kubernetes 文档中的 Kubernetes 版本和版本倾斜支持政策。我们建议您的控制面板和节点上保持相同的 Kubernetes 版本。

问：Fargate 上运行的 pods 是否通过自动集群控制面板版本升级而自动升级？

是的，Fargate pods 在责任共担模式 (p. 392)的 Amazon EKS 侧上的 Amazon 拥有账户中的基础设施中运行。Amazon EKS 使用 Kubernetes 驱逐 API 尝试正常耗尽在 Fargate 上运行的 pods。有关更多信息，请参阅 Kubernetes 文档中的驱逐 API。如果无法将 pod 驱逐，则 Amazon EKS 会发出一个 Kubernetes `delete pod` 命令。我们强烈建议将 Fargate pods 作为复制控制器（如 Kubernetes 部署）的一部分运行。这是为了在删除后，自动重新安排 pod。有关更多信息，请参阅 Kubernetes 文档中的部署。新版本的 Fargate pod 部署有一个 `kubelet` 版本，该版本与更新的集群控制面板版本相同。

> Important
>
> 如果您更新控制面板，则必须自行更新 Fargate 节点。要更新 Fargate 节点，请删除该节点表示的 Fargate pod，然后重新部署 pod。部署新的 pod 时使用 `kubelet` 版本，该版本与您的集群版本相同。

Amazon EKS 平台版本

Amazon EKS 平台版本表示 Amazon EKS 集群控制面板的功能，例如已启用哪些 Kubernetes API 服务器标志以及当前的 Kubernetes 补丁版本。每个 Kubernetes 次要版本都有一个或多个关联的 Amazon EKS 平台版本。不同 Kubernetes 次要版本的平台版本是独立的。

当 Amazon EKS 中有新的 Kubernetes 次要版本可用时，例如 1.23，则该 Kubernetes 次要版本的初始 Amazon EKS 平台版本从 `eks.1` 开始。不过，Amazon EKS 会定期发布新平台版本来支持新的 Kubernetes 控制面板设置并提供安全修复。

当次要版本的新 Amazon EKS 平台版本可用时：

- Amazon EKS 平台版本号将递增 (`eks.`*`n+1`*)。
- Amazon EKS 会自动将所有现有集群升级到其对应的 Kubernetes 次要版本的最新 Amazon EKS 平台版本。现有 Amazon EKS 平台版本的自动升级将增量推出。推出过程可能需要一段时间。如果您即刻就需要最新的 Amazon EKS 平台版本功能，您应创建新的 Amazon EKS 集群。

  如果您的集群比当前平台版本落后两个以上的平台版本，那么 Amazon EKS 可能无法自动更新您的集群。有关可能导致这种情况的原因的详细信息，请参阅 the section called “旧平台版本” (p. 485)。
- Amazon EKS 可能会发布具有对应的修补程序版本的新工作线程 AMI。但是，所有补丁版本在指定 Kubernetes 次要版本的 EKS 控制面板与节点 AMI 之间兼容。

新 Amazon EKS 平台版本不会引发破坏性的更改或导致服务中断。

集群始终使用指定 Kubernetes 版本的最新可用 Amazon EKS 平台版本 (`eks.`*`n`*) 创建。如果您将集群更新为新的 Kubernetes 次要版本，您的集群将接收所更新到的 Kubernetes 次要版本的当前 Amazon EKS 平台版本。

下列各表列出了当前和最新的 Amazon EKS 平台版本。

Kubernetes 版本 1.23

所有 1.23 平台版本都启用了以下准入控制器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIngressC

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.23.7	eks.1	面向 Amazon EKS 的 Kubernetes 版本 1.23 的初始版本。有关更多信息，请参阅Kubernetes 1.23 (p. 59)。	2022 年 8 月 11 日

Kubernetes 版本 1.22

所有 1.22 平台版本都启用了以下准入控制器：DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionWebhook、Na

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.22.10	eks.5	具有改进 etcd 弹性的新平台版本。	2022 年 8 月 15 日
1.22.10	eks.4	新的平台版本（包括安全修复和增强功能）。此平台版本还引入了一个新的标记控制器，该控制器使用 aws:eks:cluster-name 标记所有 Worker 节点，以便为这些 Worker 节点分配成本。有关更多信息，请参阅标记资源以便于计费 (p. 387)。	2022 年 7 月 21 日
1.22.10	eks.3	新的平台版本（包括安全修复和增强功能）。	2022 年 7 月 7 日
1.22.9	eks.2	新的平台版本（包括安全修复和增强功能）。	2022 年 5 月 31 日
1.22.6	eks.1	面向 Amazon EKS 的 Kubernetes 版本 1.22 的初始版本。有关更多信息，请参阅Kubernetes 1.22 (p. 60)。	2022 年 4 月 22 日

Kubernetes 版本 1.21

所有 1.21 平台版本都启用了以下准入控制器：DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionWebhook、Na

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.21.13	eks.10	具有改进 etcd 弹性的新平台版本。	2022 年 8 月 15 日
1.21.13	eks.9	新的平台版本（包括安全修复和增强功能）。此平台版本还引入了一个新的标记控制器，该控制器使用 aws:eks:cluster-name 标记所有 Worker 节点，以便为这些 Worker 节点分配成本。有关更多信息，请参阅标记资源以便于计费 (p. 387)。	2022 年 7 月 21 日

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.21.13	eks.8	新的平台版本（包括安全修复和增强功能）。	2022 年 7 月 7 日
1.21.12	eks.7	新的平台版本（包括安全修复和增强功能）。	2022 年 5 月 31 日
1.21.9	eks.6	Amazon Security Token Service 端点将从以前的平台版本还原到全局端点。如果您想在为服务账户使用 IAM 角色时使用区域端点，则必须将其启用。有关如何启用区域端点的说明，请参阅 配置服务账户的 Amazon Security Token Service 端点 (p. 405)。	2022 年 4 月 8 日
1.21.5	eks.5	使用 服务账户的 IAM 角色 (p. 395) 时，现在将默认使用 Amazon Security Token Service 区域端点，而不是全局端点。但是此更改将恢复到 eks.6 中的全局端点。 更新后的 Fargate 调度程序在大规模部署期间以更高的速度预置节点。	2022 年 3 月 10 日
1.21.5	eks.4	1.10.1-eksbuild.1 版的 Amazon VPC CNI 自主托管型 Amazon EKS 附加组件现在是原定设置部署的版本。	
1.21.2	eks.3	这是新推出的平台版本，支持在 Kubernetes 控制面板上运行的 VPC 资源控制器上进行 Windows IPv4 地址管理。增加了用于 Fargate Fluent Bit 日志记录的 Kubernetes 筛选指令。	
1.21.2	eks.2	新的平台版本（包括安全修复和增强功能）。	
1.21.2	eks.1	面向 Amazon EKS 的 Kubernetes 版本 1.21 的初始版本。有关更多信息，请参阅Kubernetes 1.21 (p. 62)。	

Kubernetes 版本 1.20

所有 1.20 平台版本都启用了以下准入控制器：DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionWebhook、Na

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.20.15	eks.8	具有改进 etcd 弹性的新平台版本。	2022 年 8 月 15 日
1.20.15	eks.7	新的平台版本（包括安全修复和增强功能）。此平台版本还引入了一个新的标记控制器，该控制器使用 aws:eks:cluster-name 标记所有 Worker 节点，以便为这些 Worker 节点分配成本。有关更多信息，请参阅标记资源以便于计费 (p. 387)。	2022 年 7 月 21 日
1.20.15	eks.6	新的平台版本（包括安全修复和增强功能）。	2022 年 5 月 31 日
1.20.15	eks.5	Amazon Security Token Service 端点将从以前的平台版本还原到全局端点。如果您想在为服务账户使用 IAM 角色时使用区域端点，则必须将其启用。有关如何启用区域端点的说明，请参阅 配置服务账户的 Amazon Security Token Service 端点 (p. 405)。	2022 年 4 月 8 日
1.20.11	eks.4	使用 服务账户的 IAM 角色 (p. 395) 时，现在将默认使用 Amazon Security Token Service 区域端点，而不是全局端点。但是此更改将恢复到 eks.5 中的全局端点。 更新后的 Fargate 调度程序在大规模部署期间以更高的速度预置节点。	2022 年 3 月 10 日
1.20.11	eks.3	这是新推出的平台版本，支持在 Kubernetes 控制面板上运行的 VPC 资源控制器上进行 Windows IPv4 地址管理。增加了用于 Fargate Fluent Bit 日志记录的 Kubernetes 筛选指令。	
1.20.7	eks.2	新的平台版本（包括安全修复和增强功能）。	
1.20.4	eks.1	面向 Amazon EKS 的 Kubernetes 版本 1.20 的初始版本。有关更多信息，请参阅Kubernetes 1.20 (p. 63)。	

Kubernetes 版本 1.19

所有 1.19 平台版本都启用了以下准入控制器：DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionWebhook、Na

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.19.16	eks.11	具有改进 etcd 弹性的新平台版本。	2022 年 8 月 15 日
1.19.16	eks.10	新的平台版本（包括安全修复和增强功能）。	2022 年 5 月 31 日
1.19.16	eks.9	Amazon Security Token Service 端点将从以前的平台版本还原到全局端点。如果您想在为服务账	2022 年 4 月 8 日

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
		户使用 IAM 角色时使用区域端点，则必须将其启用。有关如何启用区域端点的说明，请参阅 配置服务账户的 Amazon Security Token Service 端点 (p. 405)。	
1.19.15	eks.8	使用 服务账户的 IAM 角色 (p. 395) 时，现在将默认使用 Amazon Security Token Service 区域端点，而不是全局端点。但是此更改将恢复到 eks.9 中的全局端点。 更新后的 Fargate 调度程序在大规模部署期间以更高的速度预置节点。	2022 年 3 月 10 日
1.19.15	eks.7	这是新推出的平台版本，支持在 Kubernetes 控制面板上运行的 VPC 资源控制器上进行 Windows IPv4 地址管理。增加了用于 Fargate Fluent Bit 日志记录的 Kubernetes 筛选指令。	
1.19.8	eks.6	新的平台版本（包括安全修复和增强功能）。	
1.19.8	eks.5	支持使用 Fargate 的自定义安全组的新平台版本。	
1.19.8	eks.4	新的平台版本（包括安全修复和增强功能）。	
1.19.8	eks.3	新的平台版本（包括安全修复和增强功能）。	
1.19.6	eks.2	新的平台版本（包括安全修复和增强功能）。	
1.19.6	eks.1	面向 Amazon EKS 的 Kubernetes 版本 1.19 的初始版本。有关更多信息，请参阅Kubernetes 1.19 (p. 63)。	

Kubernetes 版本 1.18

所有 1.18 平台版本都启用了以下准入控制器：DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionWebhook、Na

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.18.20	eks.13	具有改进 etcd 弹性的新平台版本。	2022 年 8 月 15 日
1.18.20	eks.12	新的平台版本（包括安全修复和增强功能）。	2022 年 5 月 31 日
1.18.20	eks.11	Amazon Security Token Service 端点将从以前的平台版本还原到全局端点。如果您想在为服务账户使用 IAM 角色时使用区域端点，则必须将其启用。有关如何启用区域端点的说明，请参阅 配置服务账户的 Amazon Security Token Service 端点 (p. 405)。	2022 年 4 月 8 日
1.18.20	eks.10	使用 服务账户的 IAM 角色 (p. 395) 时，现在将默认使用 Amazon Security Token Service 区域端点，	2022 年 3 月 10 日

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
		而不是全局端点。但是此更改将恢复到 `eks.11` 中的全局端点。 更新后的 Fargate 调度程序在大规模部署期间以更高的速度预置节点。	
`1.18.20`	`eks.9`	这是新推出的平台版本，支持在 Kubernetes 控制面板上运行的 VPC 资源控制器上进行 Windows `IPv4` 地址管理。增加了用于 Fargate Fluent Bit 日志记录的 Kubernetes 筛选指令。	
`1.18.20`	`eks.8`	新的平台版本（包括安全修复和增强功能）。	
`1.18.16`	`eks.7`	支持使用 Fargate 的自定义安全组的新平台版本。	
`1.18.16`	`eks.6`	新的平台版本（包括安全修复和增强功能）。	
`1.18.16`	`eks.5`	新的平台版本（包括安全修复和增强功能）。	
`1.18.9`	`eks.4`	新的平台版本（包括安全修复和增强功能）。	
`1.18.9`	`eks.3`	包括对 Amazon EKS 附加组件 (p. 341)和 Fargate 日志记录 (p. 138) 的支持。	
`1.18.9`	`eks.2`	新的平台版本（包括安全修复和增强功能）。	
`1.18.8`	`eks.1`	面向 Amazon EKS 的 Kubernetes 版本 `1.18` 的初始版本。有关更多信息，请参阅Kubernetes 1.18 (p. 65)。	

Autoscaling

弹性伸缩是一项功能，可以自动上下伸缩您的资源以满足不断变化的需求。若没有此项重要的 Kubernetes 功能，则需要耗费大量的人力资源来手动执行这些工作。

Amazon EKS 支持两款弹性缩放产品：Kubernetes Cluster Autoscaler 和 Karpenter 开源自动缩放项目。Cluster Autoscaler 使用 Amazon 缩放组，而 Karpenter 则直接使用 Amazon EC2 Fleet。

Cluster Autoscaler

当 pods 失败或被重新安排到其他节点时，Kubernetes Cluster Autoscaler 会自动调整集群中的节点数。Cluster Autoscaler 通常作为部署安装在集群中。它使用领导选举来确保高可用性，但一次只能由一个副本完成扩展。

在部署集群 Cluster Autoscaler 之前，请确保您熟悉 Kubernetes 概念与 Amazon 功能相互联系的方式。本主题中使用了以下术语：

- Kubernetes Cluster Autoscaler – Kubernetes 控制面板的核心组件，用于制定调度和缩放决策。有关更多信息，请参阅 GitHub 上的 Kubernetes 控制面板常见问题。
- Amazon 云提供商实施 – Kubernetes Cluster Autoscaler 的扩展，通过与 Amazon 产品和服务（例如 Amazon EC2）通信实施 Kubernetes Cluster Autoscaler 的决策。有关更多信息，请在 GitHub 上参阅 Amazon 上的 Cluster Autoscaler。
- 节点组 – 集群中一组节点的 Kubernetes 抽象。节点组不是真正的 Kubernetes 资源，但它们在 Cluster Autoscaler、集群 API 和其他组件中作为抽象被找到。在单个节点组中找到的节点可能共享多个常见属性，如标签和污点。但是，它们仍然可以由多个可用区或实例类型组成。

- Amazon EC2 Auto Scaling 组 – 为 Cluster Autoscaler 所用的一个Amazon功能。Auto Scaling 组适用于大量使用案例。Amazon EC2 Auto Scaling 组配置为启动自动加入其 Kubernetes 集群的实例。它们还将标签和污点应用到 Kubernetes API 中的相应节点资源。

作为参考，托管节点组 (p. 87) 使用 Amazon EC2 Auto Scaling 组进行管理，并且与 Cluster Autoscaler 兼容。

本主题说明如何将 Cluster Autoscaler 部署到 Amazon EKS 集群以及如何配置它来修改 Amazon EC2 Auto Scaling 组。

先决条件

在部署集群 Cluster Autoscaler之前，您必须满足以下先决条件：

- 现有 Amazon EKS 集群：如果您没有集群，请参阅 创建 Amazon EKS 集群 (p. 21)。
- 集群的现有 IAM OIDC 提供商。要确定是否具有 IAM OIDC 提供商，还是需要创建一个，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。
- 带有 Auto Scaling 组标签的节点组。Cluster Autoscaler 要求您 Auto Scaling 组上具有以下标签，以便能够自动发现它们。
  - 如果您使用 `eksctl` 创建节点组，则节点组会自动应用这些标签。
  - 如果未使用 `eksctl`，则必须使用以下标签手动标记您的 Auto Scaling 组。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南 中的标记 Amazon EC2 资源。

密钥	值
`k8s.io/cluster-autoscaler/`*`my-cluster`*	`owned`
`k8s.io/cluster-autoscaler/enabled`	`true`

创建 IAM policy 和角色

创建授予 Cluster Autoscaler 使用 IAM 角色所需权限的 IAM policy。将整个过程中的所有 *`example values`* 替换为您自己的值。

1. 创建一个 IAM policy。

   a. 将以下内容保存到名为 `cluster-autoscaler-policy.json` 的文件中。如果现有节点组是使用 `eksctl` 创建的并且您使用了 `--asg-access` 选项，则此策略已存在，您可以跳至第 2 步。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "autoscaling:SetDesiredCapacity",
                "autoscaling:TerminateInstanceInAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/k8s.io/cluster-autoscaler/my-cluster": "owned"
                }
            }
```

```
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeAutoScalingGroups",
                "ec2:DescribeLaunchTemplateVersions",
                "autoscaling:DescribeTags",
                "autoscaling:DescribeLaunchConfigurations"
            ],
            "Resource": "*"
        }
    ]
}
```

b. 使用以下命令创建策略。您可以更改 `policy-name` 的值。

```
aws iam create-policy \
    --policy-name AmazonEKSClusterAutoscalerPolicy \
    --policy-document file://cluster-autoscaler-policy.json
```

记下输出中返回的 Amazon Resource Name (ARN)。您需要在后面的步骤中用到它。

2. 您可以创建一个 IAM 角色并使用 `eksctl` 或 Amazon Web Services Management Console 向其附加 IAM policy。为以下说明选择所需的选项卡。

eksctl

1. 如果您使用 `eksctl` 创建 Amazon EKS 集群，请运行以下命令。如果您使用 `--asg-access` 选项创建集群，请将 *AmazonEKSClusterAutoscalerPolicy* 替换为 `eksctl` 为您创建的 IAM policy 的名称。策略名称类似于 `eksctl-`*my-cluster*`-nodegroup-ng-`*xxxxxxxx*`-PolicyAutoScaling`。

```
eksctl create iamserviceaccount \
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=cluster-autoscaler \
  --attach-policy-
arn=arn:aws:iam::111122223333:policy/AmazonEKSClusterAutoscalerPolicy \
  --override-existing-serviceaccounts \
  --approve
```

2. 如果您使用 `--asg-access` 选项创建集群，我们建议您分离 `eksctl` 创建的 IAM policy，然后将其附加到 `eksctl` 为节点组创建的 Amazon EKS 节点 IAM 角色 (p. 427)。将策略与节点 IAM 角色分离，以便 Cluster Autoscaler 正常运行。分离策略不会为节点上的其他 pods 提供策略中的权限。有关更多信息，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的删除 IAM 身份权限。

Amazon Web Services Management Console

a. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
b. 在左侧导航窗格中，选择 Roles（角色）。然后选择 Create role (创建角色)。
c. 在 Trusted entity type（受信任的实体类型）部分中，选择 Web identity（Web 身份）。
d. 在 Web identity（Web 身份）部分：
   i. 对于 Identity provider（身份提供商），为您的集群选择 OpenID Connect provider URL（提供商 URL）（如 Amazon EKS 中的集群 Overview（概述）选项卡中所示）。
   ii. 对于 Audience (受众)，请选择 `sts.amazonaws.com`。
e. 选择 Next (下一步)。

f. 在 Filter policies (筛选器策略) 框中，输入 **AmazonEKSClusterAutoscalerPolicy**。然后，选中搜索返回的策略名称左侧的复选框。

g. 选择 Next (下一步)。

h. 对于 Role name（角色名称），请为角色输入唯一名称，例如 **AmazonEKSClusterAutoscalerRole**。

i. 对于 Description（说明），请输入描述性文本，例如 **Amazon EKS - Cluster autoscaler role**。

j. 选择 Create role（创建角色）。

k. 创建角色后，在控制台中选择角色以将其打开进行编辑。

l. 选择 Trust relationships（信任关系）选项卡，然后选择 Edit trust policy（编辑信任策略）。

m. 该行看起来类似于以下行：

```
"oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

将该行更改为类似于以下行。将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 替换为集群的 OIDC 提供商 ID。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

```
"oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:cluster-autoscaler"
```

n. 选择 Update policy（更新策略）以完成操作。

部署 Cluster Autoscaler

要部署 Cluster Autoscaler，请完成以下步骤。建议您查看 部署注意事项 (p. 77) 并优化 Cluster Autoscaler 部署，然后再将其部署到生产集群。

部署 Cluster Autoscaler

1.
```
curl -o cluster-autoscaler-autodiscover.yaml https://raw.githubusercontent.com/
kubernetes/autoscaler/master/cluster-autoscaler/cloudprovider/aws/examples/cluster-
autoscaler-autodiscover.yaml
```

2. 修改 YAML 文件并将 *<YOUR CLUSTER NAME>* 替换为您的集群名称。另请考虑替换您的环境决定的 `cpu` 和 `memory` 值。

3. 将 YAML 文件应用于集群。

   中国（宁夏）或中国（北京）

   a. 使用下面的命令下载清单。

   ```
   curl -o cluster-autoscaler-autodiscover.yaml https://raw.githubusercontent.com/
   kubernetes/autoscaler/master/cluster-autoscaler/cloudprovider/aws/examples/cluster-
   autoscaler-autodiscover.yaml
   ```

   b. 修改清单。

      i. 查看您下载的一个或多个清单文件，并记下镜像的名称。使用下面的命令将镜像下载到本地。

      ```
      docker pull image:tag
      ```

      ii. 使用以下命令标记要推送到中国 Amazon Elastic Container Registry 存储库的镜像。

```
docker tag image:tag 111122223333.dkr.ecr.cn-north-1.amazonaws.com.cn/image:tag
```

iii. 通过以下命令将镜像推送到中国 Amazon ECR 存储库。

```
docker push image:tag 111122223333.dkr.ecr.cn-north-1.amazonaws.com.cn/image:tag
```

iv. 更新 Kubernetes 的一个或多个清单文件，以引用您所在区域的 Amazon ECR 镜像 URL。

c. 运用该清单。

```
kubectl apply -f cluster-autoscaler-autodiscover.yaml
```

4. 使用您以前创建的 IAM 角色的 ARN 对 `cluster-autoscaler` 服务账户添加注释。将 *example values* 替换为您自己的值。

```
kubectl annotate serviceaccount cluster-autoscaler \
  -n kube-system \
  eks.amazonaws.com/role-
arn=arn:aws:iam::ACCOUNT_ID:role/AmazonEKSClusterAutoscalerRole
```

5. 使用以下命令修补部署以向 Cluster Autoscaler pods 添加 `cluster-autoscaler.kubernetes.io/safe-to-evict` 注释。

```
kubectl patch deployment cluster-autoscaler \
  -n kube-system \
  -p '{"spec":{"template":{"metadata":{"annotations":{"cluster-
autoscaler.kubernetes.io/safe-to-evict": "false"}}}}}'
```

6. 使用以下命令编辑 Cluster Autoscaler 部署。

```
kubectl -n kube-system edit deployment.apps/cluster-autoscaler
```

编辑用于添加以下选项的 `cluster-autoscaler` 容器命令。`--balance-similar-node-groups` 确保所有可用区都有足够的可用计算。`--skip-nodes-with-system-pods=false` 确保缩减到零时不会出现问题。

- `--balance-similar-node-groups`
- `--skip-nodes-with-system-pods=false`

```
    spec:
      containers:
      - command
        - ./cluster-autoscaler
        - --v=4
        - --stderrthreshold=info
        - --cloud-provider=aws
        - --skip-nodes-with-local-storage=false
        - --expander=least-waste
        - --node-group-auto-discovery=asg:tag=k8s.io/cluster-autoscaler/enabled,k8s.io/
cluster-autoscaler/my-cluster
        - --balance-similar-node-groups
        - --skip-nodes-with-system-pods=false
```

保存并关闭该文件以应用更改。

7. 在 Web 浏览器中从 GitHub 打开 Cluster Autoscaler 版本页面，找到与您集群的 Kubernetes 主版本和次要版本相匹配的 Cluster Autoscaler 最新版本。例如，如果您集群的 Kubernetes 版本是 `1.23`，则查

找以 1.23 开头的最新 Cluster Autoscaler 版本。记录该版本的语义版本号 (1.23.*n*) 以在下一步中使用。

8. 使用以下命令，将 Cluster Autoscaler 映像标签设置为您在上一步中记录的版本。将 *1.23.n* 替换为您自己的值。

```
kubectl set image deployment cluster-autoscaler \
  -n kube-system \
  cluster-autoscaler=k8s.gcr.io/autoscaling/cluster-autoscaler:v1.23.n
```

查看 Cluster Autoscaler 日志

部署 Cluster Autoscaler 之后，您可以查看日志并验证它在监控您的集群负载。

使用以下命令查看您的 Cluster Autoscaler 日志。

```
kubectl -n kube-system logs -f deployment.apps/cluster-autoscaler
```

输出示例如下。

```
I0926 23:15:55.165842       1 static_autoscaler.go:138] Starting main loop
I0926 23:15:55.166279       1 utils.go:595] No pod using affinity / antiaffinity found in
 cluster, disabling affinity predicate for this loop
I0926 23:15:55.166293       1 static_autoscaler.go:294] Filtering out schedulables
I0926 23:15:55.166330       1 static_autoscaler.go:311] No schedulable pods
I0926 23:15:55.166338       1 static_autoscaler.go:319] No unschedulable pods
I0926 23:15:55.166345       1 static_autoscaler.go:366] Calculating unneeded nodes
I0926 23:15:55.166357       1 utils.go:552] Skipping ip-192-168-3-111.region-
code.compute.internal - node group min size reached
I0926 23:15:55.166365       1 utils.go:552] Skipping ip-192-168-71-83.region-
code.compute.internal - node group min size reached
I0926 23:15:55.166373       1 utils.go:552] Skipping ip-192-168-60-191.region-
code.compute.internal - node group min size reached
I0926 23:15:55.166435       1 static_autoscaler.go:393] Scale down status:
 unneededOnly=false lastScaleUpTime=2019-09-26 21:42:40.908059094 ...
I0926 23:15:55.166458       1 static_autoscaler.go:403] Starting scale down
I0926 23:15:55.166488       1 scale_down.go:706] No candidates for scale down
```

部署注意事项

请查看以下注意事项，以优化 Cluster Autoscaler 部署。

扩展注意事项

Cluster Autoscaler 可以配置为包含节点的任何附加功能。这些功能可以包括附加到节点的 Amazon EBS 卷、节点的 Amazon EC2 实例类型或 GPU 加速器。

跨多个可用区设定节点组的范围

我们建议您配置多个节点组，将每个组的范围设定为单个可用区，并启用 `--balance-similar-node-groups` 功能。如果您只创建一个节点组，则将该节点组的范围设置为跨越多个可用区。

如果将 `--balance-similar-node-groups` 设为 true，请确保希望 Cluster Autoscaler 平衡的节点组具有匹配的标签（自动添加的区域标签除外）。可以向有着不同标签的各个节点传递一个 `--balancing-ignore-label` 标志，这样就会直接对它们进行平衡，而不考虑它们的标签；但只能在需要时这样做。

优化节点组

Cluster Autoscaler 会假定您使用节点组的方式。包括您在组中使用的实例类型。要与这些假设保持一致，请根据以下注意事项和建议配置节点组：

- 节点组中的每个节点必须具有相同的调度属性，包括标签、污点和资源。
  - 对于 `MixedInstancePolicies`，实例类型必须具有兼容的 CPU、内存和 GPU 规范。
  - 策略中指定的第一个实例类型模拟调度。
  - 如果您的策略具有拥有更多资源的其他实例类型，则在横向扩展后可能会浪费资源。
  - 如果您的策略具有其他实例类型，其资源比原始实例类型少，则 pods 在实例上调度可能失败。
- 请使用较多节点配置较少数量的节点组，因为相反的配置可能会对可扩展性产生不利影响。
- 只要两个系统都为 Amazon EC2 功能提供支持，就可以使用 Amazon EC2 功能（例如，使用区域和 `MixedInstancePolicy`。）

如果可能，建议您使用 托管节点组 (p. 87)。托管节点组附带强大的管理功能。这包括适用于 Cluster Autoscaler 的功能，例如 Amazon EC2 Auto Scaling 自动组发现功能和正常节点终止功能。

将 EBS 卷用作持久性存储

持久性存储对于构建有状态应用程序（如数据库和分布式缓存）至关重要。借助 Amazon EBS 卷，您可以在 Kubernetes 上构建有状态的应用程序。但是，您只能在单个可用区中构建。有关更多信息，请参阅如何使用 Amazon EKS 中的持久性存储？。要获得更好的解决方案，请考虑为每个可用区使用单独的 Amazon EBS 卷构建跨多个可用区的有状态应用程序。这样做意味着您的应用程序可以实现高度可用性。此外，Cluster Autoscaler 还可以平衡 Amazon EC2 Auto Scaling 组的扩展。为此，请确保满足以下条件：

- 通过设置 `balance-similar-node-groups=true` 启用节点组平衡。
- 将节点组配置为具有相同的设置（除了位于多个可用区内并使用不同的 Amazon EBS 卷之外）。

协同调度

Machine Learning 分布式培训作业可从同区节点配置的最小化延迟中获益匪浅。这些工作负载将多个 pods 部署到特定区域。您可以通过使用 `topologyKey: topology.kubernetes.io/zone` 为所有协同调度 pods 或节点亲和性设置 pod 亲和性来实现这一点。使用此配置，Cluster Autoscaler 可横向扩展特定区域以满足需求。分配多个 Amazon EC2 Auto Scaling 组，每个可用区分配一个组，以便为整个协同调度工作负载启用故障转移。确保满足以下条件：

- 通过设置 `balance-similar-node-groups=true` 启用节点组平衡。
- 当集群同时包含区域节点组和可用区节点组时，使用节点亲和性或 pod 抢占，或同时使用这二者。
  - 使用节点亲和性强制或鼓励使用区域 pods 并避免可用区节点组。
  - 不要将可用区 pods 调度到区域节点组上。这样做可能会导致用于区域 pods 的容量不均衡。
  - 如果您的可用区工作负载能够容忍中断和重新定位，则可以配置 pod 抢占。这样做会强制抢占和重新调度区域扩展 pods 的争夺较少的区。

加速器和 GPU

某些集群使用专用硬件加速器，例如专用 GPU。横向扩展时，加速器可能需要几分钟才能将资源通告到集群。在此期间，Cluster Autoscaler 会模拟此节点具有加速器的情况。但是，在加速器准备就绪并更新节点的可用资源之前，无法在节点上调度待处理的 pods。这可能会导致重复、不必要的横向扩展。

即使加速器未使用，也不会考虑缩减具有加速器和高 CPU 或内存利用率的节点。但这可能会导致不必要的成本。要避免这些成本，在节点具有未占用的加速器的情况下，Cluster Autoscaler 可以应用特殊规则来考虑节点进行缩减。

要确保在这些情况下执行正确行为，请在加速器节点上配置 `kubelet`，以便在节点加入集群之前对其进行标注。Cluster Autoscaler 会使用此标注选择器调用经加速器优化的行为。确保满足以下条件：

- 用于 GPU 节点的 `kubelet` 配置有 `--node-labels k8s.amazonaws.com/accelerator=$ACCELERATOR_TYPE`。
- 带有加速器的节点遵守相同的调度属性规则。

从零扩展

Cluster Autoscaler 可以将节点组缩减到零和从零扩展节点组。这可以节省大量成本。Cluster Autoscaler 会通过检查其 `LaunchConfiguration` 或者 `LaunchTemplate` 中指定的 `InstanceType` 来检测 Auto Scaling 组的 CPU、内存和 GPU 资源。某些 pods 需要额外的资源，例如 `WindowsENI` 或者 `PrivateIPv4Address`。或者它们可能需要特定的 `NodeSelectors` 或 `Taints`。后两者不能从 `LaunchConfiguration` 发现。但是 Cluster Autoscaler 可以通过从 Auto Scaling 组上的以下标签中发现它们来解释这些因素。

```
Key: k8s.io/cluster-autoscaler/node-template/resources/$RESOURCE_NAME
Value: 5
Key: k8s.io/cluster-autoscaler/node-template/label/$LABEL_KEY
Value: $LABEL_VALUE
Key: k8s.io/cluster-autoscaler/node-template/taint/$TAINT_KEY
Value: NoSchedule
```

Note

- 当缩减到零时，您的容量将返回到 Amazon EC2，并可能会在将来变得不可用。
- 您可以使用 describeNodegroup 诊断缩减到零和从零扩展时托管节点组存在的问题。

其他配置参数

有许多配置选项可用于调整 Cluster Autoscaler 的行为和性能。有关参数的完整列表，请参阅 GitHub 上的面向 CA 的参数是什么？。

性能注意事项

您可以更改一些关键项目来优化 Cluster Autoscaler 的性能和可扩展性。主要项目包括提供给进程的任何资源、算法的扫描间隔以及集群中的节点组数。但是，此算法的真正运行时间复杂性也涉及其他几个因素，其中包括调度插件的复杂性和 pods 的数量。这些因素被视为不可配置的参数，因为它们是集群工作负载的组成部分，而且不容易进行优化。

可扩展性是指随着 Kubernetes 集群中的 pods 和节点数量的增加，Cluster Autoscaler 的性能如何。如果达到其可扩展性配额，则 Cluster Autoscaler 的性能和功能会降低。此外，当超出其可扩展性配额时，Cluster Autoscaler 就无法再添加或删除集群中的节点。

性能是指 Cluster Autoscaler 可以做出和实施扩展决策的速度。性能完美的 Cluster Autoscaler 可以立即做出决策并调用扩展操作，以对特定状况做出响应，例如 pod 变得不可调度。

熟悉弹性伸缩算法的运行时间复杂性。这样可以更轻松地优化 Cluster Autoscaler，以使其在大型集群（具有超过 1000 个节点）中运行良好。

Cluster Autoscaler 会将整个集群的状态加载到内存中，包括 pods、节点和节点组。在每个扫描间隔时间期间，算法会识别不可调度的 pods 并模拟每个节点组的调度。请注意，以不同的方式优化这些因素都会需要做出不同的折衷。

垂直弹性伸缩

您可以通过增加其部署的资源请求，将 Cluster Autoscaler 扩展到较大的集群。这是实现这一点的更简单的方法之一。增加大型集群的内存和 CPU。请注意，内存和 CPU 应该增加的量在很大程度上取决于特定的集群大小。自动缩放算法将所有 pods 和节点存储在内存中。在某些情况下，这可能会导致内存占用大于 1

GB。您通常需要手动增加资源。如果您发现经常需要手动增加资源，请考虑使用 Addon Resizer 或 Vertical Pod Autoscaler 来自动执行该过程。

减少节点组数量

您可以减少节点组的数量来提高 Cluster Autoscaler 在大型集群中的性能。如果您以单个团队或应用程序为基础构建节点组，此做法可能具有挑战性。虽然 Kubernetes API 完全支持此做法，但它被认为是 Cluster Autoscaler 反面模式，会影响可扩展性。使用多个节点组有许多好处，例如同时使用 Spot 或 GPU 实例。在许多情况下有替代设计，可在使用少量组的同时实现同样效果。确保满足以下条件：

- 使用命名空间而非节点组来隔离 pods。
  - 您可能无法在低信任度、多租户集群中采用此做法。
  - 恰当设置 pod `ResourceRequests` 和 `ResourceLimits` 来避免资源争用。
  - 使用较大的实例类型实现更优化的装箱并减少系统 pod 开销。
- 避免使用 `NodeTaints` 或 `NodeSelectors` 来调度 pods。仅在少数情况下使用它们。
- 将区域资源定义为具有多个可用区的单个 Amazon EC2 Auto Scaling 组。

缩短扫描间隔时间

缩短扫描间隔时间（如原定设置为 10 秒）可确保 Cluster Autoscaler 在 pods 变为不可调度时尽快做出响应。但是，每次扫描都会导致对 Kubernetes API 和 Amazon EC2 Auto Scaling 组或 Amazon EKS 托管节点组 API 进行多次 API 调用。这些 API 调用可能会导致您的 Kubernetes 控制面板的速率受限甚至服务不可用。

原定设置的扫描间隔时间为十秒，但在Amazon上，启动节点需要更长的时间来启动新的实例。这意味着，可以在不显著增加整体纵向扩展时间的情况下提高扫描间隔时间。例如，如果启动节点需要两分钟时间，请不要将扫描间隔更改为一分钟，因为这可能会出现虽然 API 调用减少 6 倍但纵向扩展速度减慢 38%。

跨节点组共享

您可以将 Cluster Autoscaler 配置为对一组特定的节点组进行操作。通过使用此功能，您可以部署 Cluster Autoscaler 的多个实例。将每个实例配置为在一组不同的节点组上运行。这样，您就可以使用任意大量的节点组，从而换取可扩展性的成本。但是，我们只建议您在万不得已时才采用此做法来提高 Cluster Autoscaler 的性能。

这种配置有其缺点：可能会导致多个节点组不必要的横向扩展。在 `scale-down-delay` 后，额外节点会相应缩减。

```
metadata:
  name: cluster-autoscaler
  namespace: cluster-autoscaler-1

...

--nodes=1:10:k8s-worker-asg-1
--nodes=1:10:k8s-worker-asg-2

---

metadata:
  name: cluster-autoscaler
  namespace: cluster-autoscaler-2

...

--nodes=1:10:k8s-worker-asg-3
--nodes=1:10:k8s-worker-asg-4
```

确保满足以下条件。

- 每个分区都配置为指向一组唯一的 Amazon EC2 Auto Scaling 组。
- 每个分区都部署到一个单独的命名空间，以避免领导选举冲突。

成本效益和可用性

优化 Cluster Autoscaler 成本效益的主要选择与预置 Amazon EC2 实例有关。此外，成本效益必须与可用性保持平衡。本节介绍使用 Spot 实例降低成本和超额配置以减少创建新节点时的延迟等策略。

- 可用性 – 能够快速、不中断地调度 Pod。即使是在需要调度新创建的 pods 以及缩减的节点终止向其调度的任何剩余 pods 时，也能做到这一点。
- 成本 – 由横向扩展和横向缩减事件背后的决策确定。如果现有节点未充分利用，或添加的新节点对于传入的 pods 而言太大，则资源将会被浪费。根据具体的使用案例，可能会由于积极的缩减决策导致产生与过早终止 pods 相关的成本。

Spot 实例

您可以使用节点组中的 Spot 实例，按需价格最高可节省 90%。这可能会在 Amazon EC2 需要收回容量时随时中断 Spot 实例。每当您的 Amazon EC2 Auto Scaling 组由于缺乏可用容量而无法纵向扩展时，就会发生 `Insufficient Capacity Errors`。选择许多不同的实例系列有两个主要好处。首先，它可以通过利用许多 Spot 容量池来提高您实现所需扩展的机会。其次，它还可以减少 Spot 实例中断对集群可用性的影响。包含 Spot 实例的混合实例策略是在不增加节点组数量的情况下提高多样性的好方法。但是，请注意，如果您需要有保障的资源，请使用按需实例，而不要使用 Spot 实例。

当对实例的需求增加时，可能会终止 Spot 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 Spot 实例中断部分。Amazon Node Termination Handler 项目会在节点出现故障时自动提醒 Kubernetes 控制面板。该项目使用 Kubernetes API 封锁节点，确保节点未安排有任何新工作，然后耗尽节点并删除任何现有工作。

在配置混合实例策略时，所有实例类型具有相似的资源容量至关重要。Autoscaler 的调度模拟器使用混合实例策略中的第一个实例类型。如果后续实例类型较大，在纵向扩展后可能会造成资源浪费。如果实例较小，由于容量不足，pods 可能无法在新实例上实现调度。例如，`M4`、`M5`、`M5a`，和 `M5n` 实例都具有相似数量的 CPU 和内存，是混合实例策略的绝佳候选项。Amazon EC2 实例选择器工具可以帮助您识别相似的实例类型或其他重要条件，例如大小。有关更多信息，请参阅 GitHub 上的 Amazon EC2 实例选择器。

我们建议您将按需实例和 Spot 实例容量分离到不同的 Amazon EC2 Auto Scaling 组中。由于按需实例和 Spot 实例的调度属性不同，我们建议使用基本容量策略。Spot 实例可以随时中断。当 Amazon EC2 需要收回容量时，抢占节点通常会受到影响，因此需要对抢占行为具有明确的 pod 容忍度。这就导致了不同的节点调度属性，因此应将它们分到多个 Amazon EC2 Auto Scaling 组中。

Cluster Autoscaler 涉及 Expander 概念。它们共同为选择要扩展的节点组提供了不同的策略。`--expander=least-waste` 策略是一种不错的通用型原定设置，如果您打算使用多个节点组实现 Spot 实例多样化（如前所述），那么它可以通过扩缩在扩缩活动后利用率最佳的组来进一步优化节点组的成本。

确定节点组或 Auto Scaling 组的优先级

您还可以通过使用 `Priority` 扩展器来配置基于优先级的弹性伸缩。`--expander=priority` 使您的集群能够确定节点组或 Auto Scaling 组的优先级，如果节点组由于任何原因无法扩展，则其会选择优先级列表中下一个节点组。这在某些情况下非常有用，例如您想要使用 `P3` 实例类型，因为它们的 GPU 为您的工作负载提供了最佳性能，但您也可以将 `P2` 实例类型作为第二个选择。例如：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-autoscaler-priority-expander
  namespace: kube-system
data:
  priorities: |-
```

```
    10:
      - .*p2-node-group.*
    50:
      - .*p3-node-group.*
```

Cluster Autoscaler 尝试纵向扩展与名称 `p3-node-group` 相匹配的 Amazon EC2 Auto Scaling 组。如果此操作在 `--max-node-provision-time` 内未成功，则其会尝试扩展与名称 `p2-node-group` 相匹配的 Amazon EC2 Auto Scaling 组。此值原定设置为 15 分钟，可以降低此值以选择响应更快的节点组。但是，如果此值太低，则可能会发生不必要的横向扩展。

超额配置

Cluster Autoscaler 可以确保仅在需要节点时将节点添加到集群，并在节点未使用时删除节点，以此来帮助最大限度地降低成本。这会极大地影响部署延迟，因为许多 pods 必须等待节点纵向扩展后才能进行调度。节点可能需要几分钟才能使用，这可能会将 pod 调度延迟增加一个数量级。

此问题可以通过超额配置来缓解，超额配置是以成本换取调度延迟。使用具有负优先级的临时 pods 来实施超额配置。这些 pods 占用集群中的空间。当新创建的 pods 不可调度且具有较高的优先级时，会抢占临时 pods 以腾出空间。然后，临时 pods 变得不可调度，从而导致 Cluster Autoscaler 横向扩展新的超额配置节点。

超额配置还有其他好处。在没有超额配置的情况下，高利用率集群中的 pods 可以使用 `preferredDuringSchedulingIgnoredDuringExecution` 规则做出次优调度决定。这种情况的一个常见用例是使用 `AntiAffinity` 跨可用区为高可用性应用程序分离 pods。超额配置可以显著提高所需区域的节点的可用性几率。

选择适当数量的超额配置容量非常重要。确保选择适当数量的一种方法是，用平均纵向扩展频率除以纵向扩展新节点所需的时间。例如，如果您平均每 30 秒需要一个新节点，而 Amazon EC2 预置新节点需要 30 秒，则超额配置单个节点可确保始终有一个额外的节点可用。这样就可以以单个额外 Amazon EC2 实例为成本将调度延迟减少 30 秒。为了做出更好的分区调度决策，您还可以超额配置节点数，使其与 Amazon EC2 Auto Scaling 组中的可用区数量相同。这样可确保调度器可以为传入 pods 选择最佳区域。

防止缩减移出

移出某些工作负载成本非常高昂。大数据分析、机器学习任务和测试运行器可能需要很长时间才能完成，而且如果这些过程被中断，必须要重新启动。Cluster Autoscaler 可以帮助缩减 `scale-down-utilization-threshold` 下的任意节点。此操作会中断节点上的所有剩余 pods。但是，您可以通过确保使用经 Cluster Autoscaler 认可的标注来保护移出费用较为昂贵的 pods，来防止发生这种情况。为此，请确保移出费用较为昂贵的 pods 具有标注 `cluster-autoscaler.kubernetes.io/safe-to-evict=false`。

Karpenter

Amazon EKS 支持 Karpenter 开源弹性缩放项目。要部署该项目，请参阅 Karpenter 文档。

关于 Karpenter

Karpenter 是一款灵活的高性能 Kubernetes 集群自动缩放器，可帮助提高应用程序可用性和集群效率。Karpenter 只需不到一分钟时间，即可启动适当规模的计算资源（例如 Amazon EC2 实例）来响应不断变化的应用程序负载。通过将 Kubernetes 与 Amazon 相集成，Karpenter 可以即时调配精准满足工作负载需求的计算资源。Karpenter 会根据集群工作负载的具体需求来自动调配新的计算资源。这包括计算、存储、加速和调度需求。Amazon EKS 支持使用 Karpenter 的集群，但 Karpenter 可以与任何合规的 Kubernetes 集群配合使用。

Karpenter 的工作原理

Karpenter 通过在集群的整个生命周期内观察传入的 pods，与 Kubernetes 调度器协同工作。它通过启动或终止节点来最大限度提高应用程序可用性和集群利用率。当集群内有足够的容量时，Kubernetes 调度器会

像往常一样部署传入的 pods。当启用无法使用集群现有容量调度的 pods 时，Karpenter 会绕过 Kubernetes 调度器，直接使用提供商的计算服务（例如 Amazon EC2）来启动满足这些 pods 所需的最少计算资源，并将 pods 绑定到所调配的节点。当 pods 被删除或重新调度到其他节点后，Karpenter 会寻找机会终止未充分利用的节点。在集群中运行数量更少、规模更大的节点可以减少 daemonsets 和 Kubernetes 系统组件的开销，并提供更多实现高效装填的机会。

先决条件

在部署 Karpenter 之前，您必须满足以下先决条件：

- 现有 Amazon EKS 集群：如果您没有集群，请参阅 创建 Amazon EKS 集群 (p. 21)。
- 集群的现有 IAM OIDC 提供商。要确定是否具有 IAM OIDC 提供商，还是需要创建一个，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。
- 一个有权创建集群的用户或角色。
- Amazon CLI
- 安装或更新 `kubectl` (p. 4)
- 将 Helm 与 Amazon EKS 结合使用 (p. 384)

如果愿意，您可以使用 `eksctl` 部署 Karpenter。请参阅安装或更新 `eksctl` (p. 10)。

Amazon EKS 节点

您的 Amazon EKS 集群可以在 自行管理的节点 (p. 106)、Amazon EKS 托管节点组 (p. 87) 和 Amazon Fargate (p. 127) 的任意组合中调度 pods。要了解集群中部署的节点的更多信息，请参阅 查看 Kubernetes 资源 (p. 458)。

> **Note**
>
> 节点必须位于与您创建集群时选择的子网相同 VPC 中。不过，节点不需要在相同子网中。

下表提供了在决定哪些选项最符合您的要求时需要评估的几个标准。此表不包括在 Amazon EKS 之外创建的已连接节点 (p. 495)，这些节点只能够查看。

> **Note**
>
> Bottlerocket 与此表中的一般信息有一些具体的区别。有关更多信息，请参阅 GitHub 上的 Bottlerocket 文档。

标准	EKS 托管节点组	自行管理的节点	Amazon Fargate
可以部署到 Amazon Outposts	否	是 – 有关更多信息，请参阅 Amazon Outposts 上的 Amazon EKS 节点 (p. 189)。	否
可以部署到 Amazon 本地区域	否	是 – 有关更多信息，请参阅 Amazon EKS 和 Amazon 本地区域 (p. 476)。	否
可以运行需要 Windows 的容器	否	是 (p. 48) – 尽管如此，您的集群仍然需要至少一个 Linux 节点（建议两个，以确保可用）。	否
可以运行需要 Linux 的容器	是	是	是
可以运行需要 Inferentia 芯片的工作负载	是 (p. 350) – 仅限 Amazon Linux 节点	是 (p. 350) – 仅限 Amazon Linux	否
可以运行需要 GPU 的工作负载	是 (p. 158) – 仅限 Amazon Linux 节点	是 (p. 158) – 仅限 Amazon Linux	否
可以运行需要 ARM 处理器的工作负载	是 (p. 159)	是 (p. 159)	否
可以运行 Amazon Bottlerocket	是	是 (p. 112)	否
pods 之间共享内核运行时间环境	是 – 每个节点上的所有 pods	是 – 每个节点上的所有 pods	否 – 每个 pod 拥有专用的内核

标准	EKS 托管节点组	自行管理的节点	Amazon Fargate
pods 之间共享 CPU、内存、存储和网络资源。	是 – 可能导致每个节点上存在未使用的资源	是 – 可能导致每个节点上存在未使用的资源	否 – 每个 pod 都有专用资源，并且可以独立调整大小以最大限度地提高资源利用率。
容器组（pod）可以使用多于 pod 规范中所请求的硬件和内存	是 – 如果 pod 需要的资源多于请求的资源，并且节点上有可用的资源，那么该 pod 可以使用额外的资源。	是 – 如果 pod 需要的资源多于请求的资源，并且节点上有可用的资源，那么该 pod 可以使用额外的资源。	否 – 但是，可以使用更大的 vCPU 和内存配置重新部署 pod。
必须部署和管理 Amazon EC2 实例	是 (p. 90) – 如果您部署了 Amazon EKS 优化版 AMI，则可通过 Amazon EKS 自动执行。如果您部署了自定义 AMI，则必须手动更新实例。	是 – 手动配置或使用 Amazon EKS 提供的 Amazon CloudFormation 模板来部署 Linux (x86) (p. 107)、Linux (Arm) (p. 159)或 Windows (p. 48) 节点。	否
必须保护、维护和修补 Amazon EC2 实例的操作系统	是	是	否
可以在部署节点时提供引导参数，例如额外的 kubelet 实际参数。	是 – 结合使用启动模板 (p. 100)和自定义 AMI	是 – 有关更多信息，请查看 GitHub 上的引导脚本使用信息。	否
可以从与分配给节点的 IP 地址不同的 CIDR 块为 pods 分配 IP 地址。	是 – 结合使用启动模板 (p. 100)和自定义 AMI	是的，使用 教程：自定义联网 (p. 258)。	否
可以通过 SSH 访问节点	是	是	否 – 没有主机操作系统要通过 SSH 访问的节点。
可以将您自己的自定义 AMI 部署到节点	是 – 使用启动模板 (p. 100)	是	否
可以将自己的自定义 CNI 部署到节点	是 – 结合使用启动模板 (p. 100)和自定义 AMI	是	否

标准	EKS 托管节点组	自行管理的节点	Amazon Fargate
必须自己更新节点 AMI	是 (p. 95) – 如果部署 Amazon EKS 优化版 AMI，更新可用时，Amazon EKS 控制台会通知您。可以在控制台中单击一键执行更新。如果部署自定义 AMI，更新可用时，Amazon EKS 控制台不会通知您。必须自己执行更新。	是 (p. 124) - 使用 Amazon EKS 控制台以外的其他工具。这是因为无法使用 Amazon EKS 控制台管理自行管理的节点。	否
必须自己更新节点 Kubernetes 版本	是 (p. 95) – 如果部署 Amazon EKS 优化版 AMI，更新可用时，Amazon EKS 控制台会通知您。可以在控制台中单击一键执行更新。如果部署自定义 AMI，更新可用时，Amazon EKS 控制台不会通知您。必须自己执行更新。	是 (p. 124) - 使用 Amazon EKS 控制台以外的其他工具。这是因为无法使用 Amazon EKS 控制台管理自行管理的节点。	否 – 您不管理节点。
可以为 pods 使用 Amazon EBS 存储	是 (p. 195)	是 (p. 195)	否
可以为 pods 使用 Amazon EFS 存储	是 (p. 210)	是 (p. 210)	是 (p. 210)
可以为 pods 使用 Amazon FSx for Lustre 存储	是 (p. 220)	是 (p. 220)	否
可以对服务使用 Network Load Balancer	是 (p. 327)	是 (p. 327)	是的，当使用 创建网络负载均衡器 (p. 329) 时
Pod 可以在公有子网中运行	是	是	否
可以将不同的 VPC 安全组分配给各个 pods	是 (p. 273) – 仅限 Linux 节点	是 (p. 273) – 仅限 Linux 节点	是，在版本 `1.18` 或更高版本的集群中
可以运行 Kubernetes DaemonSets	是	是	否
在 pod 清单中支持 `HostPort` 和 `HostNetwork`	是	是	否
Amazon Web Services 区域 可用性	所有 Amazon EKS 支持的区域	所有 Amazon EKS 支持的区域	部分 Amazon EKS 支持的区域 (p. 127)
可以在 EC2 专属主机上运行容器	否	是	否

标准	EKS 托管节点组	自行管理的节点	Amazon Fargate
定价	运行多个 pods 的 Amazon EC2 实例的成本。有关更多信息，请参阅 Amazon EC2定价。	运行多个 pods 的 Amazon EC2 实例的成本。有关更多信息，请参阅 Amazon EC2定价。	单个 Fargate 内存和 CPU 配置的成本。每个 pod 都有自己的成本。有关更多信息，请参阅 Amazon Fargate 定价。

托管节点组

Amazon EKS 托管节点组为 Amazon EKS Kubernetes 集群自动对节点（Amazon EC2 实例）进行预置和生命周期管理。

使用 Amazon EKS 托管节点组，您无需单独预置或注册 Amazon EC2 实例以提供计算容量来运行 Kubernetes 应用程序。您可以通过单个操作为集群创建、自动更新或终止节点。节点的更新和终止操作会自动耗尽节点，以确保您的应用程序保持为可用的状态。

所有托管节点均作为由 Amazon EKS 为您管理的 Amazon EC2 Auto Scaling 组的一部分进行预置。所有资源（包括实例和 Auto Scaling 组在内）都在您的 Amazon 账户中运行。每个节点组跨您定义的多个可用区运行。

您可以使用 Amazon EKS 控制台、`eksctl`、Amazon CLI、Amazon API 或基础设施即代码工具（包括 Amazon CloudFormation），将托管节点组添加到新的或现有集群中。作为托管节点组一部分启动的节点会自动进行标记，以便由 Kubernetes Cluster Autoscaler 自动发现它们。您可以使用节点组将 Kubernetes 标签应用到节点并随时更新它们。

使用 Amazon EKS 托管节点组没有额外的费用，您只需为预置的 Amazon 资源付费。这些资源包括 Amazon EC2 实例、Amazon EBS 卷、Amazon EKS 集群小时数和任何其他 Amazon 基础设施。无最低费用，无预先承诺。

要开始使用新 Amazon EKS 集群和托管节点组，请参阅 开始使用 Amazon EKS – Amazon Web Services Management Console 和 Amazon CLI (p. 15)。

要将托管节点组添加到现有集群，请参阅 创建托管节点组 (p. 90)。

托管节点组概念

- Amazon EKS 托管节点组为您创建和管理 Amazon EC2 实例。
- 所有托管节点均作为由 Amazon EKS 为您管理的 Amazon EC2 Auto Scaling 组的一部分进行预置。此外，包括 Amazon EC2 实例和 Auto Scaling 组在内的所有资源都在您的 Amazon 账户中运行。
- 托管节点组的弹性伸缩组涵盖您在创建组时指定的所有子网。
- Amazon EKS 标记托管节点组资源，以便将其配置为使用 Kubernetes Cluster Autoscaler (p. 72)。

  Important

  如果要使用 Kubernetes Cluster Autoscaler (p. 72) 在由 Amazon EBS 卷支持的多个可用区中运行有状态应用程序，则应该配置多个节点组，每个节点组的范围都限定为一个可用区。此外，您还应该启用 `--balance-similar-node-groups` 功能。
- 在部署托管节点时，可以使用自定义启动模板以获得更大灵活性和自定义性。如果您使用启动模板进行部署，也可以使用自定义 AMI。有关更多信息，请参阅启动模板支持 (p. 100)。如果首次创建托管节点组时没有使用自定义启动模板，则会有自动生成的启动模板。不要手动修改此自动生成的模板，否则会发生错误。

- Amazon EKS 在托管节点组上遵循 CVE 和安全补丁的责任共担模式。当托管节点运行 Amazon EKS 优化版 AMI 时，Amazon EKS 负责在报告 Bug 或问题时构建 AMI 的修补版本。我们可以发布修复程序。但是，您负责将这些修补的 AMI 版本部署到托管节点组。当托管节点运行自定义 AMI 时，您要负责在报告 Bug 或问题时构建 AMI 的修补版本，然后部署 AMI。有关更多信息，请参阅更新托管节点组 (p. 95)。
- Amazon EKS 托管节点组既可以在公有子网也可以在私有子网中启动。如果您在不早于 2020 年 4 月 22 日启动了公有子网中的托管节点组，则子网必须将 `MapPublicIpOnLaunch` 设置为 true，这些实例才能成功加入集群。如果公有子网是使用 `eksctl` 或 Amazon EKS 发布的 Amazon CloudFormation 模板 (p. 228)并在 2020 年 3 月 26 日或之后创建的，则此设置已设置为 true。如果在 2020 年 3 月 26 日之前创建了公有子网，则必须手动更改设置。有关更多信息，请参阅修改子网的公有 `IPv4` 寻址属性。
- 在私有子网中使用 VPC 端点时，您必须为 `com.amazonaws.region.ecr.api`、`com.amazonaws.region.ecr.dkr` 创建端点，并为 Amazon S3 创建网关端点。有关更多信息，请参阅 Amazon ECR 接口 VPC 端点 (Amazon PrivateLink)。
- 托管节点组无法部署在 Amazon Outposts (p. 189) 上或者 Amazon Wavelength 或 Amazon Local Zones 中。
- 您可以在单个集群中创建多个托管节点组。例如，您可以为某些工作负载使用标准的 Amazon EKS 优化版 Amazon Linux 2 AMI 创建一个节点组，为需要 GPU 支持的工作负载使用 GPU 变体创建另一个节点组。
- 如果您的托管节点组遇到 Amazon EC2 实例状况检查失败，则 Amazon EKS 返回错误消息以帮助您诊断问题。有关更多信息，请参阅托管节点组错误 (p. 479)。
- Amazon EKS 将 Kubernetes 标签添加到托管节点组实例。Amazon EKS 提供的这些标签带有前缀 `eks.amazonaws.com`。
- 在终止或更新期间，Amazon EKS 使用 Kubernetes API 自动耗尽节点。更新操作遵循您为 pods 设置的 pod 中断预算。
- 使用 Amazon EKS 托管节点组不会产生额外的费用。您仅需为预置的Amazon资源付费。
- 如果要为节点加密 Amazon EBS 卷，可以使用启动模板部署节点。要在不使用启动模板的情况下部署包含加密 Amazon EBS 卷的托管节点，必须对账户中创建的所有新 Amazon EBS 卷进行加密。有关更多信息，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的默认加密。

托管节点组容量类型

创建托管节点组时，您可以选择按需容量或 Spot 容量类型。Amazon EKS 使用 Amazon EC2 Auto Scaling 组部署托管节点组，该组只包含按需实例或只包含 Amazon EC2 竞价型实例。您可以将容错应用程序的 pods 调度到 Spot 托管节点组，将容错应用程序调度到单个 Kubernetes 集群中的按需节点组。预设情况下，托管节点组部署按需 Amazon EC2 实例。

按需

使用按需实例，您按秒为计算容量支付费用，无需长期订阅。

工作方式

预设情况下，如果您没有指定容量类型，则会使用按需实例预置托管节点组。托管节点组会代表您配置 Amazon EC2 Auto Scaling 组，并应用以下设置：

- 预置按需容量的分配策略设置为 `prioritized`。托管节点组使用实例类型在 API 中的传入顺序，确定在满足按需容量时首先使用哪种实例类型。例如，您可以按以下顺序指定三种实例类型：`c5.large`、`c4.large` 和 `c3.large`。在启动您的按需实例时，托管节点组满足按需容量的顺序是从 `c5.large` 开始，然后是 `c4.large`，再然后是 `c3.large`。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 Amazon EC2 Auto Scaling 组。
- Amazon EKS 会向您指定了容量类型的托管节点组中的所有节点添加以下 Kubernetes 标签：`eks.amazonaws.com/capacityType: ON_DEMAND`。您可以使用此标注在按需节点上调度有状态或不容错的应用程序。

Spot 实例

Amazon EC2 Spot 实例是备用的 Amazon EC2 容量，价格相当于在按需实例价格上打了巨大折扣。当 EC2 需要收回容量时，Amazon EC2 Spot 实例可能中断并发出一个两分钟的中断通知。有关更多信息，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的竞价型实例。您可以使用 Amazon EC2 Spot 实例配置托管节点组，以优化 Amazon EKS 集群中运行的计算节点的成本。

工作方式

要在托管节点组中使用 Spot 实例，请将容量类型设置为 `spot`，创建托管节点组。托管节点组使用以下 Spot 最佳实践代表您配置 Amazon EC2 Auto Scaling 组：

- 预置 Spot 容量的分配策略设置为 `capacity-optimized`，以确保您的 Spot 节点在最佳 Spot 容量池中进行配置。要增加可用于分配容量的 Spot 容量池的数量，请将托管节点组配置为使用多个实例类型。
- 启用了 Amazon EC2 Spot 容量再平衡，以便 Amazon EKS 能够顺畅地耗尽和重新平衡您的 Spot 节点，从而在 Spot 节点中断风险增加时最大限度地减少应用程序中断。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 Amazon EC2 Auto Scaling 容量再平衡。
  - Spot 节点收到再平衡建议时，Amazon EKS 会自动尝试启动新的替换 Spot 节点，并等待它成功加入集群。
  - 当替换 Spot 节点引导启动并在 Kubernetes 中处于 `Ready` 状态后，Amazon EKS 将封锁并耗尽收到再平衡建议的 Spot 节点。封锁 Spot 节点可确保服务控制器不会向此 Spot 节点发送任何新请求。它还将其从其运行状况良好的活动 Spot 节点列表中删除。耗尽 Spot 节点可确保正在运行的 pods 被自然地逐出。
  - 如果 Spot 两分钟中断通知在替换 Spot 节点进入 `Ready` 状态之前到达时间，Amazon EKS 将开始耗尽收到再平衡建议的 Spot 节点。
- Amazon EKS 会向您指定了容量类型的托管节点组中的所有节点添加以下 Kubernetes 标签：`eks.amazonaws.com/capacityType: SPOT`。您可以使用此标注在 Spot 节点上安排容错应用程序。

选择容量类型的注意事项

在决定是部署具有按需容量还是 Spot 容量的节点组时，应考虑以下条件：

- Spot 实例适合无状态、容错且灵活的应用程序。其中包括批处理和机器学习培训工作负载、大数据 ETL（例如 Apache Spark）、队列处理应用程序和无状态 API 端点。由于 Spot 是备用 Amazon EC2 容量，这些容量可能会随时间变化，因此我们建议您对可容忍中断的工作负载使用 Spot 容量。更具体地说，Spot 容量适合可以容忍所需容量不可用的时段的工作负载。
- 我们建议您为不容错的应用程序使用 On-Demand。这包括集群管理工具，例如监控和操作工具，需要 `StatefulSets` 的部署，以及有状态的应用程序，如数据库。
- 为了在使用 Spot 实例时最大限度地提高应用程序的可用性，我们建议您将 Spot 托管节点组配置为使用多个实例类型。我们建议在使用多个实例类型时应用以下规则：
  - 在托管节点组中，如果使用 Cluster Autoscaler (p. 72)，我们建议使用具有相同数量 vCPU 和内存资源的灵活实例类型集。这是为了确保集群中的节点按预期进行扩缩。例如，如果需要 4 个 vCPU 和 8 GiB 内存，请使用 `c3.xlarge`、`c4.xlarge`、`c5.xlarge`、`c5d.xlarge`、`c5a.xlarge`、`c5n.xlarge` 或其他类似实例类型。
  - 为了提高应用程序可用性，我们建议部署多个 Spot 托管节点组。为此，每个组应使用一组具有相同 vCPU 和内存资源的灵活实例类型。例如，如果您需要 4 个 vCPU 和 8 GiB 内存，我们建议您使用 `c3.xlarge`、`c4.xlarge`、`c5.xlarge`、`c5d.xlarge`、`c5a.xlarge`、`c5n.xlarge` 或其他类似的实例类型创建一个托管节点组，并使用 `m3.xlarge`、`m4.xlarge`、`m5.xlarge`、`m5d.xlarge`、`m5a.xlarge`、`m5n.xlarge` 或其他类似的实例类型创建第二个托管节点组。
  - 用使用自定义启动模板的 Spot 容量类型部署节点组时，请使用 API 传递多个实例类型。不要通过启动模板传递单个实例类型。有关使用启动模板部署节点组的更多信息，请参阅 启动模板支持 (p. 100)。

创建托管节点组

本主题介绍了如何启动向 Amazon EKS 集群注册的节点的 Amazon EKS 托管节点组。在这些节点加入集群后，您可以向其部署 Kubernetes 应用程序。

如果这是您首次启动 Amazon EKS 托管节点组，建议您改为遵循我们的 开始使用 Amazon EKS (p. 4) 指南之一。这些指南提供了有关创建包含节点的 Amazon EKS 集群的演练。

Important

- Amazon EKS 节点是标准的 Amazon EC2 实例。将根据正常的 Amazon EC2 价格向您计费。有关更多信息，请参阅 Amazon EC2 定价。
- 您无法在启用了 Amazon Outposts、Amazon Wavelength 或 Amazon Local Zones 的 Amazon Web Services 区域 创建托管节点。您可以在启用了 Amazon Outposts 或 Amazon Wavelength 的 Amazon Web Services 区域创建自行管理的节点。有关更多信息，请参阅 启动自行管理的 Amazon Linux 节点 (p. 107) 或 启动自行管理的 Windows 节点 (p. 114)。

先决条件

- 现有 版 Amazon EKS 集群。要部署一个角色，请参阅 创建 Amazon EKS 集群 (p. 21)。
- （可选，但建议设置）Amazon VPC CNI plugin for Kubernetes 附加组件已配置自己的 IAM 角色，并附加了必要的 IAM policy。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。
- 熟悉 选择 Amazon EC2 实例类型 (p. 145) 中所列出的注意事项。根据您选择的实例类型，您的集群和 VPC 可能还有其他先决条件。

您可以使用 `eksctl` 或 Amazon Web Services Management Console 创建托管节点组。

eksctl

使用 **`eksctl`** 创建托管节点组

此过程需要 `eksctl` 版本 `0.110.0` 或更高版本。可以使用以下命令来查看您的版本：

```
eksctl version
```

有关如何安装或升级 `eksctl` 的说明，请参阅 安装或更新 `eksctl` (p. 10)。

1. （可选）如果 AmazonEKS_CNI_Policy 托管 IAM policy 附加到您的 the section called “节点 IAM 角色” (p. 427)，我们建议将其分配给您与 Kubernetes `aws-node` 服务账户关联的 IAM 角色。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。
2. 使用或不使用自定义启动模板创建托管节点组。手动指定启动模板可允许对节点组进行更好的自定义。例如，它可以允许部署自定义 AMI 或向 Amazon EKS 优化的 AMI 中的 `boostrap.sh` 脚本提供参数。要查看所有可用选项和原定设置的完整列表，请输入以下命令。

   ```
   eksctl create nodegroup --help
   ```

   在以下命令中，将 *`my-cluster`* 替换为您的集群名称，并将 *`my-mng`* 替换为您的节点组名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。将剩余的 *`example values`* 替换为您自己的值。

Important

如果首次创建托管节点组时没有使用自定义启动模板，则以后不要对节点组使用模板。如果没有指定自定义启动模板，系统会自动生成启动模板，我们不建议您手动修改该模板。手动修改此自动生成的启动模板可能会导致错误。

- 不使用启动模板 – `eksctl`在您的账户中创建默认 Amazon EC2 启动模板，并使用它根据您指定的选项创建的启动模板来部署节点组。在为 `--node-type` 指定值之前，请参阅 选择 Amazon EC2 实例类型 (p. 145)。

  将 *my-key* 替换为您的 Amazon EC2 密钥对或公有密钥的名称。此密钥用于在节点启动后通过 SSH 进入节点。如果还没有 Amazon EC2 密钥对，可以在 Amazon Web Services Management Console 中创建一个。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 Amazon EC2 密钥对。

  如果满足以下条件，我们建议阻止容器组（pod）访问 IMDS：

  - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便容器组（pod）只具有他们所需的最低权限。
  - 集群中没有任何容器组（pod）需要出于其他原因（例如检索当前 Amazon Web Services 区域）访问 Amazon EC2 实例元数据服务（IMDS）。

  有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

  如果要阻止 pod 对 IMDS 的访问，请将 **`--disable-pod-imds`** 选项添加到以下命令。

  ```
  eksctl create nodegroup \
    --cluster my-cluster \
    --region region-code \
    --name my-mng \
    --node-type m5.large \
    --nodes 3 \
    --nodes-min 2 \
    --nodes-max 4 \
    --ssh-access \
    --ssh-public-key my-key
  ```

  您的实例可以选择性地为 pods 分配更多的 IP 地址，为其他 CIDR 块（而不是实例的 CIDR 块）中的 pods 分配 IP 地址，以及部署到没有网络访问权限的集群。有关更多信息，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量 (p. 270)、教程：自定义联网 (p. 258) 和 私有集群要求 (p. 56)，以获取要添加到上一个命令中的其他选项。

  托管节点组将根据实例类型计算并应用单个值，以作为可以在节点组的每个节点上运行的最大 pods 数量。如果创建具有不同实例类型的节点组，则在所有实例类型中计算得出的最小值将应用为可以在节点组中每种实例类型上运行的最大 pods 数量。托管节点组会使用 Amazon EKS 建议每种 Amazon EC2 实例类型的最大 pods 数量 (p. 146) 中引用的脚本计算值。
- 使用启动模板 – 启动模板必须已存在，并且必须满足 启动模板配置基础知识 (p. 100) 中指定的要求。

  如果满足以下条件，我们建议阻止容器组（pod）访问 IMDS：

  - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便容器组（pod）只具有他们所需的最低权限。
  - 集群中没有任何容器组（pod）需要出于其他原因（例如检索当前 Amazon Web Services 区域）访问 Amazon EC2 实例元数据服务（IMDS）。

  有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

  如果要阻止 pod 访问 IMDS，请在启动模板中指定必要的设置。

a. 将以下内容复制到您的设备。替换*###*，然后运行修改后的命令以创建 `eks-nodegroup.yaml` 文件。在不使用启动模板的情况下进行部署时指定的多个设置将移动到启动模板中。如果未指定 `version`，则使用模板的默认版本。

```
cat >eks-nodegroup.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
- name: my-mng
  launchTemplate:
    id: lt-id
    version: "1"
EOF
```

有关 `eksctl` 配置文件设置的完整列表，请参阅 `eksctl` 文档中的配置文件架构。您的实例可以选择为 pods 分配更多 IP 地址，为其他 CIDR 块（而不是实例的 CIDR 块）中的 pods 分配 IP 地址，使用 `containerd` 运行时间，并将其部署到没有出站网络访问权限的集群。有关更多信息，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量 (p. 270)、教程：自定义联网 (p. 258)、启用 `containerd` 运行时间引导标记 (p. 157) 和 私有集群要求 (p. 56)，以获取添加到配置文件中的其他选项。

如果您没有在启动模板中指定 AMI ID，则托管节点组将根据实例类型计算并应用单个值，以作为可以在节点组的每个节点上运行的最大 pods 数量。如果创建具有不同实例类型的节点组，则在所有实例类型中计算得出的最小值将应用为可以在节点组中每种实例类型上运行的最大 pods 数量。托管节点组会使用 Amazon EKS 建议每种 Amazon EC2 实例类型的最大 pods 数量 (p. 146) 中引用的脚本计算值。

如果您在启动模板中指定了 AMI ID，请指定可以在节点组的每个节点上运行的最大 pods 数量（如果您使用自定义网络 (p. 258)或者想要增加分配到实例的 IP 地址数量 (p. 270)）。有关更多信息，请参阅Amazon EKS 建议每种 Amazon EC2 实例类型的最大 pods 数量 (p. 146)。

b. 使用以下命令部署节点组。

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

Amazon Web Services Management Console

要使用 Amazon Web Services Management Console 创建托管节点组

1. 等待集群状态显示为 `ACTIVE`。无法为状态尚未处于 `ACTIVE` 的集群创建托管节点组。
2. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
3. 选择要在其中创建托管节点组的集群的名称。
4. 选择 Compute（计算）选项卡。
5. 请选择 Add node group（添加节点组）。
6. 在 Configure node group (配置节点组) 页面上，填写相应参数，然后选择 Next (下一步)。
   - 名称 – 为托管节点组输入唯一名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。
   - 节点 IAM 角色 – 选择要与节点组一起使用的节点实例角色。有关更多信息，请参阅Amazon EKS 节点 IAM 角色 (p. 427)。

Important

- 您不能使用创建任何集群时使用的相同角色。
- 我们建议使用任何自行管理节点组当前未使用的角色。否则，计划与新的自行管理节点组配合使用。有关更多信息，请参阅删除托管节点组 (p. 105)。

- 使用启动模板 - （可选）选择是否要使用现有启动模板。选择 Launch Template Name（启动模板名称）。然后，选择 Launch template version（启动模板版本）。如果您未选择版本，Amazon EKS 将使用模板的默认版本。启动模板允许您对节点组进行更多自定义，例如允许您部署自定义 AMI、为 pods 分配更多的 IP 地址、将 IP 地址分配到其他 CIDR 块（而不是实例的 CIDR 块）中的 pods、为实例启用 `containerd` 运行时间，并将节点部署到没有出站网络访问权限的集群。有关更多信息，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量 (p. 270)、教程：自定义联网 (p. 258)、启用 `containerd` 运行时间引导标记 (p. 157)和 私有集群要求 (p. 56)。

  启动模板必须满足 启动模板支持 (p. 100) 中的要求。如果您不使用自己的启动模板，Amazon EKS API 会在您的账户中创建默认 Amazon EC2 启动模板，并使用默认启动模板部署节点组。

  如果实施服务账户的 IAM 角色 (p. 395)，请将必要的权限直接分配到需要访问 Amazon 服务的所有 pod，如果集群中没有 pods 因其他原因（例如检索当前 Amazon Web Services 区域）而需要访问 IMDS，那么您还可以在启动模板中为不使用主机网络的 pods 禁用对 IMDS 的访问。有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。
- Kubernetes 标签 – （可选）您可以选择对托管节点组中的节点应用 Kubernetes 标签。
- Kubernetes 污点 – （可选）您可以选择对托管节点组中的节点应用 Kubernetes 污点。Effect（效果）菜单中的可用选项包括 `NoSchedule`、`NoExecute` 和 `PreferNoSchedule`。
- 标签 – （可选）您可以选择对 Amazon EKS 托管节点组进行标记。这些标签不会传播到节点组中的其他资源，例如 弹性缩放组或实例。有关更多信息，请参阅为您的 Amazon EKS 资源添加标签 (p. 385)。
- 节点组更新配置 – （可选）您可以选择要并行更新的节点的数量或百分比。选择 Number（数字）或者 Percentage（百分比）以输入值。更新期间，这些节点将不可用。

7. 在 Set compute and scaling configuration（设置计算和扩展配置）页面上，填写相应参数，然后选择 Next（下一步）。

   - AMI 类型 – 为 Linux 非 GPU 实例选择 Amazon Linux 2 (AL2_x86_64)，为 Linux GPU 实例选择 Amazon Linux 2 GPU 已启用 (AL2_x86_64_GPU)、为 Linux Arm 实例选择 Amazon Linux 2 Arm (AL2_ARM_64)，为 Bottlerocket x86_64 实例选择 Bottlerocket (BOTTLEROCKET_x86_64)，或为 Bottlerocket Arm 实例选择 Bottlerocket Arm (BOTTLEROCKET_ARM_64)。

     如果您要部署 Arm 实例，请务必在部署前查看 Amazon EKS 优化版 Arm Amazon Linux AMI (p. 159) 中的注意事项。

     如果您在上一页指定了启动模板，并在启动模板中指定了 AMI，则无法选择值。此时将显示模板中的值。模板中指定的 AMI 必须满足 指定 AMI (p. 103) 中的要求。
   - 容量类型 – 选择容量类型。有关选择容量类型的更多信息，请参阅 托管节点组容量类型 (p. 88)。不能在同一节点组中混合使用不同的容量类型。如果要同时使用这两种容量类型，请创建单独的节点组，每个节点组都有自己的容量和实例类型。
   - 实例类型 – 默认指定一个或多个实例类型。要删除默认实例类型，请选择实例类型右侧的 `x`。选择要在托管节点组中使用的实例类型。有关更多信息，请参阅选择 Amazon EC2 实例类型 (p. 145)。

     控制台显示一组常用的实例类型。如果需要使用未显示的实例类型创建托管节点组，请使用 `eksctl`、Amazon CLI、Amazon CloudFormation 或 SDK 创建节点组。如果在上一页指定了启动模板，则无法选择值，因为必须在启动模板中指定实例类型。将显示启动模板中的值。如果为容量类型选择了 Spot 实例，我们建议您指定多个实例类型以增强可用性。
   - 磁盘大小 – 输入要用于节点根卷的磁盘大小（单位为 GiB）。

     如果在上一页指定了启动模板，则无法选择值，因为必须在启动模板中指定该值。

- 最小大小 – 指定托管节点组可以横向缩减到的最小节点数量。
- 最大大小 – 指定托管节点组可以横向扩展到的最大节点数量。
- 所需大小 – 指定托管节点组在启动时应当维持的当前节点数量。

  Note

  Amazon EKS 不会自动扩展或缩减节点组。但是，您可以配置 Kubernetes Cluster Autoscaler (p. 72) 以为您执行此操作。
- 对于最大不可用，选择下列选项之一，然后指定一个值：
  - Number（数字）– 选择并指定节点组中可以并行更新的节点数。这些节点在更新过程中将不可用。
  - Percentage（百分比）– 选择并指定节点组中可并行更新的节点的百分比。这些节点在更新过程中将不可用。如果您的节点组中有大量节点，这将非常有用。

8. 在 Specify networking（指定联网）页面上，相应填写参数，然后选择 Next（下一步）。
   - 子网 – 选择要在其中启动托管节点的子网。

     Important

     如果要使用 Kubernetes Cluster Autoscaler (p. 72) 在由 Amazon EBS 卷支持的多个可用区中运行有状态应用程序，则应该配置多个节点组，每个节点组的范围都限定为一个可用区。此外，您还应该启用 `--balance-similar-node-groups` 功能。

     Important

     - 如果您选择公有子网，并且您的集群仅启用公有 API 服务器端点，则子网必须将 `MapPublicIPOnLaunch` 设置为 `true`，实例才能成功加入集群。如果子网是使用 `eksctl` 或 Amazon EKS 发布的 Amazon CloudFormation 模板 (p. 228)在 2020 年 3 月 26 日或之后创建的，则此设置已设置为 `true`。如果子网是在 2020 年 3 月 26 日之前使用 `eksctl` 或 Amazon CloudFormation 模板创建的，则需要手动更改设置。有关更多信息，请参阅修改子网的公有 `IPv4` 寻址属性。
     - 如果使用启动模板并指定多个网络接口，即使 `MapPublicIpOnLaunch` 设置 `true`，Amazon EC2 也不会自动分配公有 `IPv4` 地址。在这种情况下，要让节点加入集群，您必须启用集群的私有 API 服务器端点，或者在具有出站 Internet 访问的私有子网中启动节点（Internet 访问通过如 NAT 网关等其他方法提供）。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 Amazon EC2 实例 IP 寻址。
   - Configure SSH access to nodes（配置对节点的 SSH 访问）（可选）。启用 SSH 后，如果出现问题，您可以连接到实例并收集诊断信息。请完成以下步骤以启用远程访问。我们强烈建议您在创建节点组时启用远程访问。创建节点组后，将无法启用远程访问。

     如果您选择使用启动模板，则不会显示此选项。要启用对节点的远程访问，请在启动模板中指定密钥对，并确保为您在启动模板中指定的安全组中的节点打开正确的端口。有关更多信息，请参阅使用自定义安全组 (p. 102)。
   - 对于 SSH 密钥对（可选），请选择要使用的 Amazon EC2 SSH 密钥。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 Amazon EC2 密钥对。如果您选择使用启动模板，则无法选择密钥对。使用 Bottlerocket AMI 为节点组提供 Amazon EC2 SSH 密钥后，还启用管理容器。有关更多信息，请参阅 GitHub 上的 Admin 容器。
   - 对于 Allow SSH remote access from（允许来自以下的远程访问），如果要限制对特定实例的访问，请选择与这些实例关联的安全组。如果没有选择特定的安全组，则允许从 Internet 上的任何位置进行 SSH 访问（`0.0.0.0/0`）。
9. 在 Review and create (审核并创建) 页面上，审核托管节点组配置并选择 Create (创建)。

   如果节点无法加入集群，请参阅《故障排除指南》中的 节点未能加入集群 (p. 478)。
10. 查看节点的状态并等待它们达到 `Ready` 状态。

```
kubectl get nodes --watch
```

11. （仅限 GPU 节点）如果选择 GPU 实例类型和 Amazon EKS 优化版加速型 AMI，则必须使用以下命令在集群上将适用于 Kubernetes 的 NVIDIA 设备插件用作 DaemonSet。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/nvidia-device-plugin.yml
```

12. （可选）将 Linux 节点添加到集群后，请按照 为 Amazon EKS 集群启用 Windows 支持 (p. 48) 中的步骤向集群添加 Windows 支持并添加 Windows Worker 节点。所有 Amazon EKS 集群必须至少包含一个 Linux 节点，即使您只想在集群中运行 Windows 工作负载也是如此。

现在，您已有使用节点运行的 Amazon EKS 集群，那么就可以准备开始安装 Kubernetes 插件并将应用程序部署到您的集群。以下文档主题可帮助您扩展集群的此功能。

- 创建集群的 IAM 实体（用户或角色）是唯一可以使用 `kubectl` 或 Amazon Web Services Management Console 调用 Kubernetes API 服务器的 IAM 用户。如果您希望其他 IAM 用户或角色拥有访问您集群的权限，您需要添加它们。有关更多信息，请参阅 让 IAM 用户和角色有权访问您的集群 (p. 355) 和 所需权限 (p. 458)。
- 如果满足以下条件，我们建议阻止容器组（pod）访问 IMDS：
  - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便容器组（pod）只具有他们所需的最低权限。
  - 集群中没有任何容器组（pod）需要出于其他原因（例如检索当前 Amazon Web Services 区域）访问 Amazon EC2 实例元数据服务（IMDS）。

  有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。
- Cluster Autoscaler (p. 72) – 配置 Kubernetes Cluster Autoscaler 以自动调整节点组中的节点数。
- 将示例应用程序 (p. 315)部署到您的集群。
- 集群管理 (p. 374) – 了解如何使用重要工具来管理集群。

更新托管节点组

启动托管节点组更新后，Amazon EKS 会自动更新您的节点，完成 托管节点更新行为 (p. 97) 中列出的步骤。如果您使用的是 Amazon EKS 优化版 AMI，Amazon EKS 会自动将最新的安全补丁和操作系统更新应用到您的节点，作为最新 AMI 版本的一部分。

在几种情况下，需要更新 Amazon EKS 托管节点组的版本或配置：

- 您已更新 Amazon EKS 集群的 Kubernetes 版本，并且您希望更新节点以使用相同 Kubernetes 版本。
- 新 AMI 发行版本可用于您的托管节点组。有关 AMI 版本的更多信息，请参阅以下章节：
  - Amazon EKS 优化版 Amazon Linux AMI 版本 (p. 160)
  - Amazon EKS 优化版 Bottlerocket AMI (p. 162)
  - Amazon ECS 优化版 Windows AMI 版本 (p. 181)
- 您希望调整托管节点组中的最少、最多或所需实例数。
- 您希望对托管节点组中的实例添加或删除 Kubernetes 标签。
- 您希望对托管节点组添加或删除 Amazon 标签。
- 您需要部署包含配置更改（如更新的自定义 AMI）的最新版本启动模板。
- 您已部署了版本 `1.9.0` 或更高版本的 Amazon VPC CNI 附加组件，启用了前缀委派附加组件，并希望节点组中的新 Amazon Nitro System 实例支持显著增加的 pods 数量。有关更多信息，请参阅提高 Amazon EC2 节点的可用 IP 地址数量 (p. 270)。

如果相比您的托管节点组的 Kubernetes 版本，有较新的 AMI 发行版，则可以将您的节点组的版本更新为使用较新的 AMI 版本。同样，如果您的集群运行的 Kubernetes 版本比节点组更新，则可以将节点组更新为使用与集群的 Kubernetes 版本匹配的最新 AMI 发行版。

如果托管节点组中的节点因扩缩操作或更新而终止，将会先耗尽该节点中的 pods。有关更多信息，请参阅托管节点更新行为 (p. 97)。

更新节点组版本

您可以使用 `eksctl` 或 Amazon Web Services Management Console 更新节点组。请选择包含要用于更新节点组的工具名称的选项卡。您更新到的版本不能高于控制面板的版本。

eksctl

使用 **`eksctl`** 更新节点组版本

- 使用以下命令，将托管节点组更新到节点上当前部署的相同 Kubernetes 版本的最新 AMI 版本。将每个 *`example value`* 替换为您自己的值。

```
eksctl upgrade nodegroup --name=node-group-name --cluster=my-cluster
```

> **Note**
>
> 如果要将使用启动模板部署的节点组升级到新的启动模板版本，请将 `--launch-template-version` *`version-number`* 添加到上述命令。启动模板必须满足 启动模板支持 (p. 100) 中的要求。如果启动模板包含自定义 AMI，则 AMI 必须满足 指定 AMI (p. 103) 中的要求。将节点组升级到启动模板的更新版本后，将回收所有节点，以匹配指定的启动模板版本的新配置。
> 您不能将未使用启动模板部署的节点组直接升级到新的启动模板版本。相反，您必须使用启动模板部署新的节点组，以将节点组更新为新的启动模板版本。

您可以将节点组升级到与控制面板的 Kubernetes 版本相同的版本。例如，如果您的集群运行 Kubernetes `1.23`，则您可以使用以下命令将当前运行 Kubernetes `1.22` 的节点升级到版本 `1.23`。

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --kubernetes-version=1.23
```

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 更新节点组版本

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择包含要更新的节点组的集群。
3. 如果至少有一个节点组具有可用更新，则页面顶部将出现一个提示框，通知存在可用的更新。如果选择 Compute（计算）选项卡，则在具有可用更新的节点组的 Node Groups（节点组）表中，AMI release version（AMI 版本）列会显示 Update now（立即更新）。要更新节点组，请选择 Update now（立即更新）。对于使用自定义 AMI 部署的节点组，不会显示通知。如果您的节点是使用自定义 AMI 部署的，请完成以下步骤以部署更新的自定义 AMI。
   - a. 创建 AMI 的新版本。
     b. 使用新 AMI ID 创建新的启动模板版本。
     c. 将节点升级到新版本的启动模板。
4. 在 Update Node Group version（更新节点组版本）页面上，选择：

- Update Node Group version（更新节点组版本）– 如果您部署了自定义 AMI，或者您的 Amazon EKS 优化版 AMI 当前位于集群的最新版本上，则此选项不可用。
- Launch template version（启动模板版本）– 如果部署节点组时没有使用自定义启动模板，则此选项不可用。您只能更新已使用自定义启动模板部署的节点组的启动模板版本。选择要将节点组更新到的版本。如果您的节点组配置了自定义 AMI，则您选择的版本还必须指定 AMI。升级到更新版本的启动模板后，所有节点都会被回收，以匹配指定的启动模板版本的新配置。

5. 对于 Update strategy（更新策略），选择下列选项之一，然后选择 Update（更新）。

   - Rolling update（滚动更新）– 此选项会考虑集群的 pod 中断预算。如果 pod 中断预算问题导致 Amazon EKS 无法正常耗尽在此节点组上运行的 pods，则更新将失败。
   - Force update（强制更新）– 此选项不考虑 pod 中断预算。通过强制节点重新启动，无论是否存在 pod 中断预算问题，都会进行更新。

编辑节点组配置

您可以修改托管节点组的某些配置。

编辑节点组配置

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择包含要编辑的节点组的集群。
3. 选择 Compute（计算）选项卡。
4. 选择要编辑的节点组，然后选择 Edit (编辑)。
5. （可选）在 Edit node group（编辑节点组）页面上，编辑 Group configuration（组配置）。

   - 标签 – 向节点组资源添加标签或从中删除标签。这些标签仅应用于 Amazon EKS 节点组。这些标签不会传播到其他资源，例如节点组中的子网或 Amazon EC2 实例。
   - Kubernetes 标签 – 向节点组中的节点添加或删除 Kubernetes 标签。此处显示的标签仅为已应用于 Amazon EKS 的标签。节点上可能存在此处未显示的其他标签。
   - Kubernetes 污点 – 向节点组中的节点添加或删除 Kubernetes 污点。添加的污点可以具有 `No_Schedule`、`Prefer_No_Schedule` 或 `No_Execute` 效果。

6. （可选）在 Edit node group（编辑节点组）页面上，编辑 Group size（组大小）。

   - 最小大小 – 指定托管节点组可以横向缩减到的最小节点数量。
   - 最大大小 – 指定托管节点组可以横向扩展到的最大节点数量。有关节点组中支持的最大节点数，请参阅 Amazon EKS 服务配额 (p. 389)。
   - 所需大小 – 指定托管节点组应当维持的当前节点数量。

7. （可选）编辑 Node Group update configuration（节点组更新配置）。选择 Number（数字）或 Percentage（百分比）。

   - Number（数字）– 选择并指定节点组中可以并行更新的节点数。这些节点在更新过程中将不可用。
   - Percentage（百分比）– 选择并指定节点组中可并行更新的节点的百分比。这些节点在更新过程中将不可用。如果您的节点组中有大量节点，这将非常有用。

8. 编辑完成后，选择 Save changes（保存更改）。

托管节点更新行为

Amazon EKS 托管工作节点升级策略有四个不同的阶段，将在以下各节中介绍。

设置阶段

设置阶段有以下步骤：

1. 为与您的节点组关联的 Auto Scaling 组创建新的 Amazon EC2 启动模板版本。新的启动模板版本使用目标 AMI 或客户提供的启动模板版本进行更新。
2. 对弹性缩放组进行更新以使用最新启动模板版本。
3. 使用节点组的 `updateConfig` 属性确定要并行升级的节点最大数量。最大不可用的配额为 100 个节点。原定设置值为一个节点。有关更多信息，请参阅 Amazon EKS API 参考中的 `updateConfig` 属性。

扩展阶段

升级托管节点组中的节点时，已升级的节点将在与正在升级的节点在相同可用区中启动。为了保证这一点，我们使用 Amazon EC2 的可用区重新平衡。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南的可用区重新平衡。为了满足此要求，将为托管节点组中的每个可用区启动最多两个实例。

扩展阶段有以下步骤：

1. 增加 Auto Scaling 组的最大大小和所需大小，增加幅度为以下两者中的较大者：
   - 部署 Auto Scaling 组可用区数量的最多两倍。
   - 升级不可用的最大值。

     例如，如果节点组有五个可用区，`maxUnavailable` 为 1，则升级过程可以启动最多 10 个节点。但是如果 `maxUnavailable` 为 20（或者任何大于 10 的值），则该过程将启动 20 个新节点。
2. 扩展 Auto Scaling 组后，将检查节点组中是否存在使用最新配置的节点。只有在满足以下标准时，此步骤才会成功：
   - 在节点所在的每个可用区中启动至少一个新节点。
   - 每个新节点都应该处于 `Ready` 状态。
   - 新节点应该有应用 Amazon EKS 的标签。

     以下是常规节点组中的工作节点上应用 Amazon EKS 的标签：
     - `eks.amazonaws.com/nodegroup-image=`*`$amiName`*
     - `eks.amazonaws.com/nodegroup=`*`$nodeGroupName`*

     以下是自定义启动模板或 AMI 节点组中的工作节点上应用 Amazon EKS 的标签：
     - `eks.amazonaws.com/nodegroup-image=`*`$amiName`*
     - `eks.amazonaws.com/nodegroup=`*`$nodeGroupName`*
     - `eks.amazonaws.com/sourceLaunchTemplateId=`*`$launchTemplateId`*
     - `eks.amazonaws.com/sourceLaunchTemplateVersion=`*`$launchTemplateVersion`*
3. 在没有最新标签的节点组的每个节点应用 `eks.amazonaws.com/nodegroup=unschedulable:NoSchedule` 污点。这样可以防止已在先前失败更新中更新了的节点被污染。

下面是导致这一阶段 `NodeCreationFailure` 出现错误的已知原因：

- 可用区中容量不足 - 可用区可能没有请求的实例类型的容量。建议在创建托管节点组时配置多种实例类型。
- 客户达到账户的 EC2 实例限制 - 可能需要使用 Service Quotas 增加账户可以同时运行的 Amazon EC2 实例数量。有关更多信息，请参阅适用于 Linux 实例的 Amazon Elastic Compute Cloud 用户指南中的 EC2 服务限额。
- 自定义用户数据 - 自定义用户数据有时会中断引导过程。这种情况可能导致节点不启动 `kubelet`，或者节点上没有获得预期的 Amazon EKS 标签。有关处理自定义 LT/AMI 的更多信息，请参阅 指定 AMI (p. 103)。
- 任何导致节点运行状况不佳或未准备就绪的更改 - 节点磁盘压力、内存压力和类似的情况可能导致节点不进入 `Ready` 状态。

升级阶段

升级阶段有以下步骤：

1. 随机选择一个节点，最多为该节点组配置的最大不可用性。
2. 耗尽节点的 pods。如果 pods 在 15 分钟内没有离开节点并且没有强制标志，则升级阶段将失败并显示 `PodEvictionFailure` 错误。对于这种情况，您可以使用 `update-nodegroup-version` 请求应用强制标志来删除 pods。
3. 每个 pod 被驱逐后，封锁节点并等待 60 秒钟。这样做是为了防止服务控制器向此节点发送任何新请求，并将此节点从活动节点列表中删除。
4. 向封锁节点的弹性缩放组发送终止请求。
5. 重复以前的升级步骤，直到节点组中没有使用早期版本启动模板部署的节点。

下面是导致此阶段出现 `PodEvictionFailure` 错误的已知原因：

- 积极的 PDB - 积极的 PDB 在 pod 上定义，或者有多个 PDB 指向同一个 pod。
- 容忍所有污点的部署 – 一旦每个 pod 被逐出，预计该节点将为空，因为该节点在前面的步骤中受到污染。但是，如果部署容忍每个污点，则该节点更有可能是非空的，导致 pod 驱逐失败。

缩减阶段

缩减阶段将 Auto Scaling 组的最大大小和所需大小减小一，返回更新开始之前的值。

如果升级工作流确定 Cluster Autoscaler 在工作流缩减阶段扩展节点组，则会立即退出，而不会使节点组恢复原始大小。

托管节点组上的节点污点

Amazon EKS 支持通过托管节点组配置 Kubernetes 污点。污点和容差能力协同工作，以确保不会将 pods 安排在不适当的节点上。

可以为一个节点应用一个或多个 Pod。这标志着节点不应该接受任何不容忍污点的 pods。对 pods 应用容忍度，并允许（但不要求）将 pods 安排到具有匹配污点的节点上。

可以使用Amazon Web Services Management Console或通过 Amazon EKS API，将 Kubernetes 节点污点应用于新的和现有的托管节点组。

以下是使用 Amazon CLI 创建带有污点的节点组的示例：

```
aws eks create-nodegroup \
 --cli-input-json '
{
  "clusterName": "my-cluster",
  ...
  "taints": [
     {
         "key": "dedicated",
         "value": "gpuGroup",
         "effect": "NO_SCHEDULE"
     }
   ],
}'
```

有关污点和容忍度的更多信息，请参阅 Kubernetes 文档。有关更多信息和用法示例，请参阅 Kubernetes 参考文档。

Note

- 一个节点组最多允许 50 个污点。
- 创建节点组后，可以使用 `UpdateNodegroupConfig` API 更新污点。
- 污点键必须以字母或数字开头。可以包含字母、数字、连字符 (-)、句点 (.) 和下划线。最长可为 63 个字符
- 污点键也可以以 DNS 子域前缀和单个 `/` 开头。如果它以 DNS 子域前缀开头，则长度可以为 253 个字符。
- 值是可选的，必须以字母或数字开头。可以包含字母、数字、连字符 (-)、句点 (.) 和下划线 (_)。最长可为 63 个字符
- 效果必须是 `No_Schedule`、`Prefer_No_Schedule` 或 `No_Execute` 之一。

启动模板支持

托管节点组始终部署用于 Amazon EC2 Auto Scaling 组的启动模板。Amazon EKS API 可以通过复制您提供的启动模板，或用账户中的默认值自动创建启动模板，创建此启动模板。要获得最高级别的自定义，您可以使用自己的启动模板和自定义 AMI 部署托管节点。我们建议不要修改自动生成的启动模板。因此，如果想要更加灵活，请务必在首次创建托管节点组时指定自定义节点组。

使用自己的启动模板部署托管节点组后，请使用同一启动模板的不同版本对其进行更新。将节点组更新为启动模板的其他版本时，将回收组中的所有节点，以匹配指定启动模板版本的新配置。无法直接更新不使用自定义启动模板的现有节点组。相反，您必须使用自定义启动模板创建新节点组才能执行此操作。

启动模板配置基础知识

您可以使用 Amazon Web Services Management Console、Amazon CLI 或Amazon开发工具包创建 Amazon EC2 Auto Scaling 启动模板。有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的为自动扩缩组创建启动模板。启动模板中的某些设置类似于用于托管节点配置的设置。使用启动模板部署或更新节点组时，必须在节点组配置或启动模板中指定某些设置。不要同时指定两个位置。如果某个设置出现在错误的位置，则创建或更新节点组之类的操作将失败。

下表列出启动模板中禁止的设置，还列出托管节点组配置中所需的类似设置（如果有）。列出的设置是显示在控制台中的设置。它们在 Amazon CLI 和开发工具包中可能具有相似但不同的名称。

启动模板 - 已禁止	Amazon EKS 节点组配置
Advanced details（高级详细信息）下的 IAM instance profile（IAM 实例配置文件）	Configure Node group（配置节点组）页面上 Node group configuration（节点组配置）下的 Node IAM role（节点 IAM 角色）
Network interfaces（网络接口）（Add network interface（添加网络接口））下的Subnet（子网）	Specify networking（指定联网）页面上 Node group network configuration（节点组网络配置）下的 Subnets（子网）
Advanced details（高级详细信息）下的 Shutdown behavior（关机行为）和 Stop - Hibernate behavior（停止 - 休眠行为）。在启动模板中，对于两个设置都保留默认的不包含在启动模板设置中。	无等效项 Amazon EKS 必须控制实例生命周期，而不是 Auto Scaling 组。

下表列出托管节点组配置中禁止的设置，还列出启动模板中所需的类似设置（如果有）。列出的设置是显示在控制台中的设置。它们在 Amazon CLI 和开发工具包中可能有类似的名称。

Amazon EKS 节点组配置 – 已禁止	启动模板
（仅当您在启动模板中指定了自定义 AMI 时）Set compute and scaling configuration（设置计算和扩缩配置）页面上 Node group compute configuration（节点组计算配置）下的 AMI type（AMI 类型）– 控制台显示 Specified in launch template（已在启动模板中指定）和指定的 AMI ID。 如果启动模板中没有指定 AMI 类型，则可以在节点组配置中选择 AMI。	Launch template contents（启动模板内容）下的 AMI – 如果您有以下任一要求，则必须指定 ID： • 使用自定义 AMI。如果您指定的 AMI 不满足 指定 AMI (p. 103) 中列出的要求，节点组部署将失败。 • 需要提供用户数据以将实际参数传递给随 Amazon EKS 优化版 AMI 一起提供的 `bootstrap.sh` 文件。可以允许实例为 pods 分配明显更多的 IP 地址，为其他 CIDR 块（而不是实例的 CIDR 块）中的 pods 分配 IP 地址，启用 `containerd` 运行时，或将其部署到没有出站网络访问权限的私有集群。有关更多信息，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量 (p. 270)、教程：自定义联网 (p. 258)、启用 `containerd` 运行时间引导标记 (p. 157)、私有集群要求 (p. 56) 和指定 AMI (p. 103)。
Set compute and scaling configuration（设置计算和扩展配置）页面上 Node group compute configuration（节点组计算配置）下的 Disk size（磁盘大小）– 控制台显示 Specified in launch template（已在启动模板中指定）。	Storage (Volumes)（存储（卷））（Add new volume（添加新卷））下的 Size（大小）。您必须在启动模板中指定此项。
Specify Networking（指定网络）页面上 Node group configuration（节点组配置）下的 SSH key pair（SSH 密钥对）– 控制台显示在启动模板中指定的密钥或显示 Not specified in launch template（未在启动模板中指定）。	Key pair (login)（密钥对（登录））下的 Key pair name（密钥对名称。
在使用启动模板时，您无法指定允许远程访问的源安全组。	实例的 Network settings（网络设置）下的 Security groups（安全组），或者 Network interfaces（网络接口）（Add network interface（添加网络接口））下的 Security groups（安全组），但不能同时兼具。有关更多信息，请参阅使用自定义安全组 (p. 102)。

Note

- 如果使用启动模板部署节点组，请在启动模板的 Launch template contents（启动模板内容）下指定 0 或 1 个 Instance type（实例类型）。您也可以为控制台 Set compute and scaling configuration（设置计算和扩缩配置）页面的 Instance types（实例类型）指定 0-20 个实例类型。或者，您可以使用其他使用 Amazon EKS API 的工具来执行此操作。如果您在启动模板中指定实例类型，并使用该启动模板部署节点组，则无法在控制台中或通过使用 Amazon EKS API 的其他工具指定任何实例类型。如果没有在启动模板、控制台中或通过使用 Amazon EKS API 的其他工具指定实例类型，则使用 `t3.medium` 实例类型。如果您的节点组使用 Spot 容量类型，我们建议您使用控制台指定多个实例类型。有关更多信息，请参阅托管节点组容量类型 (p. 88)。
- 如果部署到节点组的任何容器使用实例元数据服务版本 2，请确保在启动模板中将Metadata response hop limit（元数据响应跃点限制）设置为 2。有关更多信息，请参阅适用于 Linux 的 Amazon EC2 用户指南 中的实例元数据和用户数据。如果在不使用自定义启动模板的情况下部署托管节点组，则会在默认启动模板中为节点组自动设置此值。

为 Amazon EC2 实例添加标签

您可以使用启动模板的 `TagSpecification` 参数来指定将哪些标签应用于节点组中的 Amazon EC2 实例。调用 `CreateNodegroup` 或 `UpdateNodegroupVersion` API 的 IAM 实体必须具有 `ec2:RunInstances` 和 `ec2:CreateTags` 的权限，并且必须将标签添加到启动模板中。

使用自定义安全组

您可以使用启动模板指定自定义 Amazon EC2 安全组以应用到节点组中的实例。这可以在实例级安全组参数中，也可以作为网络接口配置参数的一部分。但是，您无法创建同时指定实例级安全组和网络接口安全组的启动模板。请考虑以下适用于为托管节点组使用自定义安全组的条件：

- Amazon EKS 仅允许使用包含单个网络接口规范的启动模板。
- 预设情况下，Amazon EKS 将集群安全组 (p. 231)应用于节点组中的实例，以便于节点和控制层面之间的通信。如果您使用前面提到的任一选项在启动模板中指定自定义安全组，则 Amazon EKS 不会添加集群安全组。因此，您必须确保安全组的入站和出站规则启用了与集群端点的通信。如果您的安全组规则不正确，则 worker 节点无法加入集群。有关安全组规则的更多信息，请参阅 Amazon EKS 安全组要求和注意事项 (p. 231)。
- 如果需要对节点组中的实例进行 SSH 访问，请确保包含允许该访问的安全组。

Amazon EC2 用户数据

启动模板包括自定义用户数据的部分。您可以在此部分为节点组指定配置设置，而无需手动创建单个自定义 AMI。有关 Bottlerocket 可用设置的更多信息，请参阅 GitHub 上的使用用户数据。

在启动实例时，您可以使用 `cloud-init` 在启动模板中提供 Amazon EC2 用户数据。有关更多信息，请参阅 cloud-init 文档。您的用户数据可用于执行常见配置操作。其中包括以下操作：

- 包含用户或组
- 安装程序包

启动模板中用于托管节点组的 Amazon EC2 用户数据必须采用 MIME 分段归档格式（用于 Amazon Linux AMI）和 TOML 格式（用于 Bottlerocket AMI）。这是因为您的用户数据与节点加入集群所需的 Amazon EKS 用户数据合并。不要在用户数据中指定任何启动或修改 `kubelet` 的命令。这是作为 Amazon EKS 合并的用户数据的一部分执行的。某些 `kubelet` 参数（例如节点上的设置标签）可以直接通过托管节点组 API 进行配置。

> **Note**
>
> 如果需要高级 `kubelet` 自定义，包括手动启动或传入自定义配置参数，请参阅 指定 AMI (p. 103)。如果在启动模板中指定自定义 AMI ID，Amazon EKS 不会合并用户数据。

以下详细信息提供有关 Amazon Linux 的用户数据部分的更多信息。

Amazon Linux user data

您可以将多个用户数据块合并到一个 MIME 分段文件中。例如，您可以将配置 Docker 进程守护程序的云 Boothook 与安装自定义软件包的用户数据 Shell 脚本合并。MIME 分段文件包含以下组成部分：

- 内容类型和段边界声明 - `Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="`
- MIME 版本声明 - `MIME-Version: 1.0`
- 一个或多个用户数据块，其包含以下组成部分：
  - 开口边界，表示用户数据块的开头 - `--==MYBOUNDARY==`
  - 数据块的内容类型声明：`Content-Type: text/cloud-config; charset="us-ascii"`。有关内容类型的更多信息，请参阅 cloud-init 文档。
  - 用户数据的内容（例如 Shell 命令或 `cloud-init` 指令的列表）。

- 封闭边界，表示 MIME 分段文件的结尾：`--==MYBOUNDARY==--`

以下是 MIME 分段文件的示例，您可以用它来创建您自己的文件。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
echo "Running custom user data script"

--==MYBOUNDARY==--
```

指定 AMI

如果您具有以下任一要求，请在启动模板的 `imageId` 字段中指定一个 AMI ID。选择您对其他信息的要求。

提供用户数据以将实际参数传递给随 Amazon EKS 优化版 AMI 一起提供的 `bootstrap.sh` 文件

Bootstrapping（引导启动）是一个术语，用于描述添加可以在实例启动时运行的命令。您可以使用 `eksctl` 将参数传递给 `bootstrap.sh` 脚本，无需指定启动模板。或者，您可以在启动模板的用户数据部分中指定信息来实现此目标。

eksctl without specifying a launch template

使用以下内容创建名为 *my-nodegroup.yaml* 的文件。将每个 *example value* 替换为您自己的值。`--apiserver-endpoint`、`--b64-cluster-ca` 和 `--dns-cluster-ip` 参数是可选的。但是，定义它们可使 `bootstrap.sh` 脚本避免进行 `describeCluster` 调用。在私有集群设置或频繁扩展节点的集群中，这是非常有用的。有关 `bootstrap.sh` 脚本的更多信息，请参阅 GitHub 上的 `bootstrap.sh` 文件。

- 唯一必需的参数是集群名称（*my-cluster*）。
- 要检索 `ami-`*1234567890abcdef0* 的所需值，您可以使用以下各部分中的表格：
  - Amazon EKS 优化版 Amazon Linux AMI (p. 147)
  - Amazon EKS 优化版 Bottlerocket AMI (p. 162)
  - Amazon EKS 优化版 Windows AMI (p. 169)
- 此示例使用包含在 Amazon EKS 优化版 AMI 中的 `bootstrap.sh` 脚本提供一个 `kubelet` 参数来设置一个自定义 `max-pods` 值。有关选择 *my-max-pods-value* 的帮助，请参阅 Amazon EKS 建议每种 Amazon EC2 实例类型的最大 pods 数量 (p. 146)。
- 要检索您的集群的 *certificate-authority*，请运行以下命令。

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- 要检索您的集群的 *api-server-endpoint*，请运行以下命令。

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- `--dns-cluster-ip` 的值是您的服务 CIDR，末尾为 `.10`。要检索您的集群的 *service-cidr*，请运行以下命令。例如，如果返回值为 `ipv4 10.100.0.0/16`，则您的值为 *10.100.0.10*。

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- 此示例使用 *containerd* 作为运行时创建节点组，但您可以根据需要修改它。

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    instanceType: m5.large
    privateNetworking: true
    disableIMDSv1: true
    labels: { x86-al2-specified-mng }
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster \
        --kubelet-extra-args '--max-pods=my-max-pods-value' \
        --b64-cluster-ca certificate-authority \
        --apiserver-endpoint api-server-endpoint \
        --dns-cluster-ip service-cidr.10 \
        --use-max-pods false \
        --container-runtime containerd
```

对于每一个可用的 eksctl config 文件选项，请参阅 eksctl 文档中的配置文件架构。eksctl 实用程序仍会为您创建启动模板，并使用您在 config 文件中提供的数据填充其用户数据。

使用以下命令创建节点组。

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

User data in a launch template

在启动模板的用户数据部分指定以下信息。将每个 *example value* 替换为您自己的值。--apiserver-endpoint、--b64-cluster-ca 和 --dns-cluster-ip 参数是可选的。但是，定义它们可使 bootstrap.sh 脚本避免进行 describeCluster 调用。在私有集群设置或频繁扩展节点的集群中，这是非常有用的。有关 bootstrap.sh 脚本的更多信息，请参阅 GitHub 上的 bootstrap.sh 文件。

- 唯一必需的参数是集群名称（*my-cluster*）。
- 此示例使用包含在 Amazon EKS 优化版 AMI 中的 bootstrap.sh 脚本提供一个 kubelet 参数来设置一个自定义 max-pods 值。有关选择 *my-max-pods-value* 的帮助，请参阅 Amazon EKS 建议每种 Amazon EC2 实例类型的最大 pods 数量 (p. 146)。
- 要检索您的集群的 *certificate-authority*，请运行以下命令。

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- 要检索您的集群的 *api-server-endpoint*，请运行以下命令。

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- --dns-cluster-ip 的值是您的服务 CIDR，末尾为 .10。要检索您的集群的 *service-cidr*，请运行以下命令。例如，如果返回值为 ipv4 10.100.0.0/16，则您的值为 *10.100.0.10*。

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- 此示例使用 *containerd* 作为运行时创建节点组，但您可以根据需要修改它。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
/etc/eks/bootstrap.sh my-cluster \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --use-max-pods false \
  --container-runtime containerd

--==MYBOUNDARY==--
```

由于特定的安全性、合规性或内部策略要求，运行自定义 AMI

有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 Amazon Machine Images (AMI)。Amazon EKS AMI 构建规范包含用于构建基于 Amazon Linux 2 的自定义 Amazon EKS AMI 的资源和配置脚本。有关更多信息，请参阅 GitHub 上的 Amazon EKS AMI 构建规范。要构建与其他操作系统一起安装的自定义 AMI，请参阅 GitHub 上的 Amazon EKS 示例自定义 AMI。

> **Important**
>
> 指定 AMI 时，Amazon EKS 不会合并任何用户数据。实际上，您要负责提供节点加入集群所需的 `bootstrap` 命令。如果您的节点无法加入集群，那么 Amazon EKS `CreateNodegroup` 和 `UpdateNodegroupVersion` 操作也会失败。

使用托管节点组指定 AMI ID 所涉及的限制和条件如下：

- 您必须创建一个新的节点组，以便在在启动模板中指定 AMI ID 和不指定 AMI ID 之间进行切换。
- 当有较新的 AMI 版本可用时，控制台中不会通知您。要将节点组更新为更新的 AMI 版本，需要使用更新的 AMI ID 创建新版本的启动模板。然后需要使用新的启动模板版本更新节点组。
- 如果您指定 AMI ID，则无法在 API 中设置以下字段：
  - `amiType`
  - `releaseVersion`
  - `version`
- 您无法指定 Windows AMI ID，因为 Windows 不可用在托管节点组中。

删除托管节点组

本主题介绍如何删除 Amazon EKS 托管节点组。

当您删除托管节点组时，Amazon EKS 会首先将 Auto Scaling 组的最小、最大和所需大小设置为零。然后，这会导致节点组缩小。在每个实例终止前，Amazon EKS 将发送一个信号，以耗尽该节点的 pods 并等待几

分钟。如果几分钟后没有耗尽 pods，Amazon EKS 将允许弹性伸缩继续终止实例。终止所有实例后，将删除弹性伸缩组。

Important

如果您删除的托管节点组使用的节点 IAM 角色未由集群中任何其他托管节点组使用，则该角色将从 `aws-auth` ConfigMap (p. 355) 中删除。如果集群中的任何自行托管节点组使用相同的节点 IAM 角色，则自行管理的节点将变为 `NotReady` 状态。此外，集群操作也被中断。您可以将映射重新添加到 ConfigMap，以最大限度地减少中断。

您可以使用 `eksctl` 或 Amazon Web Services Management Console 删除托管节点组。

eksctl

用 **`eksctl`** 删除托管节点组

- 输入以下命令。将每个 *`example value`* 替换为您自己的值。

```
eksctl delete nodegroup \
  --cluster my-cluster \
  --region region-code \
  --name my-mng
```

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 删除托管节点组

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择包含要删除的节点组的集群。
3. 选择 Compute（计算）选项卡。
4. 选择要删除的节点组并选择 Delete (删除)。
5. 在 Delete node group: *`my-mng`*（删除节点组：my-mng）页面，在文本字段中键入节点组的名称，然后选择 Delete（删除）。

自行管理的节点

一个集群包含一个或多个在其上调度了 pods 的 Amazon EC2 节点。Amazon EKS 节点在 Amazon 账户中运行并通过集群 API 服务器端点连接到集群的控制面板。您可以将一个或多个节点部署到节点组中。节点组是一个或多个在 Amazon EC2 Auto Scaling 组中部署的 Amazon EC2 实例。

节点组的所有实例必须具有下列特征：

- 使用相同的实例类型
- 运行相同的 Amazon Machine Image (AMI)
- 使用相同的 Amazon EKS 节点 IAM 角色 (p. 427)

  Note

  您不能使用创建任何集群时使用的相同角色。

一个集群可以包含多个节点组。如果每个节点组满足以前的要求，集群就可以包含具有不同实例类型和主机操作系统的节点组。每个节点组可以包含多个节点。

Amazon EKS 节点是标准的 Amazon EC2 实例。您需要根据 EC2 价格收取费用。有关更多信息，请参阅 Amazon EC2定价。

Amazon EKS 提供一个专用 Amazon Machine Images (AMI)，称为 Amazon EKS 优化版 AMI。AMI 配置为与 Amazon EKS 搭配使用，并包括 Docker、`kubelet` 以及 Amazon IAM 身份验证器。此 AMI 还包含专用引导脚本，从而允许它自动发现并连接到您的集群控制面板。

如果使用 CIDR 块限制对集群的公有端点的访问，建议您还启用私有端点访问。目的是为了让节点可以与集群通信。在未启用私有终端节点的情况下，您指定用于公有访问的 CIDR 块必须包含来自 VPC 的出口源。有关更多信息，请参阅Amazon EKS 集群端点访问控制 (p. 38)。

要向您的 Amazon EKS 集群添加自行管理的节点，请参阅以下主题。如果手动启动自行管理的节点，则向每个节点添加以下标签。有关更多信息，请参阅为单个资源添加和删除标签。如果按照下面指南中的步骤操作，则会将您所需的标签自动添加到节点。

密钥	值
`kubernetes.io/cluster/`*`my-cluster`*	`owned`

有关节点的更多信息（从一般 Kubernetes 角度看），请参阅 Kubernetes 文档中的节点。

主题

- 启动自行管理的 Amazon Linux 节点 (p. 107)
- 启动自行管理的 Bottlerocket 节点 (p. 112)
- 启动自行管理的 Windows 节点 (p. 114)
- 自行管理节点的更新 (p. 119)

启动自行管理的 Amazon Linux 节点

本主题介绍如何启动向 Amazon EKS 集群注册的 Linux 节点的自动扩缩组。在这些节点加入集群后，您可以向其部署 Kubernetes 应用程序。您可以使用 `eksctl` 或 Amazon Web Services Management Console 启动自行管理的 Amazon Linux 2 节点。

先决条件

- 现有 版 Amazon EKS 集群。要部署一个角色，请参阅 创建 Amazon EKS 集群 (p. 21)。如果您在启用了 Amazon Outposts、Amazon Wavelength 或 Amazon 本地区域的Amazon Web Services 区域中拥有子网，则这些子网不得在您创建集群时就已传入。
- （可选，但建议设置）Amazon VPC CNI plugin for Kubernetes 附加组件已配置自己的 IAM 角色，并附加了必要的 IAM policy。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。
- 熟悉 选择 Amazon EC2 实例类型 (p. 145) 中所列出的注意事项。根据您选择的实例类型，您的集群和 VPC 可能还有其他先决条件。

eksctl

先决条件

您的设备或 Amazon CloudShell 上安装了 `0.110.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 安装或更新 `eksctl` (p. 10)。

要使用 **`eksctl`** 启动自行管理的 Linux 节点

1. （可选）如果 AmazonEKS_CNI_Policy 托管 IAM policy 附加到您的 the section called “节点 IAM 角色” (p. 427)，我们建议将其分配给您与 Kubernetes `aws-node` 服务账户关联的 IAM 角色。

有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。

2. 以下命令在现有集群中创建节点组。将 *al-nodes* 替换为您的节点组名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。将 my-cluster 替换为您集群的名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。将剩余的 *example value* 替换为您自己的值。预设情况下，将使用与控制面板相同的 Kubernetes 版本创建节点。

   在为 `--node-type` 选择一个值之前，请查看 选择 Amazon EC2 实例类型 (p. 145)。

   将 *my-key* 替换为您的 Amazon EC2 密钥对或公有密钥的名称。此密钥用于在节点启动后通过 SSH 进入节点。如果还没有 Amazon EC2 密钥对，可以在 Amazon Web Services Management Console 中创建一个。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 Amazon EC2 密钥对。

   使用以下命令创建您的节点组。

   Important

   如果要将节点组部署到 Amazon Outposts、Wavelength 或本地区域子网，还有其他注意事项：

   - 创建集群时，不得传入子网。
   - 您必须使用配置文件创建节点组，指定子网和 `volumeType: gp2`。有关更多信息，请参阅 `eksctl` 文档中的从配置文件创建节点组和 Config 文件架构。

```
eksctl create nodegroup \
  --cluster my-cluster \
  --name al-nodes \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --ssh-access \
  --managed=false \
  --ssh-public-key my-key
```

   部署符合以下条件的节点组：

   - 可以将明显高出默认配置数量的 IP 地址分配给 pods，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量 (p. 270)。
   - 可以将 `IPv4` 地址分配给来自与实例不同的 CIDR 块的 pods，请参阅 教程：自定义联网 (p. 258)。
   - 可以将 `IPv6` 地址分配给容器组（pod）和服务，请参阅 the section called “IPv6” (p. 248)。
   - 使用 `containerd` 运行时，您必须使用 `config` 文件部署节点组。有关更多信息，请参阅启用 `containerd` 运行时间引导标记 (p. 157)。
   - 没有出站互联网访问权限，请参阅 私有集群要求 (p. 56)。

   要查看所有可用选项和默认设置的完整列表，请输入以下命令。

```
eksctl create nodegroup --help
```

   如果节点无法加入集群，请参阅《故障排除指南》中的 节点未能加入集群 (p. 478)。

   输出示例如下。创建节点时会输出几行。输出的最后几行类似于以下示例行。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. 如果满足以下条件，我们建议阻止容器组（pod）访问 IMDS：

   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便容器组（pod）只具有他们所需的最低权限。
   - 集群中没有任何容器组（pod）需要出于其他原因（例如检索当前 Amazon Web Services 区域）访问 Amazon EC2 实例元数据服务（IMDS）。

   有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

Amazon Web Services Management Console

步骤 1：要使用 Amazon Web Services Management Console 启动自行管理 Linux 节点

1. 下载最新版本的 Amazon CloudFormation 模板。

```
curl -o amazon-eks-nodegroup.yaml https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/cloudformation/2020-10-29/amazon-eks-nodegroup.yaml
```

2. 等待集群状态显示为 ACTIVE。如果在集群处于活动状态之前启动节点，则节点将无法向集群注册，您必须重新启动节点。
3. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
4. 选择 Create stack（创建堆栈），然后选择 With new resources (standard)（使用新资源（标准））。
5. 对于 Specify template（指定模板），选择 Upload a template file（上传模板文件），然后选择Choose file（选择文件）。选择您在第 2 步中下载的 amazon-eks-nodegroup.yaml 文件，然后选择 Next（下一步）。
6. 在 Specify stack details（指定堆栈详细信息）页面上，相应填写以下参数，然后选择 Next（下一步）：

   - 堆栈名称：为 Amazon CloudFormation 堆栈选择堆栈名称。例如，您可以将其命名为 ***my-cluster-nodes***。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。
   - ClusterName：输入您在创建 Amazon EKS 集群时使用的名称。此名称必须与集群名称完全匹配，否则您的节点无法加入该集群。
   - ClusterControlPlaneSecurityGroup：从您在创建 VPC (p. 228) 时生成的 Amazon CloudFormation 输出中，选择 SecurityGroups 值。

     以下步骤显示了检索适用组的一种操作。

     1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
     2. 选择集群的名称。
     3. 选择 Networking（联网）选项卡。
     4. 从 ClusterControlPlaneSecurityGroup 下拉列表中选择时使用 Additional security groups（其他安全组）值作为参考。
   - NodeGroupName：输入节点组的名称。稍后您可以使用此名称来标识为您的节点创建的弹性伸缩节点组。
   - NodeAutoScalingGroupMinSize：输入您的节点 Auto Scaling 组可缩减到的最小节点数。
   - NodeAutoScalingGroupDesiredCapacity：输入创建堆栈时要扩展到的所需节点数目。
   - NodeAutoScalingGroupMaxSize：输入您的节点 Auto Scaling 组可横向扩展到的最大节点数。

- NodeInstanceType：选择节点的实例类型。有关更多信息，请参阅选择 Amazon EC2 实例类型 (p. 145)。
- NodeImageIdSSMParam：使用用于某个 Kubernetes 版本的最近的 Amazon EKS 优化版 Amazon Linux AMI ID 的 Amazon EC2 Systems Manager 参数进行预填充。要使用 Amazon EKS 支持的其他 Kubernetes 次要版本，请将 *1.x* 替换为不同的支持版本 (p. 59)。我们建议您指定与您的集群相同的 Kubernetes 版本。

  要使用 Amazon EKS 优化版加速型 AMI，请将 *amazon-linux-2* 替换为 **amazon-linux-2-gpu**。要使用 Amazon EKS 优化版 Arm AMI，请将 *amazon-linux-2* 替换为 **amazon-linux-2-arm64**。

  Note

  Amazon EKS 节点 AMI 基于 Amazon Linux 2。您可以在 Amazon Linux 安全中心跟踪 Amazon Linux 2 的安全和隐私事件，或订阅关联的 RSS 源。安全和隐私事件包括问题的概述、受影响的程序包以及如何更新实例以解决问题。
- NodeImageId：（可选）如果您使用自定义 AMI（而不是 Amazon EKS 优化版 AMI），则输入 Amazon Web Services 区域的节点 AMI ID。如果您在此处指定值，它会覆盖 NodeImageIdSSMParam 字段中的任意值。
- NodeVolumeSize：指定您的节点的根卷大小（以 GiB 为单位）。
- KeyName：输入 Amazon EC2 SSH 密钥对的名称，您可使用该密钥对来在节点启动后使用 SSH 连接到这些节点。如果还没有 Amazon EC2 密钥对，可以在 Amazon Web Services Management Console 中创建一个。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 Amazon EC2 密钥对。

  Note

  如果此处不提供密钥对，Amazon CloudFormation 堆栈创建将失败。
- BootstrapArguments：指定要传递给节点引导脚本的任何可选参数，如额外的 `kubelet` 实际参数。有关更多信息，请查看 GitHub 上的引导脚本使用信息。

  部署符合以下条件的节点组：

  - 可以将明显高出默认配置数量的 IP 地址分配给 pods，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量 (p. 270)。
  - 可以将 `IPv4` 地址分配给来自与实例不同的 CIDR 块的 pods，请参阅 教程：自定义联网 (p. 258)。
  - 可以将 `IPv6` 地址分配给容器组（pod）和服务，请参阅 the section called “IPv6” (p. 248)。
  - 使用 `containerd` 运行时，您必须使用 `config` 文件部署节点组。有关更多信息，请参阅启用 `containerd` 运行时间引导标记 (p. 157)。
  - 没有出站互联网访问权限，请参阅 私有集群要求 (p. 56)。
- DisableIMDSv1：预设情况下，每个节点支持实例元数据服务版本 1 (IMDSv1) 和 IMDSv2。您可以禁用 IMDSv1。要防止节点组中的未来节点和 pods 使用 MDSv1，请将 DisableIMDSv1 设置为 true（真）。有关 IMDS 的更多信息，请参阅配置实例元数据服务。有关限制在节点上访问的更多信息，请参阅限制对分配给工作节点的实例配置文件的访问.
- VpcId：输入您创建的 VPC (p. 228) 的 ID。
- Subnets（子网）：选择您为 VPC 创建的子网。如果您使用 为 Amazon EKS 集群创建 VPC (p. 228) 中描述的步骤创建了 VPC，则在 VPC 中仅指定私有子网以供您的节点启动到其中。您可以通过从集群的 Networking（联网）选项卡中打开每个子网链接来查看哪些子网是私有的。

  Important

  - 如果其中的任何子网是公有子网，则其必须启用自动公有 IP 地址分配设置。如果没有为该公有子网启用该设置，则您部署到该公有子网的任何节点都不会分配到公有 IP 地址，也无法与集群或其他 Amazon 服务进行通信。如果子网是在 2020 年 3 月 26 日

之前使用任一 Amazon EKS Amazon CloudFormation VPC 模板 (p. 228)部署的，或者是使用 eksctl 部署的，则会为这些公有子网禁用自动公有 IP 地址分配。有关如何为子网启用公有 IP 地址分配的信息，请参阅修改子网的公有 IPv4 寻址属性。如果节点部署到私有子网，则可以通过 NAT 网关与集群和其他 Amazon 服务进行通信。

- 如果子网没有 Internet 访问权限，请确保您了解 私有集群要求 (p. 56) 中的注意事项和额外步骤。
- 如果要在 1.18 或更早版本的集群中部署节点，请确保您选择的子网标记了集群名称。将 *my-cluster* 替换为您的集群名称。然后，运行以下命令以查看当前标记为您的集群名称的子网列表。

```
aws ec2 describe-subnets --filters Name=tag:kubernetes.io/cluster/my-cluster,Values=shared | grep SubnetId
```

如果要选择的子网未在上一个命令的输出中返回，则手动将标签添加到子网。有关更多信息，请参阅the section called “子网要求和注意事项” (p. 227)。

- 如果您选择 Amazon Outposts、Wavelength 或本地区域子网，则这些子网不得在您创建集群时就已传入。

7. 在 Configure stack options（配置堆栈选项）页面上选择所需选项，然后选择 Next（下一步）。
8. 选择 I acknowledge that Amazon CloudFormation might create IAM resources.（我了解可能会创建 IAM 资源。）左侧的复选框，然后选择 Create stack（创建堆栈）。
9. 完成创建堆栈后，在控制台中选中它，然后选择 Outputs（输出）。
10. 记录已创建的节点组的 NodeInstanceRole。您在配置 Amazon EKS 节点时需要此值。

步骤 2：使节点能够加入集群

Note

如果您在没有出站 Internet 访问的情况下在私有 VPC 内启动了节点，请确保使节点能够从 VPC 中加入您的集群。

1. 下载、编辑和应用 Amazon IAM 身份验证器配置映射。
   a. 使用下面的命令下载配置映射。

```
curl -o aws-auth-cm.yaml https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/cloudformation/2020-10-29/aws-auth-cm.yaml
```

   b. 在 aws-auth-cm.yaml 文件中，将 rolearn 设置为您在上一个步骤中记录的值。您可以使用文本编辑器或者通过替换 *my-node-instance-role* 和运行以下命令来执行此操作：

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
```

   c. 应用配置。此命令可能需要几分钟才能完成。

```
kubectl apply -f aws-auth-cm.yaml
```

Note

如果您收到任何授权或资源类型错误，请参阅故障排除部分中的未经授权或访问被拒绝 (kubectl) (p. 478)。

如果节点无法加入集群，请参阅《故障排除指南》中的 节点未能加入集群 (p. 478)。

2. 查看节点的状态并等待它们达到 Ready 状态。

```
kubectl get nodes --watch
```

输入 Ctrl+C 以返回到 Shell 提示符。

3. （仅限 GPU 节点）如果选择 GPU 实例类型和 Amazon EKS 优化版加速型 AMI，则必须使用以下命令在集群上将适用于 Kubernetes 的 NVIDIA 设备插件用作 DaemonSet。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/nvidia-device-plugin.yml
```

4. （可选）部署示例应用程序 (p. 315)以测试集群和 Linux 节点。
5. （可选）如果 AmazonEKS_CNI_Policy 托管 IAM policy（如果您拥有 IPv4 集群）或 *AmazonEKS_CNI_IPv6_Policy*（如果您拥有 IPv6 集群时自行创建 (p. 246)）附加到 the section called “节点 IAM 角色” (p. 427)，我们建议将其分配到关联到 Kubernetes aws-node 服务账户的 IAM 角色。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。
6. 如果满足以下条件，我们建议阻止容器组（pod）访问 IMDS：

   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便容器组（pod）只具有他们所需的最低权限。
   - 集群中没有任何容器组（pod）需要出于其他原因（例如检索当前 Amazon Web Services 区域）访问 Amazon EC2 实例元数据服务（IMDS）。

   有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

启动自行管理的 Bottlerocket 节点

Note

托管节点组可能会为您的使用案例带来一些优势。有关更多信息，请参阅托管节点组 (p. 87)。

本主题介绍了如何启动向 Amazon EKS 集群注册的 Bottlerocket 节点的自动扩缩组。Bottlerocket 是 Amazon 中的一个基于 Linux 的开源操作系统，用于在虚拟机或裸机主机上运行容器。在这些节点加入集群后，您可以向其部署 Kubernetes 应用程序。有关 Bottlerocket 的更多信息，请参阅 GitHub 上的将 Bottlerocket AMI 用于 Amazon EKS 和 eksctl 文档中的自定义 AMI 支持。

有关就地升级的信息，请参阅 GitHub 上的 Bottlerocket Update Operator。

Important

Amazon EKS 节点是标准的 Amazon EC2 实例，您需要基于常规的 Amazon EC2 实例价格为其付费。有关更多信息，请参阅 Amazon EC2定价。

Important

- 您可以部署到采用 x86 或 Arm 处理器的 Amazon EC2 实例。但是，您无法部署到具有 Inferentia 芯片的实例。
- 没有 Amazon CloudFormation 模板用来部署节点。
- Bottlerocket 图像不附带 SSH 服务器或 shell。您可以使用带外访问方法来允许 SSH 启用管理员容器，并执行一些带有用户数据的引导启动配置步骤。有关更多信息，请参阅 GitHub 上的 bottlerocket README.md：
  - Exploration (探索)
  - 管理员容器
  - Kubernetes 设置

要使用 **eksctl** 启动 Bottlerocket 节点

此过程需要 eksctl 版本 0.110.0 或更高版本。可以使用以下命令来查看您的版本：

```
eksctl version
```

有关如何安装或升级 eksctl 的说明，请参阅 安装或更新 eksctl (p. 10)。

> **Note**
>
> 此过程仅适用于使用 eksctl 创建的集群。

1. 将以下内容复制到您的设备。将 *my-cluster* 替换为您的集群名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。将 *ng-bottlerocket* 替换为您的节点组名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。要在 Arm 实例上部署，请将 *m5.large* 替换为 Arm 实例类型。将 *my-ec2-keypair-name* 替换为 Amazon EC2 SSH 密钥对的名称，您可使用该密钥对来在节点启动后使用 SSH 连接到这些节点。如果还没有 Amazon EC2 密钥对，可以在 Amazon Web Services Management Console 中创建一个。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 Amazon EC2 密钥对。将所有剩余 *example values* 替换为您自己的值。完成替换后，运行修改后的命令以创建 bottlerocket.yaml 文件。

   如果指定了 Arm Amazon EC2 实例类型，请在部署前查看 Amazon EKS 优化版 Arm Amazon Linux AMI (p. 159) 中的注意事项。有关如何使用自定义 AMI 进行部署的说明，请参阅 GitHub 上的构建 Bottlerocket 以及 eksctl 文档中的自定义 AMI 支持。要部署托管节点组，请使用启动模板部署自定义 AMI。有关更多信息，请参阅启动模板支持 (p. 100)。

   > **Important**
   >
   > 要将节点组部署到 Amazon Outposts、Amazon Wavelength 或 Amazon 本地区域子网，创建集群时不要传入 Amazon Outposts、Amazon Wavelength 或 Amazon 本地区域子网。您必须在以下示例中指定子网。有关更多信息，请参阅 eksctl 文档中的从配置文件创建节点组和 Config 文件架构。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.23'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-bottlerocket
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Bottlerocket
    ami: auto-ssm
    iam:
       attachPolicyARNs:
          - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
```

```
EOF
```

2. 使用以下命令部署您的节点。

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

输出示例如下。

创建节点时会输出几行。输出的最后几行类似于以下示例行。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. （可选）使用 Amazon EBS CSI 插件在 Bottlerocket 节点上创建 Kubernetes 持久卷。默认 Amazon EBS 驱动程序依赖于一些 Bottlerocket 中不包含的文件系统工具。有关使用驱动程序创建存储类的更多信息，请参阅 Amazon EBS CSI 驱动程序 (p. 195)。
4. （可选）kube-proxy 默认将 nf_conntrack_max 内核参数设置为默认值，该值可能与 Bottlerocket 最初在启动时设置的值不同。要保留 Bottlerocket 的原定设置，请使用以下命令编辑 kube-proxy 配置。

```
kubectl edit -n kube-system daemonset kube-proxy
```

将 --conntrack-max-per-core 和 --conntrack-min 添加到以下示例中的 kube-proxy 参数。设置为 0 意味着没有变化。

```
      containers:
      - command:
        - kube-proxy
        - --v=2
        - --config=/var/lib/kube-proxy-config/config
        - --conntrack-max-per-core=0
        - --conntrack-min=0
```

5. （可选）部署示例应用程序 (p. 315)来测试您的 Bottlerocket 节点。
6. 如果满足以下条件，我们建议阻止容器组（pod）访问 IMDS：

   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便容器组（pod）只具有他们所需的最低权限。
   - 集群中没有任何容器组（pod）需要出于其他原因（例如检索当前 Amazon Web Services 区域）访问 Amazon EC2 实例元数据服务（IMDS）。

   有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

启动自行管理的 Windows 节点

本主题介绍了如何启动向 Amazon EKS 集群注册的 Windows 节点的自动扩缩组。在这些节点加入集群后，您可以向其部署 Kubernetes 应用程序。

> Important
>
> Amazon EKS 节点是标准的 Amazon EC2 实例，您需要基于常规的 Amazon EC2 实例价格为其付费。有关更多信息，请参阅 Amazon EC2定价。

为集群启用 Windows 支持。我们建议您在启动 Windows 节点组之前查看重要的注意事项。有关更多信息，请参阅启用 Windows 支持 (p. 49)。

您可以通过 eksctl 或 Amazon Web Services Management Console 启动自行管理的 Windows 节点。

eksctl

要使用 **eksctl** 启动自行管理的 Windows 节点

此过程要求您已安装 eksctl，且您的 eksctl 版本至少为 0.110.0。可以使用以下命令来查看您的版本。

```
eksctl version
```

有关如何安装或升级 eksctl 的说明，请参阅 安装或更新 eksctl (p. 10)。

Note

此过程仅适用于使用 eksctl 创建的集群。

1. （可选）如果 AmazonEKS_CNI_Policy 托管 IAM policy（如果您拥有 IPv4 集群）或 *AmazonEKS_CNI_IPv6_Policy*（如果您拥有 IPv6 集群时自行创建 (p. 246)）附加到 the section called “节点 IAM 角色” (p. 427)，我们建议将其分配到关联到 Kubernetes aws-node 服务账户的 IAM 角色。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。
2. 此过程假设您已经有一个现有集群。如果您还没有要将 Windows 节点组添加到的 Amazon EKS 集群和 Amazon Linux 2 节点组，则建议您遵循 开始使用 Amazon EKS – eksctl (p. 13) 指南操作。指南提供使用 Amazon Linux 节点创建 Amazon EKS 集群的完整演练。

   使用以下命令创建您的节点组。将 *my-cluster* 替换为您的集群名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。将 *ng-windows* 替换为您的节点组名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。将剩余的 *example values* 替换为您自己的值。

   Important

   要将节点组部署到 Amazon Outposts、Amazon Wavelength 或 Amazon 本地区域子网，在创建集群时不要传递 Amazon Outposts、Wavelength 或本地区域子网。使用配置文件创建节点组，指定 Amazon Outposts、Wavelength 或本地区域子网。有关更多信息，请参阅 eksctl 文档中的从配置文件创建节点组和 Config 文件架构。

   Note

   Amazon EKS 优化版 Windows AMI 可配置为使用 containerd 作为运行时。当使用 eksctl 启动 Windows 节点时，在节点组配置中将 containerRuntime 指定为 containerd。有关更多信息，请参阅本用户指南中的 启用 containerd 运行时间引导标记 (p. 180) 或 eksctl 文档中的定义容器运行时。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

   ```
   eksctl create nodegroup \
       --region region-code \
       --cluster my-cluster \
       --name ng-windows \
       --node-type t2.large \
       --nodes 3 \
       --nodes-min 1 \
       --nodes-max 4 \
       --managed=false \
       --node-ami-family WindowsServer2019FullContainer
   ```

   Note

   - 如果节点无法加入集群，请参阅《故障排除指南》中的 节点未能加入集群 (p. 478)。
   - 要查看 eksctl 命令可用选项，请输入以下命令。

```
eksctl command -help
```

输出示例如下。创建节点时会输出几行。输出的最后几行类似于以下示例行。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. （可选）部署示例应用程序 (p. 315)以测试集群和 Windows 节点。
4. 如果满足以下条件，我们建议阻止容器组（pod）访问 IMDS：
   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便容器组（pod）只具有他们所需的最低权限。
   - 集群中没有任何容器组（pod）需要出于其他原因（例如检索当前 Amazon Web Services 区域）访问 Amazon EC2 实例元数据服务（IMDS）。

   有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

Amazon Web Services Management Console

先决条件

- 现有 Amazon EKS 集群和 Linux 节点组。如果您没有这些资源，我们建议您按照我们的开始使用 Amazon EKS (p. 4)指南之一创建它们。这些指南介绍如何使用 Linux 节点创建 Amazon EKS 集群。
- 符合 Amazon EKS 集群要求的现有 VPC 和安全组。有关更多信息，请参阅 Amazon EKS VPC 和子网要求和注意事项 (p. 226) 和 Amazon EKS 安全组要求和注意事项 (p. 231)。开始使用 Amazon EKS (p. 4) 指南创建符合要求的 VPC。或者，您可以按照 为 Amazon EKS 集群创建 VPC (p. 228) 手动创建一个。
- 现有 Amazon EKS 集群，它使用符合 Amazon EKS 集群要求的 VPC 和安全组。有关更多信息，请参阅创建 Amazon EKS 集群 (p. 21)。如果您在启用了 Amazon Outposts、Amazon Wavelength 或 Amazon Local Zones 的 Amazon Web Services 区域 中有子网，则创建集群时不得传入这些子网。

步骤 1：要使用 Amazon Web Services Management Console 启动自行管理 Windows 节点

1. 等待集群状态显示为 ACTIVE。如果在集群处于活动状态之前启动节点，则节点将无法向集群注册，您需要重新启动节点。
2. 打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
3. 选择 Create stack（创建堆栈）。
4. 对于 Specify template（指定模板），选择 Amazon S3 URL，然后复制以下 URL，将其粘贴到 Amazon S3 URL，然后选择 Next（下一步）两次。

   ```
   https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/cloudformation/2020-10-29/amazon-eks-windows-nodegroup.yaml
   ```

5. 在 Quick create stack（快速创建堆栈）页面上，相应地填写以下参数：
   - 堆栈名称：为 Amazon CloudFormation 堆栈选择堆栈名称。例如，您可以将其命名为 ***my-cluster*-nodes**。
   - ClusterName：输入您在创建 Amazon EKS 集群时使用的名称。

     Important

     此名称必须与 第 1 步：创建 Amazon EKS 集群 (p. 16) 中使用的名称完全匹配。否则，您的节点无法加入群集。

- ClusterControlPlaneSecurityGroup：从您在创建 VPC (p. 228) 时生成的 Amazon CloudFormation 输出中，选择安全组。

  以下步骤显示了检索适用组的一种方法。

  1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
  2. 选择集群的名称。
  3. 选择 Networking（联网）选项卡。
  4. 从 ClusterControlPlaneSecurityGroup 下拉列表中选择时使用 Additional security groups（其他安全组）值作为参考。
- NodeGroupName：输入节点组的名称。稍后您可以使用此名称来标识为您的节点创建的弹性伸缩节点组。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。
- NodeAutoScalingGroupMinSize：输入您的节点 Auto Scaling 组可缩减到的最小节点数。
- NodeAutoScalingGroupDesiredCapacity：输入创建堆栈时要扩展到的所需节点数目。
- NodeAutoScalingGroupMaxSize：输入您的节点 Auto Scaling 组可横向扩展到的最大节点数。
- NodeInstanceType：选择节点的实例类型。有关更多信息，请参阅选择 Amazon EC2 实例类型 (p. 145)。

  Note

  GitHub 上的 vpc_ip_resource_limit.go 中列出了 Amazon VPC CNI plugin for Kubernetes 最新版本支持的实例类型。您可能需要更新 CNI 版本以使用最新的受支持实例类型。有关更多信息，请参阅更新 Amazon VPC CNI plugin for Kubernetes 自行管理的附加组件 (p. 238)。
- NodeImageIdSSMParam：预先填充了当前推荐的 Amazon EKS 优化版 Windows Core AMI ID 的 Amazon EC2 Systems Manager 参数。要使用完整版本的 Windows，请将 *Core* 替换为 `Full`。
- NodeImageId：（可选）如果您使用自定义 AMI（而不是 Amazon EKS 优化版 AMI），则输入 Amazon Web Services 区域的节点 AMI ID。如果您为此字段指定值，它会覆盖 NodeImageIdSSMParam 字段中的任意值。
- NodeVolumeSize：指定您的节点的根卷大小（以 GiB 为单位）。
- KeyName：输入 Amazon EC2 SSH 密钥对的名称，您可使用该密钥对来在节点启动后使用 SSH 连接到这些节点。如果还没有 Amazon EC2 密钥对，可以在 Amazon Web Services Management Console 中创建一个。有关更多信息，请参阅 Windows 实例 Amazon EC2 用户指南中的 Amazon EC2 密钥对。

  Note

  如果您不在此处提供密钥对，Amazon CloudFormation 堆栈创建将失败。
- BootstrapArguments：指定要传递给节点引导脚本的所有可选参数，如使用 `-KubeletExtraArgs` 的其他 `kubelet` 实际参数。

  Note

  您可以配置 Amazon EKS 优化版 Windows AMI 使用 `containerd` 作为运行时。使用 Amazon CloudFormation 模板来创建 Windows 节点时，请在引导参数中指定 `-ContainerRuntime containerd` 以启用 `containerd` 运行时。有关更多信息，请参阅启用 `containerd` 运行时间引导标记 (p. 180)。
- DisableIMDSv1：预设情况下，每个节点支持实例元数据服务版本 1 (IMDSv1) 和 IMDSv2。您可以禁用 IMDSv1。要防止节点组中的未来节点和 pods 使用 MDSv1，请将 DisableIMDSv1 设置为 true（真）。有关 IMDS 的更多信息，请参阅配置实例元数据服务。
- VpcId：选择创建的 VPC (p. 228) 的 ID。

- NodeSecurityGroups：选择在您创建 VPC (p. 228) 时为您的 Linux 节点组创建的安全组。如果您的 Linux 节点附加了多个安全组，请指定所有安全组。例如，如果 Linux 节点组是使用创建的 `eksctl`。
- 子网：选择创建的子网。如果您使用 为 Amazon EKS 集群创建 VPC (p. 228) 中的步骤创建了 VPC，则在 VPC 中仅指定私有子网以供您的节点启动到其中。

  Important

  - 如果其中的任何子网是公有子网，则其必须启用自动公有 IP 地址分配设置。如果没有为该公有子网启用该设置，则您部署到该公有子网的任何节点都不会分配到公有 IP 地址，也无法与集群或其他 Amazon 服务进行通信。如果子网是在 2020 年 3 月 26 日之前使用任一 Amazon EKS Amazon CloudFormation VPC 模板 (p. 228)部署的，或者是使用 `eksctl` 部署的，则会为这些公有子网禁用自动公有 IP 地址分配。有关如何为子网启用公有 IP 地址分配的信息，请参阅修改子网的公有 IPv4 寻址属性。如果节点部署到私有子网，则可以通过 NAT 网关与集群和其他 Amazon 服务进行通信。
  - 如果子网没有 Internet 访问权限，请确保您了解 私有集群要求 (p. 56) 中的注意事项和额外步骤。
  - 如果要在 `1.18` 或更早版本的集群中部署节点，请确保您选择的子网标记了集群名称。将 *`my-cluster`* 替换为您的集群名称。然后，运行以下命令以查看当前标记为您的集群名称的子网列表。

    ```
    aws ec2 describe-subnets --filters Name=tag:kubernetes.io/cluster/my-cluster,Values=shared | grep SubnetId
    ```

    如果要选择的子网未在上一个命令的输出中返回，则手动将标签添加到子网。有关更多信息，请参阅the section called “子网要求和注意事项” (p. 227)。
  - 如果您选择 Amazon Outposts、Wavelength 或本地区域子网，则这些子网不得在您创建集群时就已传入。

6. 确认堆栈可创建 IAM 资源，然后选择 Create stack（创建堆栈）。
7. 完成创建堆栈后，在控制台中选中它，然后选择 Outputs（输出）。
8. 记录已创建的节点组的 NodeInstanceRole。您在配置 Amazon EKS Windows 节点时需要此值。

步骤 2：使节点能够加入集群

1. 下载、编辑和应用 Amazon IAM 身份验证器配置映射。

   a. 下载配置映射：

   ```
   curl -o aws-auth-cm-windows.yaml https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/cloudformation/2020-10-29/aws-auth-cm-windows.yaml
   ```

   b. 使用首选文本编辑器打开 文件。将 *`ARN of instance role (not instance profile) of **Linux** node`* 和 *`ARN of instance role (not instance profile) of **Windows** node`* 代码段替换为您为 Linux 和 Windows 节点记录的 NodeInstanceRole 值，然后保存该文件。

   Important

   - 请勿修改此文件中的任何其他行。
   - 不要对 Windows 和 Linux 节点使用相同的 IAM 角色。

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
   ```

```
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile) of **Linux** node
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: ARN of instance role (not instance profile) of **Windows** node
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
```

c. 应用配置。此命令可能需要几分钟才能完成。

```
kubectl apply -f aws-auth-cm-windows.yaml
```

Note

如果您收到任何授权或资源类型错误，请参阅故障排除部分中的未经授权或访问被拒绝 (kubectl) (p. 478)。

如果节点无法加入集群，请参阅《故障排除指南》中的 节点未能加入集群 (p. 478)。

2. 查看节点的状态并等待它们达到 Ready 状态。

```
kubectl get nodes --watch
```

3. （可选）部署示例应用程序 (p. 315)以测试集群和 Windows 节点。
4. （可选）如果 AmazonEKS_CNI_Policy 托管 IAM policy（如果您拥有 IPv4 集群）或 *AmazonEKS_CNI_IPv6_Policy*（如果您拥有 IPv6 集群时自行创建 (p. 246)）附加到 the section called “节点 IAM 角色” (p. 427)，我们建议将其分配到关联到 Kubernetes aws-node 服务账户的 IAM 角色。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。
5. 如果满足以下条件，我们建议阻止容器组（pod）访问 IMDS：

   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便容器组（pod）只具有他们所需的最低权限。
   - 集群中没有任何容器组（pod）需要出于其他原因（例如检索当前 Amazon Web Services 区域）访问 Amazon EC2 实例元数据服务（IMDS）。

   有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

自行管理节点的更新

当发布新的 Amazon EKS 优化版 AMI 时，考虑将您自行管理节点组中的节点替换为这一新的 AMI。同样，如果为 Amazon EKS 集群更新 Kubernetes 版本，则还应更新节点将其用于同一 Kubernetes 版本。

Important

本主题介绍适用于自行管理的节点的节点更新。如果您正在使用 托管节点组 (p. 87)，请参阅 更新托管节点组 (p. 95)。

有两种基本方法可以更新集群中自行管理的节点组以使用新 AMI：

- 迁移到新的节点组 (p. 120) – 创建一个新的节点组并将您的 pods 迁移到该组。迁移到新节点组比简单地在现有 Amazon CloudFormation 堆栈中更新 AMI ID 更好。这是因为迁移过程会将旧节点组污染为 `NoSchedule`，并在新堆栈准备好接受现有 pod 工作负载之后耗尽节点。
- 更新现有自行管理的节点组 (p. 124) – 更新现有节点组的 Amazon CloudFormation 堆栈以使用新 AMI。使用 `eksctl` 创建的节点组不支持此方法。

迁移到新的节点组

本主题介绍如何创建新的节点组，将您的现有应用程序自然地迁移到新组，然后从您的集群中删除旧的节点组。您可以使用 `eksctl` 或 Amazon Web Services Management Console 迁移到新的节点组。

使用 `eksctl` 将您的应用程序迁移到新的节点组

此过程需要 `eksctl` 版本 `0.110.0` 或更高版本。可以使用以下命令来查看您的版本：

```
eksctl version
```

有关如何安装或升级 `eksctl` 的说明，请参阅 安装或更新 eksctl (p. 10)。

> **Note**
>
> 此过程仅适用于使用 `eksctl` 创建的集群和节点组。

1. 检索您现有节点组的名称，同时将 *`my-cluster`* 替换为您的集群名称。

   ```
   eksctl get nodegroups --cluster=my-cluster
   ```

   输出示例如下。

   ```
   CLUSTER      NODEGROUP          CREATED               MIN SIZE      MAX SIZE
    DESIRED CAPACITY     INSTANCE TYPE     IMAGE ID
   default      standard-nodes   2019-05-01T22:26:58Z  1             4             3
                t3.medium         ami-05a71d034119ffc12
   ```

2. 用 `eksctl` 和下面的命令启动新节点组。将命令中的所有 *`example value`* 替换为您自己的值。版本号不能高于控制面板的 Kubernetes 版本。此外，它不能比控制面板的 Kubernetes 版本低两个以上的次要版本。我们建议您使用与控制面板相同的版本。

   如果满足以下条件，我们建议阻止容器组（pod）访问 IMDS：

   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便容器组（pod）只具有他们所需的最低权限。
   - 集群中没有任何容器组（pod）需要出于其他原因（例如检索当前 Amazon Web Services 区域）访问 Amazon EC2 实例元数据服务（IMDS）。

   有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

   要阻止 pod 对 IMDS 的访问，请将 `--disable-pod-imds` 选项添加到以下命令。

   > **Note**
   >
   > 有关更多可用标志及其说明，请参阅 https://eksctl.io/。

   ```
   eksctl create nodegroup \
     --cluster my-cluster \
   ```

```
  --version 1.23 \
  --name standard-nodes-new \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --node-ami auto
```

3. 当上一个命令完成时，使用以下命令验证您的所有节点是否已达到 `Ready` 状态：

```
kubectl get nodes
```

4. 使用以下命令删除原始节点组。在命令中，将每个 *example value* 替换为集群和节点组名称：

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes
```

使用 Amazon Web Services Management Console 和 Amazon CLI 将您的应用程序迁移到新的节点组。

1. 执行 启动自行管理的 Amazon Linux 节点 (p. 107) 中概述的步骤，启动新的节点组。
2. 完成创建堆栈后，在控制台中选中它，然后选择 Outputs（输出）。
3. 记录已创建的节点组的 NodeInstanceRole。您需要它来将新的 Amazon EKS 节点添加到集群。

   Note

   如果您已将任何其他 IAM policy 附加到旧节点组 IAM 角色，则应将这些相同的策略附加到新节点组 IAM 角色以在新组上保持该功能。这适用于为 Kubernetes Cluster Autoscaler 添加权限的情况。
4. 同时更新两个节点组的安全组，以便它们可以相互通信。有关更多信息，请参阅Amazon EKS 安全组要求和注意事项 (p. 231)。

   a. 记下两个节点组的安全组 ID。这在 Amazon CloudFormation 堆栈输出中显示为 NodeSecurityGroup 值。

      您可以使用以下 Amazon CLI 命令从堆栈名称中获取安全组 ID。在这些命令中，`oldNodes` 是您的较旧节点堆栈的 Amazon CloudFormation 堆栈名称，`newNodes` 是要迁移到的堆栈的名称。将每个 *example value* 替换为您自己的值。

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
```

   b. 向每个节点安全组添加入口规则，以便它们接受彼此的流量。

      以下 Amazon CLI 命令向每个安全组添加入站规则，以允许来自另一个安全组的所有协议上的所有流量。通过此配置，在您将工作负载迁移到新组时，每个节点组中的 pods 都可以相互通信。

```
aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 authorize-security-group-ingress --group-id $newSecGroup \
```

```
--source-group $oldSecGroup --protocol -1
```

5. 编辑 aws-auth configmap 以在 RBAC 中映射新的节点实例角色。

```
kubectl edit configmap -n kube-system aws-auth
```

为新的节点组添加新的 mapRoles 条目。

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile)
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

将 *ARN of instance role (not instance profile)* 代码段替换为您在上一步 (p. 121)中记录的 NodeInstanceRole 值，然后保存该文件。保存并关闭该文件以应用更新后的 configmap。

6. 查看节点的状态并等待新节点加入您的集群并达到 Ready 状态。

```
kubectl get nodes --watch
```

7. （可选）如果使用 Kubernetes Cluster Autoscaler，请将部署缩减到 0 个副本以避免相互冲突的扩缩操作。

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. 使用以下命令，对要使用 NoSchedule 删除的每个节点执行 Taint 操作。这样不会在要替换的节点上安排或重新安排新 pods。

```
kubectl taint nodes node_name key=value:NoSchedule
```

如果您要将节点升级到新的 Kubernetes 版本，则可以使用以下代码段标识特定 Kubernetes 版本（此示例中为 1.21）的所有节点并对其执行污点操作。版本号不能高于控制面板的 Kubernetes 版本。此外，它不能比控制面板的 Kubernetes 版本低两个以上的次要版本。我们建议您使用与控制面板相同的版本。

```
K8S_VERSION=1.21
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==\"v
$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Tainting $node"
    kubectl taint nodes $node key=value:NoSchedule
done
```

9. 确定集群的 DNS 提供商。

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

下面是输出。此集群使用 CoreDNS 解析 DNS，但您的集群可能会返回 kube-dns）：

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

10. 如果您的当前部署所运行的副本少于 2 个，请将部署扩展到 2 个副本。如果您的上一个命令输出返回了该项，请将 *coredns* 替换为 **kubedns**。

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11. 使用以下命令耗尽要从集群中删除的每个节点：

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

如果您要将节点升级到新的 Kubernetes 版本，则使用以下代码段标识并耗尽特定 Kubernetes 版本（此示例中为 *1.21*）的所有节点。

```
K8S_VERSION=1.21
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==\"v
$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-local-data
done
```

12. 在旧节点耗尽后，请撤销您之前授权的安全组入口规则。然后删除 Amazon CloudFormation 堆栈以终止实例。

    Note

    如果您已将任何其他 IAM policy 附加到旧节点组 IAM 角色（例如，为 Kubernetes Cluster Autoscaler 添加权限），请先将这些附加策略与该角色分离，然后才能删除 Amazon CloudFormation 堆栈。

    a. 撤销您之前为节点安全组创建的入站规则。在这些命令中，`oldNodes` 是您的较旧节点堆栈的 Amazon CloudFormation 堆栈名称，`newNodes` 是要迁移到的堆栈的名称。

    ```
    oldNodes="old_node_CFN_stack_name"
    newNodes="new_node_CFN_stack_name"

    oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
    --query 'StackResources[?
    ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
    --output text)
    newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
    --query 'StackResources[?
    ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
    --output text)
    aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \
    --source-group $newSecGroup --protocol -1
    aws ec2 revoke-security-group-ingress --group-id $newSecGroup \
    --source-group $oldSecGroup --protocol -1
    ```

    b. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。

    c. 选择您的旧节点堆栈。

    d. 选择 Delete（删除）。

    e. 在 Delete stack（删除堆栈）确认对话框中，请选择 Delete stack（删除堆栈）。

13. 编辑 `aws-auth` configmap 以从 RBAC 中删除旧节点实例角色。

```
kubectl edit configmap -n kube-system aws-auth
```

删除旧节点组的 mapRoles 条目。

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-15-NodeInstanceRole-U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
```

保存并关闭该文件以应用更新后的 configmap。

14. （可选）如果您使用的是 Kubernetes Cluster Autoscaler，请将部署缩减为 1 个副本。

    Note

    您还必须适当地标记新的弹性缩放组（例如，k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/*my-cluster*）并将您的 Cluster Autoscaler 部署命令更新为指向新标记的 Auto Scaling 组。有关更多信息，请参阅 Amazon 上的 Cluster Autoscaler。

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15. （可选）确认您使用的是最新版本的 Kubernetes 的 Amazon VPC CNI 插件。您可能需要更新 CNI 版本以使用最新的受支持实例类型。有关更多信息，请参阅更新 Amazon VPC CNI plugin for Kubernetes 自行管理的附加组件 (p. 238)。
16. 如果您的集群使用适用于 DNS 解析的 kube-dns（请参阅上一步 (p. 122)），请将 kube-dns 部署缩减为 1 个副本。

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

更新现有自行管理的节点组

本主题介绍如何使用新 AMI 更新现有 Amazon CloudFormation 自行管理的节点堆栈。在集群更新后，可以使用此过程将节点更新为新版本的 Kubernetes。或者，您可以更新到针对现有 Kubernetes 版本的最新 Amazon EKS 优化 AMI。

Important

本主题介绍适用于自行管理的节点的节点更新。有关使用 托管节点组 (p. 87) 的信息，请参阅更新托管节点组 (p. 95)。

最新的默认 Amazon EKS 节点 Amazon CloudFormation 模板将配置为使用新 AMI 在集群中启动实例，然后再删除旧 AMI，一次删除一个。此配置可确保您在滚动更新期间始终具有集群中 Auto Scaling 组所需的活动实例计数。

Note

使用 eksctl 创建的节点组不支持此方法。如果您使用 eksctl 创建了集群或节点组，请参阅 迁移到新的节点组 (p. 120)。

更新现有节点组

1. 确定集群的 DNS 提供商。

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

输出示例如下。此集群使用 CoreDNS 解析 DNS，但您的集群可能会返回 `kube-dns`。

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

2. 如果您的当前部署所运行的副本少于 2 个，请将部署扩展到 2 个副本。如果您的上一个命令输出返回了该项，请将 *coredns* 替换为 **kube-dns**。

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

3. （可选）如果使用 Kubernetes Cluster Autoscaler，请将部署缩减到 0 个副本以避免相互冲突的扩缩操作。

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

4. 确定当前节点组的实例类型和所需的实例计数。以后更新该组的 Amazon CloudFormation 模板时将输入这些值。
   a. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
   b. 请在左侧导航窗格中，选择 Launch Configurations（启动配置），然后记下现有节点启动配置的实例类型。
   c. 请在左侧导航窗格中，选择 Auto Scaling Groups（Auto Scaling 组），并记下现有节点 Auto Scaling 组的 Desired（所需）实例计数。
5. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
6. 选择您的节点组堆栈，然后选择 Update（更新）。
7. 选择替换当前模板，然后选择 Amazon S3 URL。
8. 对于 Amazon S3 URL，请将以下 URL 粘贴到文本区域中以确保您使用的是最新版本的节点 Amazon CloudFormation 模板。接下来，选择 Next（下一步）：

```
https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/cloudformation/2020-10-29/amazon-eks-nodegroup.yaml
```

9. 在 Specify stack details (指定堆栈详细信息) 页面上，填写以下参数，然后选择 Next (下一步)：
   - NodeAutoScalingGroupDesiredCapacity - 输入在上一步 (p. 125)记录的所需实例计数。或者输入更新堆栈时要扩展到的所需节点数目。
   - NodeAutoScalingGroupMaxSize：输入您的节点 Auto Scaling 组可横向扩展到的最大节点数。此值必须比所需容量多至少一个节点。这样更新期间可以对节点进行滚动更新而不会减少节点数。
   - NodeInstanceType - 选择在上一步 (p. 125)记录的实例类型。也可以为节点选择不同的实例类型。在选择其他实例类型之前，请查看 选择 Amazon EC2 实例类型 (p. 145)。每种 Amazon EC2 实例类型支持最大数量的弹性网络接口（网络接口），每个网络接口都支持最大数量的 IP 地址。由于为每个 Worker 节点和容器组（pod）分配了自己的 IP 地址，因此请务必选择一个实例类型，该实例类型可以支持您希望在每个 Amazon EC2 节点上运行的最大数量的 pods。有关实例类型支持的网络接口和 IP 地址数量的列表，请参阅每种实例类型的每个网络接口的 IP 地址数。例如，`m5.large` 实例类型最多支持工作线程节点和 Pod 的 30 个 IP 地址。

Note

对于适用于 Kubernetes 的 Amazon VPC CNI 插件的最新版本，所支持的实例类型在 GitHub 上的 vpc_ip_resource_limit.go 中显示。您可能需要更新 CNI 版本以使用最新的受支持实例类型。有关更多信息，请参阅更新 Amazon VPC CNI plugin for Kubernetes 自行管理的附加组件 (p. 238)。

Important

并非所有实例类型在所有 Amazon Web Services 区域中都可用。

- NodeImageIdSSMParam – 要更新到的 AMI ID 的 Amazon EC2 Systems Manager 参数。以下值为 Kubernetes 版本 `1.23` 使用最新 Amazon EKS 优化版 AMI。

```
/aws/service/eks/optimized-ami/1.23/amazon-linux-2/recommended/image_id
```

  您可以将 *1.23* 替换为支持的 Kubernetes 版本 (p. 67)。或者，它应该比控制面板上运行的 Kubernetes 版本高最多一个版本。我们建议您将节点保持在与控制层面相同的版本。如果要使用 Amazon EKS 优化版加速型 AMI，请将 *amazon-linux-2* 替换为 *amazon-linux-2-gpu*。

  Note

  使用 Amazon EC2 Systems Manager 参数让您可以在以后更新节点而无需查找和指定 AMI ID。如果您的 Amazon CloudFormation 堆栈使用此值，则任何堆栈更新将为您指定的 Kubernetes 版本始终启动最新建议的 Amazon EKS 优化版 AMI。即使不更改模板中的任何值，情况也是如此。

- NodeImageId – 要使用您的自定义 AMI，请输入要使用的 AMI 的 ID。

  Important

  此值覆盖为 NodeImageIdSSMParam 指定的任意值。如果您要使用 NodeImageIdSSMParam 值，请确保 NodeImageId 的值为空白。

- DisableIMDSv1 - 每个节点默认支持实例元数据服务版本 1 (IMDSv1) 和 IMDSv2。但可以禁用 IMDSv1。如果不希望节点组中计划的任何节点或任何 pods 使用 IMDSv1，请选择 true（真）。有关 IMDS 的更多信息，请参阅配置实例元数据服务。如果已为服务账户实施 IAM 角色，请将所需权限直接分配给需要访问 Amazon 服务的所有 pods。这样，集群中就没有任何 pods 需要出于其他原因（例如检索当前 Amazon Web Services 区域）访问 IMDS。然后，对于不使用主机网络的 pods，您还可以禁用 IMDSv2 的访问权限。有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

10. （可选）在 Options (选项) 页面上，为您的堆栈资源添加标签。选择 Next (下一步)。
11. 在 Review（审核）页面上审核您的信息，确认堆栈可创建 IAM 资源，然后选择 Update stack（更新堆栈）。

    Note

    集群中每个节点的更新需要几分钟时间。等待所有节点更新完成，然后再执行后续步骤。

12. 如果您的集群的 DNS 提供商是 `kube-dns`，请将 `kube-dns` 部署缩减为 1 个副本。

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13. （可选）如果您使用的是 Kubernetes Cluster Autoscaler，请将部署缩减为所需数量的副本。

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

14. （可选）确认您使用的是最新版本的 Kubernetes 的 Amazon VPC CNI 插件。您可能需要更新 CNI 版本以使用最新的受支持实例类型。有关更多信息，请参阅更新 Amazon VPC CNI plugin for Kubernetes 自行管理的附加组件 (p. 238)。

Amazon Fargate

本主题讨论使用 Amazon EKS 在 Amazon Fargate 上运行 Kubernetes pods。

Amazon Fargate 是一种为容器按需提供大小合适的计算容量的技术。使用 Amazon Fargate，您不必再自己预置、配置或扩展虚拟机组即可运行容器。您无需再选择服务器类型、确定扩展节点组的时间和优化集群打包。您可以控制要在 Fargate 上启动的 pods，以及它们如何利用 Fargate 配置文件 (p. 131)运行。Fargate 配置文件被定义为 Amazon EKS 集群的一部分。

Amazon EKS 使用由 Amazon 构建的控制器（使用 Kubernetes 提供的上游可扩展模型）将 Kubernetes 与 Amazon Fargate 集成。这些控制器作为 Amazon EKS 托管 Kubernetes 控制面板的一部分运行，负责将本机 Kubernetes pods 安排到 Fargate 上。除了若干转换和验证准入控制器外，Fargate 控制器还包括一个与默认 Kubernetes 调度器一起运行的新调度器。当您启动满足 Fargate 上的运行条件的 pod 时，集群中运行的 Fargate 控制器会识别、更新 pod 并将其安排到 Fargate 上。

本主题介绍 Fargate 上运行的 pods 的不同组件，还列出了将 Fargate 与 Amazon EKS 结合使用时的特别注意事项。

Amazon Fargate 注意事项

以下是使用 Amazon EKS 上的 Fargate 时要考虑的一些事项。

- Amazon EKS 上的 Amazon Fargate 在除 Amazon GovCloud（美国东部）和 Amazon GovCloud（美国西部）以外的所有 Amazon EKS 区域推出。
- 在 Fargate 上运行的每组 pod 都有自己的隔离边界。其不与其他组的 pod 共享底层内核、CPU 资源、内存资源或弹性网络接口。
- 网络负载均衡器和 Application Load Balancer (ALB) 只能与具有 IP 目标的 Fargate 一起使用。有关更多信息，请参阅 创建网络负载均衡器 (p. 329) 和 Amazon EKS 上的应用程序负载均衡 (p. 332)。
- Fargate 暴露的服务仅在目标类型 IP 模式下运行，而不是在节点 IP 模式下运行。推荐通过服务名称进行连接来检查托管节点上运行的服务和 Fargate 上运行服务的连接性。
- 在对 Pod 进行安排时，它们必须与 Fargate 配置文件匹配，才能在 Fargate 上运行。与 Fargate 配置文件不匹配的容器组可能会卡在 `Pending` 状态。如果存在匹配的 Fargate 配置文件，您可以删除已创建的待处理 pods，以将它们重新安排到 Fargate。
- 您只能将 pods 的安全组 (p. 273) 与在作为 `1.18` 或更高版本集群一部分的 Fargate 上运行的 pods 一起使用。
- Fargate 上不支持 Daemonset。如果您的应用程序需要进程守护程序，则应将该进程守护程序重新配置为在您的 pods 中作为辅助容器运行。
- Fargate 上不支持特权容器。
- 在 Fargate 上运行的容器组（pod）无法在 pod 清单中指定 `HostPort` 或 `HostNetwork`。
- 对于 Fargate pods，默认 `nofile` 和 `nproc` 软限制为 1024，硬限制为 65535。
- GPU 目前在 Fargate 上不可用。
- 在 Fargate 上运行的 Pod 仅在私有子网上受支持（对Amazon服务具有 NAT 网关访问权限，但没有到互联网网关的直接路由），因此您的集群的 VPC 必须具有可用的私有子网。对于没有出站 Internet 访问的集群，请参阅 私有集群要求 (p. 56)。
- 您可以使用 Vertical Pod Autoscaler (p. 321) 为 Fargate pods 设置正确的初始 CPU 和内存大小，然后使用 Horizontal Pod Autoscaler (p. 325) 来扩展这些 pods。如果您希望 Vertical Pod Autoscaler 自动将 pods 重新部署到具有更大 CPU 和内存组合的 Fargate，请将 Vertical Pod Autoscaler 的模式设置为 `Auto` 或 `Recreate`，以确保功能正常运行。有关更多信息，请参阅 GitHub 上的 Vertical Pod Autoscaler 文档。
- 必须为 VPC 启用 DNS 解析和 DNS 主机名。有关详细信息，请参阅查看和更新您的 VPC 的 DNS 支持。

- Amazon EKS Fargate 通过将虚拟机（VM）中的每个容器组（pod）隔离出来，从而为 Kubernetes 应用程序添加深度防御。此 VM 边界可防止在容器逃离情况下访问其他容器组使用的基于主机的资源，容器逃离是攻击容器化应用程序和获取容器外部资源访问权限的常见方法。

  使用 Amazon EKS 不会更改您在责任共担模式 (p. 392)下的责任。您应该仔细考虑集群安全和治理控制的配置。隔离应用程序的最安全方法始终是在单独的集群中运行该程序。
- Fargate 配置文件支持从 VPC 辅助 CIDR 块指定子网。您可能想要指定辅助 CIDR 块。这是因为子网中可用 IP 地址的数量是有限的。因此，可在集群中创建的 pods 数量受到限制。对 pods 使用不同的子网让您可以增加可用 IP 地址的数量。有关更多信息，请参阅向 VPC 中添加 IPv4 CIDR 块。
- 部署到 Fargate 节点的 pods 无法使用 Amazon EC2 实例元数据服务（IMDS）。如果您在 Fargate 部署了需要 IAM 凭证的 pods，请使用 服务账户的 IAM 角色 (p. 395) 将它们分配到您的 pods。如果您的 pods 需要通过 IMDS 访问其他可用信息，则必须将此信息硬编码到 pod 规范中。这包括 pod 部署到的 Amazon Web Services 区域 或可用区。
- 您不能将 Fargate pods 部署到 Amazon Outposts、Amazon Wavelength 或 Amazon 本地区域。
- Amazon EKS 必须定期修补 Amazon Fargate pods 来确保它们的安全。我们以减少影响的方式尝试更新，但是有时候，如果没有成功驱逐 pods，必须删除它们。您可以采取一些措施来尽量减少中断。有关更多信息，请参阅Fargate pod 修补 (p. 136)。
- 适用于 Amazon EKS 的 Amazon VPC CNI 插件安装在 Fargate 节点上。您不可以将 the section called “备选的兼容 CNI 插件” (p. 287) 与 Fargate 节点一起使用。

通过 Amazon EKS 开始使用 Amazon Fargate

本主题描述如何通过 Amazon EKS 集群开始在 Amazon Fargate 上运行 pods。

如果使用 CIDR 块限制对集群的公有端点的访问，建议您还启用私有端点访问。通过这种方式，Fargate pods 可以与集群通信。在未启用私有端点的情况下，您指定用于公有访问的 CIDR 块必须包含来自 VPC 的出站源。有关更多信息，请参阅Amazon EKS 集群端点访问控制 (p. 38)。

先决条件

现有集群。Amazon EKS 上的 Amazon Fargate 在除 Amazon GovCloud（美国东部）和 Amazon GovCloud（美国西部）以外的所有 Amazon EKS 区域推出。如果您还没有 Amazon EKS 集群，请参阅 开始使用 Amazon EKS (p. 4)。

确保现有节点可以与 Fargate pods 进行通信

如果您使用的是没有节点的新集群，或是只有托管节点组 (p. 87)的集群，可以跳至 创建 Fargate pod 执行角色 (p. 129)。

假设您正在使用已有关联节点的现有集群。确保这些节点上的 pods 可以与 Fargate 上运行的 pods 自由通信。Fargate 上运行的 Pods 会自动配置为为与这些节点关联的集群使用集群安全组。确保集群中的任何现有节点都可以向集群安全组发送流量以及从中接收流量。托管节点组 (p. 87) 也自动配置为使用集群安全组，这样无需修改或检查它们是否具备此兼容性。

针对已使用 `eksctl` 或者 Amazon EKS 管理 Amazon CloudFormation 模板创建的现有节点组，您可以手动将集群安全组添加到节点。或者，您也可以修改节点组的 Auto Scaling 组启动模板，以便将集群安全组附加到实例。想要了解更多有关信息，请参阅 Amazon VPC 用户指南中的更改实例的安全组主题。

您可以在 Amazon Web Services Management Console 中集群的 Networking（联网）部分下，检查您集群的安全组。或者，您可以使用下面的 Amazon CLI 命令进行这项操作。使用此命令时，请将 *my-cluster* 替换为您的集群名称。

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

创建 Fargate pod 执行角色

当您的集群在 Amazon Fargate 上创建 pods 时，在 Fargate 基础设施上运行的组件必须代表您调用 Amazon API。Amazon EKS pod 执行角色提供执行此操作的 IAM 权限。要创建 Amazon Fargate pod 执行角色，请参阅 Amazon EKS pod 执行 IAM 角色 (p. 430)。

> Note
>
> 如果您使用 `--fargate` 选项通过 `eksctl` 创建了集群，则您的集群已具有 pod 执行角色，您可以使用模式 `eksctl-`*`my-cluster`*`-FargatePodExecutionRole-`*`ABCDEFGHIJKL`* 在 IAM 控制台中找到它。同样，如果您使用 `eksctl` 创建 Fargate 配置文件，则 `eksctl` 会创建您的 pod 执行角色（如果尚未创建）。

为您的集群创建 Fargate 配置文件

您必须先定义一个 Fargate 配置文件，以指定在启动时哪些 pods 使用 Fargate，然后才能安排在集群中的 Fargate 上运行的 pods。有关更多信息，请参阅Amazon Fargate 配置文件 (p. 131)。

> Note
>
> 如果您使用 `--fargate` 选项通过 `eksctl` 创建了集群，则已经为您的集群创建了 Fargate 配置文件，而且其包含 `kube-system` 和 `default` 命名空间中所有 pods 的选择器。使用以下程序为您想要用于 Fargate 的任何其他命名空间创建 Fargate 配置文件。

您可以使用 `eksctl` 或 Amazon Web Services Management Console 创建 Fargate 配置文件。

eksctl

此过程需要 `eksctl` 版本 `0.110.0` 或更高版本。可以使用以下命令来查看您的版本：

```
eksctl version
```

有关如何安装或升级 `eksctl` 的说明，请参阅 安装或更新 eksctl (p. 10)。

利用 **`eksctl`** 创建 Fargate 配置文件

使用以下 `eksctl` 命令创建 Fargate 配置文件，并将所有 *`example value`* 替换为您自己的值。您需要指定命名空间。但是，`--labels` 选项不是必选。

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

您可以将某些通配符用于 *`my-kubernetes-namespace`* 和 *`key=value`* 标签。有关更多信息，请参阅Fargate 配置文件通配符 (p. 132)。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 为集群创建 Fargate 配置文件

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择要为其创建 Fargate 配置文件的集群。
3. 请选择 Compute（计算）选项卡。
4. 在 Fargate profiles（Fargate 配置文件）下，选择 Add Fargate profile（添加 Fargate 配置文件）。

5. 在 Configure Fargate profile（配置 Fargate 配置文件）页面上，执行以下操作：
   a. 对于 Name（名称），为 Fargate 配置文件输入名称。名称必须唯一。
   b. 对于 Pod execution role（容器组（pod）执行角色），选择要用于您的 Fargate 配置文件的 pod 执行角色。将仅显示具有 `eks-fargate-pods.amazonaws.com` 服务委托人的 IAM 角色。如果您未看到列出的任何角色，则必须创建一个角色。有关更多信息，请参阅Amazon EKS pod 执行 IAM 角色 (p. 430)。
   c. 选择 Subnets（子网）下拉列表并取消选择名称中带有 `Public` 的所有子网。Fargate 上运行的 pods 仅支持私有子网。
   d. 对于 Tags（标签），您可以自行选择是否为 Fargate 配置文件添加标签。这些标签不会传播到与配置文件关联的其他资源，如 pods。
   e. 选择 Next (下一步)。
6. 在 Configure pod selection（配置选择）页面上，请执行以下操作：
   a. 对于 Namespace（命名空间），输入与 pods 匹配的命名空间。
      - 您可以使用匹配的特定命名空间，例如 **`kube-system`** 或 **`default`**。
      - 您可以使用某些通配符（例如 **`prod-*`**）以匹配多个命名空间（例如，`prod-deployment` 和 `prod-test`）。有关更多信息，请参阅Fargate 配置文件通配符 (p. 132)。
   b. （可选）将 Kubernetes 标签添加到选择器中。特别是将它们添加到指定命名空间中的 pods 需要匹配的那个。
      - 您可以将标签 **`infrastructure: fargate`** 添加到选择器中，以便只有指定命名空间中也具有 `infrastructure: fargate` Kubernetes 标签的 pods 与选择器匹配。
      - 您可以使用某些通配符（例如 **`key?: value?`**）以匹配多个命名空间（例如，`keya: valuea` 和 `keyb: valueb`）。有关更多信息，请参阅Fargate 配置文件通配符 (p. 132)。
   c. 选择 Next (下一步)。
7. 在 Review and create（查看和创建）页面上，查看 Fargate 配置文件的信息，然后选择 Create（创建）。

更新 CoreDNS

预设情况下，CoreDNS 配置为在 Amazon EKS 集群的 Amazon EC2 基础设施上运行。如果仅在集群的 Fargate 上运行 pods，请完成以下步骤。

> **Note**
>
> 如果使用 `--fargate` 选项用 `eksctl` 创建集群，则可以跳至 后续步骤 (p. 131)。

1. 使用以下命令为 CoreDNS 创建 Fargate 配置文件。将 *`my-cluster`* 替换为您的集群名称，将 *`111122223333`* 替换为您的账户 ID，将 *`AmazonEKSFargatePodExecutionRole`* 替换为您的 pod 执行角色名称，并将 *`0000000000000001`*、*`0000000000000002`* 和 *`0000000000000003`* 替换为您的私有子网 ID。如果没有 pod 执行角色，则必须首先创建一个 (p. 129)。

   > **Important**
   >
   > 角色 ARN 不能包含路径。角色 ARN 的格式必须为 `arn:aws:iam::111122223333:role/role-name`。有关更多信息，请参阅aws-auth ConfigMap 不授予对集群的访问权限 (p. 487)。

```
aws eks create-fargate-profile \
    --fargate-profile-name coredns \
    --cluster-name my-cluster \
    --pod-execution-role-arn
 arn:aws:iam::111122223333:role/AmazonEKSFargatePodExecutionRole \
    --selectors namespace=kube-system,labels={k8s-app=kube-dns} \
```

```
    --subnets subnet-0000000000000001 subnet-0000000000000002 subnet-0000000000000003
```

2. 运行下面的命令从 CoreDNS pods 中删除 `eks.amazonaws.com/compute-type : ec2` 注释。

```
kubectl patch deployment coredns \
    -n kube-system \
    --type json \
    -p='[{"op": "remove", "path": "/spec/template/metadata/annotations/
eks.amazonaws.com~1compute-type"}]'
```

后续步骤

- 您可以开始使用以下工作流迁移现有应用程序来在 Fargate 上运行现有应用程序。
  1. 创建 Fargate 配置文件 (p. 133)，该配置文件与您的应用程序的 Kubernetes 命名空间和 Kubernetes 标签相匹配。
  2. 删除并重新创建所有现有的 pods，以便可以在 Fargate 上安排它们。例如，以下命令触发 `coredns` 部署的推广。您可以修改命名空间和部署类型以更新特定 pods。

     ```
     kubectl rollout restart -n kube-system deployment coredns
     ```
- 部署 Amazon EKS 上的应用程序负载均衡 (p. 332) 以允许在 Fargate 上运行的 pods 的入口对象。
- 您可以使用 Vertical Pod Autoscaler (p. 321) 为 Fargate pods 设置正确的初始 CPU 和内存大小，然后使用 Horizontal Pod Autoscaler (p. 325) 来扩展这些 pods。如果您希望 Vertical Pod Autoscaler 自动将 pods 重新部署到具有更高 CPU 和内存组合的 Fargate，请将 Vertical Pod Autoscaler 的模式设置为 `Auto` 或 `Recreate`。这是为了确保正确的功能。有关更多信息，请参阅 GitHub 上的 Vertical Pod Autoscaler 文档。
- 您可以按照这些说明设置 Amazon Distro for OpenTelemetry (ADOT) 收集器，用于监控应用程序。

Amazon Fargate 配置文件

您必须先定义至少一个 Fargate 配置文件，以指定在启动时哪些 pods 使用 Fargate，然后在集群中的 Fargate 上安排 pods。

作为管理员，您可以使用 Fargate 配置文件声明哪些 pods 在 Fargate 上运行。您可以通过配置文件的选择器执行此操作。您最多可以为每个配置文件添加五个选择器。每个选择器都包含一个命名空间和可选标签。您必须为每个选择器定义一个命名空间。标注字段由多个可选键值对组成。与选择器匹配的容器组（pod）被安排在 Fargate 上。容器组（pod）使用选择器中指定的命名空间和标签进行匹配。如果定义了没有标签的命名空间选择器，Amazon EKS 会尝试使用配置文件将在该命名空间中运行的所有 pods 安排到 Fargate。如果待安排的 pod 与 Fargate 配置文件中的任意一个选择器匹配，则该 pod 将被安排到 Fargate 上。

如果 pod 匹配多个 Fargate 配置文件，您可以指定 pod 使用的配置文件，方法为将以下 Kubernetes 标签添加到 pod 规范：`eks.amazonaws.com/fargate-profile: my-fargate-profile`。pod 必须匹配该配置文件中的选择器才能被安排到 Fargate 上。Kubernetes 亲和力/反亲和力规则不适用，Amazon EKS Fargate pods 不需要这些规则。

创建 Fargate 配置文件时，必须指定 pod 执行角色。此执行角色适用于使用配置文件在 Fargate 基础设施上运行的 Amazon EKS 组件。此角色将被添加到集群的 Kubernetes 基于角色的访问控制（RBAC）以进行授权。这样，Fargate 基础设施上运行的 `kubelet` 可以注册到您的 Amazon EKS 集群，并在您的集群中作为节点显示。pod 执行角色还提供对 Fargate 基础设施的 IAM 权限，以允许对 Amazon ECR 映像存储库进行读取访问。有关更多信息，请参阅Amazon EKS pod 执行 IAM 角色 (p. 430)。

您无法更改 Fargate 配置文件。但是，您可以创建新的更新配置文件来替换现有配置文件，然后删除原始配置文件。

Note

删除配置文件后，使用 Fargate 配置文件运行的任何 pods 都会停止并进入待处理状态。

如果集群中有任何 Fargate 配置文件处于 `DELETING` 状态，您必须等待删除该 Fargate 配置文件后，才能在该集群中创建其他配置文件。

Amazon EKS 和 Fargate 在 Fargate 配置文件定义的每个子网中分布 pods。但是，最终可能会出现不均匀的分布。如果您必须获得均匀的分布，请使用两个 Fargate 配置文件。在您希望部署两个副本并且不希望造成任何停机的方案中，均匀的分布非常重要。我们建议每个配置文件只有一个子网。

Fargate 配置文件组件

Fargate 配置文件中包含以下组件。

- 容器组（pod）执行角色：当您的集群在 Amazon Fargate 上创建 pods 时，在 Fargate 基础设施上运行的 `kubelet` 必须代表您调用 Amazon API。例如，它需要调用才能从 Amazon ECR 提取容器镜像。Amazon EKS pod 执行角色提供执行此操作的 IAM 权限。

  创建 Fargate 配置文件时，必须指定要用于 pods 的 pod 执行角色。此角色将被添加到集群的 Kubernetes 基于角色的访问控制（RBAC）以进行授权。这允许在 Fargate 基础设施上运行的 `kubelet` 注册到您的 Amazon EKS 集群，以便它可以作为节点显示在您的集群中。有关更多信息，请参阅Amazon EKS pod 执行 IAM 角色 (p. 430)。
- 子网 – 将 pods 启动到其中的使用此配置文件的子网的 ID。目前，在 Fargate 上运行的 pods 没有分配公有 IP 地址。因此，此参数仅接受私有子网（没有到互联网网关的直接路由）。
- 选择器 – 要匹配 pods 以使用此 Fargate 配置文件的选择器。您最多可以在 Fargate 配置文件中指定五个选择器。选择器具有以下组件：
  - 命名空间 – 您必须为选择器指定命名空间。选择器仅匹配在此命名空间中创建的 pods。但是，您可以创建多个选择器来定位多个命名空间。
  - 标签 – 您可以选择指定 Kubernetes 标签来匹配选择器。选择器只匹配具有在选择器中指定的所有标签的 pods。

Fargate 配置文件通配符

除了 Kubernetes 允许的字符外，您可以在命名空间、标签键和标签值的选择器标准中使用 **`*`** 和 **`?`**：

- **`*`** 表示无、一个或多个字符。例如，**`prod*`** 可以表示 `prod` 和 `prod-metrics`。
- **`?`** 表示单个字符（例如，**`value?`** 可以表示 `valuea`）。但是，它不能表示 `value` 和 `value-a`，因为 `?` 只能准确地表示一个字符。

这些通配符可以在任何位置组合使用（例如，**`prod*`**、**`*dev`** 以及 **`frontend*?`**）。不支持其他通配符和模式匹配形式，例如正则表达式。

如果容器组（pod）规范中有多个配置文件与命名空间和标签匹配，Fargate 会根据配置文件名称的字母数字排序来选取配置文件。例如，如果配置文件 A（名称为 `beta-workload`）和配置文件 B（名称为 `prod-workload`）都有匹配的选择器可供要启动的容器组（pod）使用，Fargate 会为容器组（pod）选取配置文件 A（`beta-workload`）。容器组（pod）具有标签，显示容器组（pod）上有配置文件 A（例如 `eks.amazonaws.com/fargate-profile=beta-workload`）。

如果要将现有 Fargate 容器组（pod）迁移到使用通配符的新配置文件中，有两种方法可执行此操作：

- 使用匹配的选择器创建一个新的配置文件，然后删除旧的配置文件。标有旧配置文件的容器组（pod）将被重新安排到新的匹配配置文件中。
- 若要迁移工作负载，但不确定每个 Fargate 容器组（pod）上有哪些 Fargate 标签，则可以使用以下方法。创建一个新的配置文件，首先在同一个集群上的配置文件中按字母数字排序配置文件名称。然后，回收需要迁移到新配置文件的 Fargate 容器组（pod）。

创建 Fargate 配置文件

本主题介绍如何创建 Fargate 配置文件。Amazon EKS 上的 Amazon Fargate 在除 Amazon GovCloud（美国东部）和 Amazon GovCloud（美国西部）以外的所有 Amazon EKS 区域推出。您还必须已经创建了要用于 Fargate 配置文件的 pod 执行角色。有关更多信息，请参阅Amazon EKS pod 执行 IAM 角色 (p. 430)。在 Fargate 上运行的容器组（pod）仅在私有子网上受支持（对 Amazon Web Services 具有 NAT 网关访问权限，但没有到互联网网关的直接路由）。因此，您的集群的 VPC 必须有可用的私有子网。您可以使用 `eksctl` 或 Amazon Web Services Management Console 创建配置文件。选择包含要用来创建 Fargate 配置文件的工具名称的选项卡。

此过程需要 `eksctl` 版本 `0.110.0` 或更高版本。可以使用以下命令来查看您的版本：

```
eksctl version
```

有关如何安装或升级 `eksctl` 的说明，请参阅 安装或更新 `eksctl` (p. 10)。

eksctl

利用 **`eksctl`** 创建 Fargate 配置文件

使用以下 `eksctl` 命令创建 Fargate 配置文件，并将所有 *`example value`* 替换为您自己的值。您需要指定命名空间。但是，`--labels` 选项不是必选。

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

您可以将某些通配符用于 *`my-kubernetes-namespace`* 和 *`key=value`* 标签。有关更多信息，请参阅Fargate 配置文件通配符 (p. 132)。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 为集群创建 Fargate 配置文件

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择要为其创建 Fargate 配置文件的集群。
3. 请选择 Compute（计算）选项卡。
4. 在 Fargate profiles（Fargate 配置文件）下，选择 Add Fargate profile（添加 Fargate 配置文件）。
5. 在 Configure Fargate profile（配置 Fargate 配置文件）页面上，执行以下操作：
   a. 对于 Name（名称），为 Fargate 配置文件输入唯一名称，例如 ***`my-profile`***。
   b. 对于 Pod execution role（容器组（pod）执行角色），选择要用于您的 Fargate 配置文件的 pod 执行角色。将仅显示具有 `eks-fargate-pods.amazonaws.com` 服务委托人的 IAM 角色。如果您未看到列出的任何角色，则必须创建一个角色。有关更多信息，请参阅Amazon EKS pod 执行 IAM 角色 (p. 430)。
   c. 选择 Subnets（子网）下拉列表并取消选择名称中带有 `Public` 的所有子网。Fargate 上运行的 pods 仅支持私有子网。
   d. 对于 Tags（标签），您可以自行选择是否为 Fargate 配置文件添加标签。这些标签不会传播到与配置文件关联的其他资源，例如 pods。
   e. 选择 Next (下一步)。
6. 在 Configure pod selection（配置选择）页面上，请执行以下操作：
   a. 对于 Namespace（命名空间），输入与 pods 匹配的命名空间。

- 您可以使用匹配的特定命名空间，例如 **kube-system** 或 **default**。
- 您可以使用某些通配符（例如 **prod-***）来匹配多个命名空间（例如，prod-deployment 和 prod-test）。有关更多信息，请参阅Fargate 配置文件通配符 (p. 132)。

b. （可选）将 Kubernetes 标签添加到选择器中。特别是将它们添加到指定命名空间中的 pods 需要匹配的那个。

- 您可以将标签 **infrastructure: fargate** 添加到选择器中，以便只有指定命名空间中也具有 infrastructure: fargate Kubernetes 标签的 pods 与选择器匹配。
- 您可以使用某些通配符（例如 **key?: value?**）来匹配多个命名空间（例如，keya: valuea 和 keyb: valueb）。有关更多信息，请参阅Fargate 配置文件通配符 (p. 132)。

c. 选择 Next (下一步)。

7. 在 Review and create（查看和创建）页面上，查看 Fargate 配置文件的信息，然后选择 Create（创建）。

删除 Fargate 配置文件

本主题介绍如何删除 Fargate 配置文件。

当您删除 Fargate 配置文件时，使用该配置文件安排到 Fargate 的所有 pods 都将被删除。如果这些 pods 与另一个 Fargate 配置文件匹配，则将使用该配置文件在 Fargate 上安排它们。如果它们不再匹配任何 Fargate 配置文件，则不会被安排到 Fargate，并仍可能处于待处理状态。

集群中一次只能有一个 Fargate 配置文件处于 DELETING 状态。等待 Fargate 配置文件完成删除，然后才能删除该集群中的任何其他配置文件。

您可以使用 eksctl、Amazon Web Services Management Console 或 Amazon CLI 来删除配置文件。请选择包含要用于删除您的配置文件的工具名称的选项卡。

eksctl

使用 **eksctl** 删除 Fargate 配置文件

使用以下命令从集群中删除配置文件。请将每个 *example value* 替换为您自己的值。

```
eksctl delete fargateprofile  --name my-profile --cluster my-cluster
```

Amazon Web Services Management Console

利用 Amazon Web Services Management Console 从集群中删除 Fargate 配置文件

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 在左侧导航窗格中，选择集群。在集群列表中，选择要从中删除 Fargate 配置文件的集群。
3. 请选择 Compute（计算）选项卡。
4. 选择要删除的 Fargate 配置文件，然后选择 Delete（删除）。
5. 在 Delete Fargate profile（删除 Fargate 配置文件）页面上，输入配置文件的名称，然后选择 Delete（删除）。

Amazon CLI

使用 Amazon CLI 删除 Fargate 配置文件

使用以下命令从集群中删除配置文件。请将每个 *example value* 替换为您自己的值。

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-cluster
```

Fargate pod 配置

本节介绍在 Amazon Fargate 上运行 Kubernetes pods 的一些唯一 pod 配置的详细信息。

Pod CPU 和内存

使用 Kubernetes，您可以定义请求、最低 vCPU 量以及分配到 pod 中每个容器的内存资源。容器组（pod）由 Kubernetes 安排，以确保至少每个 pod 的请求资源可用于计算资源。有关更多信息，请参阅 Kubernetes 文档中的管理容器的计算资源。

> Note
>
> 由于 Amazon EKS Fargate 对每个节点只运行一个 pod，因此在资源较少的情况下不会出现驱逐 pods 的情况。所有 Amazon EKS Fargate pods 都以担保优先级运行，因此请求的 CPU 和内存必须等于所有容器的限制。有关更多信息，请参阅 Kubernetes 文档中的为容器组（pod）配置服务质量。

在 Fargate 上安排 pods 后，pod 规格中的 vCPU 和内存预留将确定为 pod 预置的 CPU 和内存量。

- 超出所有 Init 容器的最大请求用于确定 Init 请求 vCPU 和内存要求。
- 将所有长时间运行的容器的请求相加来确定长时间运行的请求的 vCPU 和内存要求。
- 然后为要用于 pod 的 vCPU 和内存请求选择先前两个值中较大的值。
- Fargate 会为所需 Kubernetes 组件（`kubelet`、`kube-proxy` 和 `containerd`）的每个 pod 的内存预留增加 256 MB。

Fargate 向上舍入到下列最接近 vCPU 与内存请求总和的计算配置，以确保 pods 始终拥有运行所需的资源。

如果未指定 vCPU 和内存组合，则使用最小的可用组合（.25 vCPU 和 0.5 GB 内存）。

下表显示了 Fargate 上运行的 pods 可以使用的 vCPU 和内存组合。

vCPU 值	内存值
.25 vCPU	0.5GB、1GB、2GB
.5 vCPU	1GB、2GB、3GB、4GB
1 vCPU	2GB、3GB、4GB、5GB、6GB、7GB、8GB
2 vCPU	4GB 到 16GB 之间（以 1GB 为增量）
4 vCPU	8GB 到 30GB 之间（以 1GB 为增量）

为 Kubernetes 组件保留的额外内存可能会导致 Fargate 任务的 vCPU 数量超过请求预置的 vCPU 数量。例如，由于没有具有 1 个 vCPU 和 9 GB 内存的任务可用，因此如果请求内容为 1 个 vCPU 和 8 GB 内存，则会为其内存请求增加 256 MB，并会为 Fargate 任务预置 2 个 vCPU 和 9 GB 内存。

Fargate 上运行的 pod 大小与 Kubernetes 使用 `kubectl get nodes` 报告的节点大小之间没有相关性。报告的节点大小通常大于 pod 的容量。您可以使用以下命令验证 pod 容量。将 *pod-name* 替换为 pod 的名称。

```
kubectl describe pod pod-name
```

输出示例如下。

```
...
annotations:
    CapacityProvisioned: 0.25vCPU 0.5GB
...
```

`CapacityProvisioned` 注释表示强制执行的 pod 容量，它决定了在 Fargate 上运行的 pod 的成本。有关这些计算配置的定价信息，请参阅 Amazon Fargate 定价。

Fargate 存储

预置后，Fargate 上运行的每个 pod 将接收 20 GB 的容器镜像层存储。Pod 存储是短暂存储。pod 停止后，该存储将被删除。对于 2020 年 5 月 28 日或之后在 Fargate 上启动的新 pods，默认情况下已启用短暂存储卷加密。使用 Amazon Fargate 托管密钥通过 AES-256 加密算法对短暂 pod 存储进行加密。

> **Note**
>
> Fargate 上运行的 Amazon EKS pods 的可用存储空间不到 20 GB，因为 `kubelet` 以及 pod 内部加载的其他 Kubernetes 模块占用了一些空间。

Fargate pod 修补

Amazon EKS 必须定期修补 Amazon Fargate pods 来确保它们的安全。尝试更新的方式对您的服务造成的影响最小。但是，如果 pods 没有成功被驱逐，有时必须删除它们。以下是您可以采取的最大限度减少潜在中断的操作：

- 设置适当的 pod 中断预算 (PDB)，从而控制同时关闭的 pods 数量。
- 创建事件规则来在删除 pods 之前对失败的驱逐做出反应。

Amazon EKS 与 Kubernetes 社群密切合作，以便尽快提供错误修复和安全补丁。Kubernetes 的新补丁版本作为 Fargate 平台版本的一部分提供。所有 Fargate pods 都从最新的 Kubernetes 修补程序版本开始，可从集群的 Kubernetes 版本的 Amazon EKS 中获取。如果您的 pod 具有较旧的修补程序版本，Amazon EKS 可能会重新启动它以将其更新到最新版本。这可以确保您的 pods 配备了最新的安全更新。这样一来，如果存在关键常见漏洞和风险（CVE）问题，您可以随时了解最新信息以降低安全风险。

要限制 pods 修补时同时停机的 pods 数量，您可以设置 pod 中断预算 (PDB)。您可以使用 PDB 根据每个应用程序的要求定义最低可用性，同时仍允许进行更新。有关更多信息，请参阅 Kubernetes 文档中的为应用程序指定中断预算。

Amazon EKS 使用驱逐 API 在遵守您为应用程序设置的 PDB 的同时安全地耗尽 pod。容器组（pod）被可用区驱逐出来，以最大限度地减少影响。如果驱逐成功，新 pod 将获得最新的补丁，无需进一步操作。

当 pod 的驱逐失败时，Amazon EKS 会向您的账户发送一个事件，其中包含驱逐失败的 pods 的详细信息。您可以在计划的终止时间之前对消息采取行动。具体时间根据补丁的紧迫性而有所不同。此时，Amazon EKS 尝试再次驱逐 pods。但是，如果驱逐失败，这次不会发送新的事件。如果驱逐再次失败，您现有的 pods 将定期删除，以便 pods 可以拥有最新的补丁。

以下是在 pod 驱逐失败时收到的示例事件。它包含有关集群、pod 名称、pod 命名空间、Fargate 配置文件和计划终止时间的详细信息。

```
{
    "version": "0",
    "id": "12345678-90ab-cdef-0123-4567890abcde",
    "detail-type": "EKS Fargate Pod Scheduled Termination",
    "source": "aws.eks",
    "account": "111122223333",
    "time": "2021-06-27T12:52:44Z",
    "region": "region-code",
```

```
    "resources": [
        "default/my-database-deployment"
    ],
    "detail": {
        "clusterName": "my-cluster",
        "fargateProfileName": "my-fargate-profile",
        "podName": "my-pod-name",
        "podNamespace": "default",
        "evictErrorMessage": "Cannot evict pod as it would violate the pod's disruption
 budget",
        "scheduledTerminationTime": "2021-06-30T12:52:44.832Z[UTC]"
    }
}
```

此外，将多个 PDB 与一个 pod 关联可能会导致驱逐失败事件。此事件将返回以下错误消息。

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction
 subresource does not support",
```

您可以基于此事件创建所需的操作。例如，您可以调整 pod 中断预算 (PDB)，从而控制驱逐 pods 的方式。更具体地说，假设您从指定可用 pods 的目标百分比的 PDB 开始。在升级期间强制终止 pods 之前，您可以将 PDB 调整为不同百分比的 pods。要接收此事件，您必须在 Amazon Web Services 账户 和集群所属的区域中创建 Amazon EventBridge 规则。该规则必须使用以下自定义模式。有关更多信息，请参阅《Amazon EventBridge 用户指南》中的创建对事件作出反应的 Amazon EventBridge 规则。

```
{
  "source": ["aws.eks"],
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

可以为事件设置合适的目标来捕获它。有关可用目标的完整列表，请参阅《Amazon EventBridge 用户指南》中的 Amazon EventBridge 目标。

Fargate 指标

您可以收集系统指标和 Amazon Fargate 的 CloudWatch 使用量指标。

应用程序指标

对于在 Amazon EKS 和 Amazon Fargate 上运行的应用程序，您可以使用 Amazon Distro for OpenTelemetry (ADOT)。ADOT 允许您收集系统指标并将其发送到 CloudWatch Container Insights 控制面板。要开始为在 Fargate 上运行的应用程序使用 ADOT，请参阅 ADOT 文档中的将 CloudWatch Container Insights 与 Amazon Distro for OpenTelemetry 结合使用。

使用情况指标

您可以使用 CloudWatch 用量指标来提供账户资源使用情况的可见性。使用这些指标在 CloudWatch 图表和控制面板上可视化当前服务用量。

Amazon Fargate 用量指标与 Amazon 服务配额对应。您可以配置警报，以在用量接近服务配额时向您发出警报。有关 Fargate 服务配额的更多信息，请参阅 Amazon EKS 服务配额 (p. 389)。

Amazon Fargate 在 `Amazon/Usage` 命名空间中发布以下指标。

指标	描述
`ResourceCount`	您账户中运行的指定资源的总数量。资源由与指标关联的维度定义。

以下维度用于优化由 Amazon Fargate 发布的用量指标。

维度	描述
`Service`	包含该资源的 Amazon 服务的名称。对于 Amazon Fargate 用量指标，此维度的值为 `Fargate`。
`Type`	正在报告的实体的类型。目前，Amazon Fargate 用量指标的唯一有效值为 `Resource`。
`Resource`	正在运行的资源的类型。 目前，Amazon Fargate 会返回有关 Fargate 按需使用情况的信息。Fargate 按需使用情况的资源值为 `OnDemand`。 Note Fargate 按需使用情况结合了使用 Fargate 的 Amazon EKS pods、使用 Fargate 启动类型的 Amazon ECS 任务和使用 `FARGATE` 容量提供程序的 Amazon ECS 任务。
`Class`	所跟踪的资源的类。目前，Amazon Fargate 不使用类维度。

创建 CloudWatch 警报以监控 Fargate 资源使用情况指标

Amazon Fargate 提供 CloudWatch 使用情况指标，这些指标与 Fargate 按需资源使用情况的Amazon服务配额相对应。在 Service Quotas 控制台中，您可以在图表上可视化您的使用情况。还可以配置警报，以在用量接近服务配额时向您发出警报。有关更多信息，请参阅Fargate 指标 (p. 137)。

使用以下步骤根据 Fargate 资源使用情况指标创建 CloudWatch 警报。

根据您的 Fargate 使用情况配额 (Amazon Web Services Management Console) 创建警报

1. 访问 https://console.aws.amazon.com/servicequotas/，打开 Service Quotas 控制台。
2. 在左侧导航窗格中，选择 Amazon services（Amazon 服务）。
3. 从 Amazon services（亚马逊云科技服务）列表中，搜索并选择 Amazon Fargate。
4. 在 Service quotas（服务配额）列表中，选择要为其创建警报的 Fargate 使用情况配额。
5. 在 Amazon CloudWatch 警报部分中，选择 Create（创建）。
6. 对于警报阈值，选择要设置为警报值的适用配额值的百分比。
7. 对于警报名称，输入警报名称，然后选择创建。

Fargate 日志记录

Fargate 上的 Amazon EKS 提供了一个基于 Fluent Bit 的内置日志路由器。这意味着您没有明确地将 Fluent Bit 容器作为 Sidecar 运行，但 Amazon 会为您运行它。您只需配置日志路由器即可。通过必须满足以下条件的专用 `ConfigMap` 进行配置：

- 名为 `aws-logging`
- 在名为 `aws-observability` 的专用命名空间中创建

一旦您创建了 `ConfigMap`，Fargate 上的 Amazon EKS 会自动检测到它并使用它来配置日志路由器。Fargate 使用的 Amazon for Fluent Bit 版本是 Fluent Bit 的一种上游合规发行版，由 Amazon 托管。有关更多信息，请参阅 GitHub 上的 Amazon for Fluent Bit。

日志路由器允许您使用广泛的Amazon服务进行日志分析和存储。您可以将日志从 Fargate 直接流式传输到 Amazon CloudWatch、Amazon OpenSearch Service。您还可以通过 Amazon Kinesis Data Firehose 将日志流式传输到 Amazon S3、Amazon Kinesis Data Streams 和合作伙伴工具等目标。

先决条件

- 现有的 Fargate 配置文件，用于指定您将 Fargate 容器组（pod）部署到的现有 Kubernetes 命名空间。有关更多信息，请参阅为您的集群创建 Fargate 配置文件 (p. 129)。
- 现有的 Fargate Pod 执行角色。有关更多信息，请参阅创建 Fargate pod 执行角色 (p. 129)。

日志路由器配置

配置日志路由器

在以下步骤中，将每个 *example value* 替换为您自己的值。

1. 创建名为 `aws-observability` 的专用 Kubernetes 命名空间。

   a. 将以下内容保存到计算机上名为 *aws-observability-namespace*`.yaml` 的文件中。`name` 的值必须为 `aws-observability`，并且 `aws-observability: enabled` 标注是必需的。

   ```
   kind: Namespace
   apiVersion: v1
   metadata:
     name: aws-observability
     labels:
       aws-observability: enabled
   ```

   b. 创建命名空间。

   ```
   kubectl apply -f aws-observability-namespace.yaml
   ```

2. 创建带有 `Fluent Conf` 数据值的 `ConfigMap`，以将容器日志发送到某个目标。Fluent Conf 为 Fluent Bit，是一种快速轻量级日志处理器配置语言，用于将容器日志路由到您选择的日志目标。有关更多信息，请参阅 Fluent Bit 文档中的配置文件。

   Important

   典型 `Fluent Conf` 中包含的主要部分为 `Service`、`Input`、`Filter` 和 `Output`。但是，Fargate 日志路由器仅接受：

   - `Filter` 和 `Output` 部分，并管理 `Service` 和 `Input` 部分本身。
   - `Parser` 部分。

   如果您提供了除 `Filter`、`Output` 和 `Parser` 之外的任何其他部分，则这些部分会被拒绝。

   在创建 `ConfigMap` 时，请考虑 Fargate 用于验证字段的以下规则：

   - `[FILTER]`、`[OUTPUT]` 和 `[PARSER]` 应该在每个相应的键下指定。例如，`[FILTER]` 必须在 `filters.conf` 下。在 `filters.conf` 下可以有一个或多个 `[FILTER]`。`[OUTPUT]` 和 `[PARSER]` 部分也应在其相应的键下。通过指定多个 `[OUTPUT]` 部分，您可以同时将日志路由到不同的目标。
   - Fargate 会验证每个部分所需的键。`Name` 和 `match` 是每个 `[FILTER]` 和 `[OUTPUT]` 所必需的。`Name` 和 `format` 是每个 `[PARSER]` 所必需的。键不区分大小写。
   - 在 `ConfigMap` 中不允许使用环境变量（例如 `${ENV_VAR}`）。
   - 对于每个 `filters.conf`、`output.conf` 和 `parsers.conf` 中的指令或键值对，缩进必须是相同的。键值对的缩进必须多于指令。

- Fargate 根据以下受支持的筛选条件进行验证：`grep`、`parser`、`record_modifier`、`rewrite_tag`、`throttle`、`nest`、`modify` 和 `kubernetes`。
- Fargate 根据以下受支持的输出进行验证：`es`、`firehose`、`kinesis_firehose`、`cloudwatch`、`cloudwatch_logs` 和 `kinesis`。
- `ConfigMap` 中必须至少提供一个受支持的 `Output` 插件才能启用日志记录。不需要 `Filter` 和 `Parser` 即可启用日志记录。

您还可以使用所需的配置在 Amazon EC2 上运行 Fluent Bit，以对验证过程中出现的任何问题进行故障排除。使用以下示例之一创建您的 `ConfigMap`。

Important

Amazon EKS Fargate 日志记录不支持 `ConfigMaps` 的动态配置。对 `ConfigMaps` 所做的任何更改均仅应用于新 Pod。不会将更改应用于现有 Pod。

使用所需日志目标的示例创建 `ConfigMap`。

CloudWatch

为 CloudWatch 创建 **`ConfigMap`**

使用 CloudWatch 视时，您有两个输出选项：

- 用 C 语言编写的输出插件
- 用 Golang 编写的输出插件

以下示例为您展示了如何使用 `cloudwatch_logs` 插件将日志发送到 CloudWatch。

1. 将以下内容保存到名为 *`aws-logging-cloudwatch-configmap`*`.yaml` 的文件中。将 *`region-code`* 替换为集群所在的 Amazon Web Services 区域。`[OUTPUT]` 下的参数是必需的。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
        Match   *
        region region-code
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
        auto_create_group true
        log_key log

  parsers.conf: |
    [PARSER]
        Name crio
        Format Regex
        Regex ^(?<time>[^ ]+) (?<stream>stdout|stderr) (?<logtag>P|F) (?<log>.*)$
        Time_Key    time
        Time_Format %Y-%m-%dT%H:%M:%S.%L%z

  filters.conf: |
     [FILTER]
```

```
        Name parser
        Match *
        Key_name log
        Parser crio
```

2. 将清单应用于集群。

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

3. 将 CloudWatch IAM policy 下载到您的计算机。您还可以在 GitHub 上查看策略。

```
curl -o permissions.json https://raw.githubusercontent.com/aws-samples/
amazon-eks-fluent-logging-examples/mainline/examples/fargate/cloudwatchlogs/
permissions.json
```

Amazon OpenSearch Service

要创建 **ConfigMap** for Amazon OpenSearch Service

如果您希望将日志发送到 Amazon OpenSearch Service，您可以使用 es 输出，这是以 C 语言编写的一种插件。以下示例为您展示了如何使用该插件将日志发送到 OpenSearch。

1. 将以下内容保存到名为 *aws-logging-opensearch-configmap*.yaml 的文件中。请将每个 *example value* 替换为您自己的值。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
      Name  es
      Match *
      Host  search-example-gjxdcilagiprbglqn42jsty66y.region-
code.es.amazonaws.com
      Port  443
      Index example
      Type  example_type
      AWS_Auth On
      AWS_Region region-code
      tls   On
```

2. 将清单应用于集群。

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

3. 将 OpenSearch IAM policy 下载到您的计算机。您还可以在 GitHub 上查看策略。

```
curl -o permissions.json https://raw.githubusercontent.com/aws-samples/amazon-
eks-fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/
permissions.json
```

确保 OpenSearch 控制面板的访问控制配置正确。OpenSearch 控制面板中的 `all_access role` 需要映射 Fargate Pod 执行角色和 IAM 角色。必须为 `security_manager` 角色执行同样的映射。您可以添加以前的映射，方法为：选择 `Menu`，然后依次选择 `Security` 和 `Roles`，再选择相应的角色。有关更多信息，请参阅如何对 CloudWatch Logs 进行故障排除，以便将其流式传输到我的 Amazon ES 域？

Kinesis Data Firehose

为 Kinesis Data Firehose 创建 **ConfigMap**

将日志发送到 Kinesis Data Firehose 时，您有两个输出选项：

- `kinesis_firehose` – 用 C 语言编写的输出插件。
- `firehose` – 用 Golang 编写的输出插件。

以下示例为您展示了如何使用 `kinesis_firehose` 插件将日志发送到 Kinesis Data Firehose。

1. 将以下内容保存到名为 *aws-logging-firehose-configmap*.yaml 的文件中。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
     Name  kinesis_firehose
     Match *
     region region-code
     delivery_stream my-stream-firehose
```

2. 将清单应用于集群。

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. 将 Kinesis Data Firehose IAM policy 下载到您的计算机。您还可以在 GitHub 上查看策略。

```
curl -o permissions.json https://raw.githubusercontent.com/aws-samples/
amazon-eks-fluent-logging-examples/mainline/examples/fargate/kinesis-firehose/
permissions.json
```

3. 使用上一步中下载的策略文件创建 IAM policy。

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document
 file://permissions.json
```

4. 使用以下命令将 IAM policy 附加到为 Fargate 配置文件指定的容器组（pod）执行角色。请将 *111122223333* 替换为您的账户 ID。将 *AmazonEKSFargatePodExecutionRole* 替换为您的容器组（pod）执行角色（有关更多信息，请参阅 创建 Fargate pod 执行角色 (p. 129)）。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Kubernetes 筛选器支持

此功能需要以下最低 Kubernetes 版本和平台版本或更高版本。

Kubernetes 版本	平台版本
1.22 和更高版本	eks.1
1.21	eks.3
1.20	eks.3
1.19	eks.7
1.18	eks.9

Fluent Bit Kubernetes 筛选器允许您将 Kubernetes 元数据添加到您的日志文件。有关筛选器的更多信息，请参阅 Fluent Bit 文档中的 Kubernetes。您可以使用 API 服务器终端节点应用筛选器。

```
filters.conf: |
    [FILTER]
        Name              kubernetes
        Match             kube.*
        Merge_Log           On
        Buffer_Size         0
        Kube_Meta_Cache_TTL 300s
```

Important

- `Kube_URL`、`Kube_CA_File`、`Kube_Token_Command` 和 `Kube_Token_File` 是服务拥有的配置参数，不能指定。Amazon EKS Fargate 填充这些值。
- `Kube_Meta_Cache_TTL` 是 Fluent Bit 与 API 服务器通信以获取最新元数据等待的时间。如果未指定 `Kube_Meta_Cache_TTL`，则 Amazon EKS Fargate 会追加原定设置值 30 分钟，以减轻 API 服务器的负载。

将 Fluent Bit 流程日志发送到您的账户

您可以使用下面的 `ConfigMap` 将 Fluent Bit 流程日志发送到 Amazon CloudWatch。将 *`region-code`* 替换为集群所在的 Amazon Web Services 区域。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
  labels:
data:
  # Configuration files: server, input, filters and output
  # ======================================================
  flb_log_cw: "true"  #ships fluent-bit process logs to CloudWatch

  output.conf: |
    [OUTPUT]
        Name cloudwatch
        Match kube.*
        region region-code
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
        auto_create_group true
```

日志位于 CloudWatch 下集群所在的 Amazon Web Services 区域中。日志组名称为 *`my-cluster`*`-fluent-bit-logs`，Fluent Bit 日志流名称为 `fluent-bit-`*`podname`*`-`*`pod-namespace`*。

Note

- 仅当 Fluent Bit 流程成功开始后，才会发送流程日志。如果启动 Fluent Bit 时出现故障，则会丢失流程日志。您只能将流程日志发送到 CloudWatch。
- 要调试将流程日志发送到您的账户的过程，可以应用以前的 `ConfigMap` 以获取流程日志。Fluent Bit 无法启动通常是由于您的 `ConfigMap` 在启动时未被 Fluent Bit 解析或接受。

测试应用程序

1. 部署示例 Pod。

   a. 将以下内容保存到计算机上名为 *sample-app*.yaml 的文件中。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
  namespace: same-namespace-as-your-fargate-profile
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - name: http
              containerPort: 80
```

   b. 将清单应用于集群。

```
kubectl apply -f sample-app.yaml
```

2. 使用您在 `ConfigMap` 中配置的目标查看 NGINX 日志。

大小注意事项

我们建议您为日志路由器规划最多 50 MB 的内存。如果您希望应用程序以非常高的吞吐量生成日志，那么您应该规划高达 100 MB 的内存。

问题排查

要确认日志记录功能是否因某种原因（例如无效 `ConfigMap`）而启用还是禁用，以及无效的原因，请使用 **`kubectl describe pod`** ***`pod_name`*** 检查您的 Pod（一组容器）事件。输出可能包含澄清是否已启用日志记录的 Pod 事件，例如以下示例输出。

```
...
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
                      Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
                      kubernetes.io/psp: eks.privileged
...
Events:
```

```
  Type     Reason           Age        From
           Message
  ----     ------           ----       ----
           -------
  Warning  LoggingDisabled  <unknown>  fargate-scheduler
             Disabled logging because aws-logging configmap was not found. configmap "aws-
logging" not found
```

POD 事件是短暂性的，时间长短取决于设置。您也可以使用 **`kubectl describe pod`** ***`pod-name`*** 查看 Pod 的注释。Pod 注释中包含有关日志记录功能是已启用还是已禁用状态以及对应原因的信息。

选择 Amazon EC2 实例类型

Amazon EC2 提供 Worker 节点的各种不同的实例类型。每种实例类型提供不同的计算、内存、存储和网络功能。每个实例也按照这些功能分组到实例系列。有关列表，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的可用实例类型和《适用于 Windows 实例的 Amazon EC2 用户指南》中的可用实例类型。Amazon EKS 发布了多种 Amazon EC2 AMI 的变体以启用支持。要确保您选择的实例类型与 Amazon EKS 兼容，请考虑以下标准。

- 所有 Amazon EKS AMI 当前不支持 `g5g` 和 `mac` 系列。
- Arm 和非加速 Amazon EKS AMI 不支持 `g3`、`g4`、`inf` 和 `p` 系列。
- 加速 Amazon EKS AMI 不支持 `a`、`c`、`hpc`、`m` 和 `t` 系列。

在选择 Amazon EKS 支持的实例类型时，请考虑每种类型的以下功能。

- 节点组中的实例数 – 一般来说，数量较少、规模较大的实例更好，当您有很多 Daemonsets 时更是如此。每个实例都需要对 API 服务器进行 API 调用，因此您拥有的实例越多，API 服务器上的负载就越多。
- 操作系统：查看 Linux、Windows 和 Bottlerocket 支持的实例类型。在创建 Windows 实例之前，请查看 the section called “启用 Windows 支持” (p. 48)。
- 硬件架构 – 您需要 x86 还是 Arm？ 您只能在 Arm 上部署 Linux。在部署 Arm 实例之前，请查看 the section called “Arm” (p. 159)。您需要基于 Nitro System（Linux 或 Windows）构建或是拥有加速功能的实例吗？ 如果您需要加速功能，则只能将 Linux 与 Amazon EKS 结合使用。
- pods 的最大数量 - 由于每个 pod 都分配了自己的 IP 地址，因此实例类型支持的 IP 地址数量是决定实例上可以运行的 pods 数量的因素之一。要手动确定实例类型支持多少个 pods，请参阅 Amazon EKS 建议每种 Amazon EC2 实例类型的最大 pods 数量 (p. 146)。

  > **Note**
  >
  > 如果您使用 `v20220406` 版本或更新版本的经 Amazon EKS 优化的 Amazon Linux 2 AMI，您可以在不升级到最新 AMI 的情况下使用新的实例类型。对于这些 AMI，如果需要的 `max-pods` 值未在 `eni-max-pods.txt` 文件中列出，AMI 会自动计算该值。默认情况下，Amazon EKS 可能不支持当前处于预览状态的实例类型。这些类型的 `max-pods` 的值仍然需要添加到我们的 AMI 中的 `eni-max-pods.txt`。

  Amazon Nitro System 实例类型可选择性地支持比非 Nitro System 实例类型多得多的 IP 地址。但是，并非为实例分配的所有 IP 地址都可用于 pods。要为您的实例分配大量的 IP 地址，您必须在集群中安装 `1.9.0` 版或更高版本的 Amazon VPC CNI 附加组件并进行适当配置。有关更多信息，请参阅提高 Amazon EC2 节点的可用 IP 地址数量 (p. 270)。要为实例分配最大数量的 IP 地址，您必须在 `1.21` 版或更高版本的集群中安装 `1.10.1` 版或更高版本的 Amazon VPC CNI 附加组件，然后使用 `IPv6` 系列部署集群。
- IP 系列 – 您可以在将 `IPv4` 系列用于集群时使用任何受支持的实例类型，从而使您的集群可以将私有 `IPv4` 地址分配到您的 pods 和服务。但是，如果您想将 `IPv6` 系列用于集群，则您必须使用 Amazon Nitro System 实例类型或裸机实例类型。Windows 实例仅支持 `IPv4`。您的集群必须是运行 `1.10.1` 版或更高版本 Amazon VPC CNI 附加组件的新 `1.21` 版或更高版本的集群。有关使用 `IPv6` 的更多信息，请参阅 教程：将 IPv6 地址分配给 pods 和 services (p. 248)。

- 您正在运行的 Amazon VPC CNI 附加组件的版本：最新版本的适用于 Kubernetes 的 Amazon VPC CNI 插件支持这些实例类型。您可能需要更新 Amazon VPC CNI 附加组件版本来利用最新的受支持的实例类型。有关更多信息，请参阅the section called “更新 自行管理的附加组件” (p. 238)。最新版本支持与 Amazon EKS 一起使用的最新功能。以前版本并不支持所有功能。您可以在 GitHub 上的 Changelog 中查看不同版本支持的功能。
- 您要在其中创建节点的 Amazon Web Services 区域 – 并非所有实例类型都在所有 Amazon Web Services 区域中可用。
- 是否使用 pods 的安全组 - 如果使用 pods 的安全组，则仅支持特定实例类型。有关更多信息，请参阅pods 的安全组 (p. 273)。

Amazon EKS 建议每种 Amazon EC2 实例类型的最大 pods 数量

由于每个 pod 都分配了自己的 IP 地址，因此实例类型支持的 IP 地址数量是决定实例上可以运行的 pods 数量的因素之一。Amazon EKS 提供了一个脚本，您可以下载并运行该脚本，以确定 Amazon EKS 建议在每种实例类型上运行的最大 pods 数量。该脚本使用每个实例的硬件属性和配置选项来确定最大 pods 数量。如果您使用的是具有多种实例类型的托管节点组，请使用适用于所有实例类型的值。

1. 下载一个您可以用来计算每种实例类型的最大 pods 数量的脚本。

```
curl -o max-pods-calculator.sh https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/files/max-pods-calculator.sh
```

2. 将脚本标记为您计算机上的可执行文件。

```
chmod +x max-pods-calculator.sh
```

3. 运行脚本，请将 *m5.large* 替换为您计划部署的实例类型，并将 *1.9.0-eksbuild.1* 替换为您的 Amazon VPC CNI 附加组件版本。要确定附加组件版本，请参阅 管理 Amazon VPC CNI plugin for Kubernetes (p. 233) 中的更新过程。

```
./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
```

输出示例如下。

```
29
```

您可以向脚本添加以下选项，以查看使用可选功能时支持的最大 pods 数量。

- `--cni-custom-networking-enabled`：如果要从不同于实例的子网分配 IP 地址，请使用此选项。有关更多信息，请参阅the section called “自定义联网” (p. 258)。将此选项添加到具有相同示例值的上一个脚本中会生成 `20`。

您也可以使用运行带有 `--help` 选项的脚本以查看所有可用的选项。

Amazon EKS 优化版 AMI

您可以通过预构建的 Amazon EKS 优化版 Amazon 机器映像（AMI）或您自己的自定义 AMI 来部署节点。有关每种类型的 Amazon EKS 优化版 AMI 的更多信息，请参阅以下主题之一。有关如何创建您自己的自定义 AMI 的说明，请参阅 Amazon EKS 优化版 Amazon Linux AMI 生成脚本 (p. 161)。

主题

- Amazon EKS 即将结束对 Dockershim 的支持 (p. 147)
- Amazon EKS 优化版 Amazon Linux AMI (p. 147)
- Amazon EKS 优化版 Ubuntu Linux AMI (p. 162)
- Amazon EKS 优化版 Bottlerocket AMI (p. 162)
- Amazon EKS 优化版 Windows AMI (p. 169)

Amazon EKS 即将结束对 Dockershim 的支持

Kubernetes 不再支持 Dockershim。此外，Kubernetes 团队计划完全删除 Kubernetes 版本 1.24 中的运行时。Amazon EKS 将从 Kubernetes 版本 1.24 发布起，结束对 Dockershim 的支持。从 1.24 版开始，正式发布的 Amazon EKS AMI 将 containerd 作为唯一运行时。

您可以使用 kubectl 插件查看您的哪些 Kubernetes 工作负载在安装 Docker 套接字卷。有关更多信息，请参阅 GitHub 上的 Detector for Docker Socket (DDS)。运行 Kubernetes 1.18 至 1.23 版的 Amazon EKS AMI 使用 Docker 作为默认运行时，但是有一个 bootstrap 标志选项，用于使用 containerd 在任何目前支持的集群上测试工作负载。有关更多信息，请参阅启用 containerd 运行时间引导标记 (p. 157)。

我们将继续在现有 Kubernetes 版本上发布 AMI，直到支持日期结束。有关更多信息，请参阅Amazon EKS Kubernetes 发布日历 (p. 65)。如果需要更多时间在 containerd 上测试工作负载，可以继续使用 1.24 版之前的支持版本，且仍然会得到完全支持。但是，当提供的版本是 1.24 且您想将官方 Amazon EKS AMI 升级到该版本或更高版本，请务必确认您的工作负载可以在 containerd 上运行。

如果要升级到 1.24 版或更高版本，同时仍使用 Dockershim，请在安装 Docker 的情况下构建自己的 AMI。但是，当 Kubernetes 移除 Dockershim 支持时，请确保迁移到 containerd 以保持在受支持的版本上。有关更多信息，请参阅 Kubernetes 博客上的 Kubernetes is Moving on From Dockershim: Commitments and Next Steps。

containerd 运行时提供更可靠的性能和安全性，而且这是我们在 Amazon EKS 中标准化的运行时（Fargate 和 Bottlerocket 已经仅使用 containerd）。containerd 有助于最大限度地减少解决 Dockershim 常见漏洞和风险（CVE）所需的 Amazon EKS AMI 版本数量。由于 Dockershim 已经在内部使用 containerd，您可能无需进行任何更改。但是，在某些情况下可能需要或必须要进行更改：

- 您必须对挂载 Docker 套接字的所有应用程序进行更改。例如，使用容器构建容器镜像将受到影响。许多监控工具也挂载了 Docker 套接字。您可能需要等待更新或重新部署工作负载才能进行运行时监控。
- 您可能需要对依赖特定 Docker 设置的所有应用程序进行更改。例如，使用 HTTPS_PROXY 可能需要更改。有关更多信息，请参阅 Docker 文档中的 dockerd。
- 如果您使用 Amazon ECR 凭证助手拉取镜像，则必须切换到 kubelet 镜像凭证提供程序。有关更多信息，请参阅 Kubernetes 文档中的配置 kubelet 镜像凭证提供程序。

Amazon EKS 优化版 Amazon Linux AMI

Amazon EKS 优化版 Amazon Linux AMI 构建于 Amazon Linux 2 之上，并配置为用作 Amazon EKS 节点的基本镜像。此 AMI 配置为与 Amazon EKS 搭配使用，它包含 Docker、kubelet 和 Amazon IAM 身份验证器。

Note

- 您可以在 Amazon Linux 安全中心跟踪 Amazon Linux 2 的安全和隐私事件，或订阅关联的 RSS 源。安全和隐私事件包括问题的概述、受影响的程序包以及如何更新实例以解决问题。
- 在部署加速版或 Arm AMI 之前，请先查看 Amazon EKS 优化版加速型 Amazon Linux AMI (p. 158) 和 Amazon EKS 优化版 Arm Amazon Linux AMI (p. 159) 中的信息。
- Amazon EKS 优化的 Amazon Linux 2 包含可选的 bootstrap 标志，用于启用 containerd 运行时。该功能提供了迁移到 containerd 的清晰路径。Amazon EKS 将从 Kubernetes 版本 1.24

发布起，结束对 Docker 的支持。已在 Kubernetes 社群中广泛采用 `containerd` 运行时，是 CNCF 的一个分级项目。您可以通过将节点组添加到新集群或现有集群来对其进行测试。有关更多信息，请参阅启用 `containerd` 运行时间引导标记 (p. 157)。在版本 `1.21` 的 Amazon EKS 优化版加速型 Amazon Linux AMI 中引导启动时，不支持Amazon Inferentia 工作负载。

在下表中，选择特定于您的 Amazon Linux 实例的 Kubernetes 版本、Amazon Web Services 区域 和处理器类型的 View AMI ID（查看 AMI ID）。您也可以使用 Amazon Systems Manager 参数检索 ID。有关更多信息，请参阅检索 Amazon EKS 优化版 Amazon Linux AMI ID (p. 160)。

1.23

Kubernetes 版本 **`1.23`**

Amazon Web Services 区域	x86	x86 加速版	Arm
美国东部（俄亥俄）(`us-east-2`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(`us-east-1`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(`us-west-2`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(`us-west-1`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
非洲（开普敦）(`af-south-1`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（香港）(`ap-east-1`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(`ap-south-1`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（东京）(`ap-northeast-1`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(`ap-northeast-2`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(`ap-northeast-3`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(`ap-southeast-1`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(`ap-southeast-2`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（`ap-southeast-3`）	查看 AMI ID	查看 AMI ID	查看 AMI ID
加拿大（中部）(`ca-central-1`)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（北京）(`cn-north-1`)	查看 AMI ID	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	x86	x86 加速版	Arm
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

1.22

Kubernetes 版本 **1.22**

Amazon Web Services 区域	x86	x86 加速版	Arm
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	x86	x86 加速版	Arm
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	x86	x86 加速版	Arm
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

1.21

Kubernetes 版本 **1.21**

Amazon Web Services 区域	x86	x86 加速版	Arm
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	x86	x86 加速版	Arm
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

1.20

Kubernetes 版本 **1.20**

Amazon Web Services 区域	x86	x86 加速版	Arm
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	x86	x86 加速版	Arm
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	x86	x86 加速版	Arm
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

1.19

Kubernetes 版本 **1.19**

Amazon Web Services 区域	x86	x86 加速版	Arm
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	x86	x86 加速版	Arm
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

1.18

Kubernetes 版本 **1.18**

Amazon Web Services 区域	x86	x86 加速版	Arm
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	x86	x86 加速版	Arm
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	x86	x86 加速版	Arm
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID

Important

这些 AMI 需要最新的 Amazon CloudFormation 节点模板。您无法将这些 AMI 与先前版本的节点模板配合使用。如果这样做，他们将无法加入您的集群。在尝试使用这些 AMI 之前，确保使用最新的模板更新任何现有 Amazon CloudFormation 节点堆栈。

```
https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/cloudformation/2020-10-29/amazon-eks-nodegroup.yaml
```

Amazon CloudFormation 节点模板使用可触发专用引导脚本的 Amazon EC2 用户数据启动您的节点。此脚本允许您的节点自动发现并连接到您的集群控制层面。有关更多信息，请参阅 启动自行管理的 Amazon Linux 节点 (p. 107)。

启用 containerd 运行时间引导标记

Amazon EKS 优化版 Amazon Linux 2 AMI 包含可选的引导标记，用于启用 containerd 运行时间。该功能提供了迁移到 containerd 的清晰路径。Amazon EKS 将从 Kubernetes 版本 1.24 发布起，结束对 Docker 的支持。有关更多信息，请参阅Amazon EKS 即将结束对 Dockershim 的支持 (p. 147)。

您可以通过创建以下类型的节点组之一来启用引导标记。

- 自行管理 – 按照 启动自行管理的 Amazon Linux 节点 (p. 107) 中的说明创建节点组。为 BootstrapArguments 参数指定 Amazon EKS 优化版 AMI 和以下文本。

```
--container-runtime containerd
```

- 托管式 – 如果使用 eksctl，请创建一个名为 *my-nodegroup*.yaml 的文件，其中含以下内容。将每个 *example value* 替换为您自己的值。要检索 ami-*1234567890abcdef0* 的所需值，您可以使用之前的 AMI 表。

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    overrideBootstrapCommand: |
      #!/bin/bash
```

```
    /etc/eks/bootstrap.sh my-cluster --container-runtime containerd
```

Note

如果同时启动多个节点，您还可以为 `--apiserver-endpoint`、`--b64-cluster-ca` 和 `--dns-cluster-ip` 引导参数指定值以避免错误。有关更多信息，请参阅指定 AMI (p. 103)。

运行以下命令以创建节点组。

```
eksctl create nodegroup -f my-nodegroup.yaml --version 1.23
```

如果您希望使用其他工具来创建托管节点组，则必须使用启动模板部署节点组。在启动模板中，指定 Amazon EKS 优化版 AMI ID，然后使用启动模板部署节点组 (p. 100)，并提供以下用户数据。此用户数据会将实际参数传递到 `bootstrap.sh` 文件中。有关引导文件的更多信息，请参阅 GitHub 上的 bootstrap.sh。

```
/etc/eks/bootstrap.sh my-cluster \
  --container-runtime containerd
```

Amazon EKS 优化版加速型 Amazon Linux AMI

Amazon EKS 优化加速的 Amazon Linux AMI 建立在标准的 Amazon EKS 优化的 Amazon Linux AMI 之上。配置作为 Amazon EKS 节点的可选映像，以支持 GPU 和基于 Inferentia 的工作负载。

除标准 Amazon EKS 优化版 AMI 配置外，加速 AMI 还包含：

- NVIDIA 驱动程序
- `nvidia-container-runtime`（作为默认运行时）
- Amazon Neuron 容器运行时

Note

- Amazon EKS 优化版加速型 AMI 仅支持基于 GPU 和 Inferentia 的实例类型。务必在节点 Amazon CloudFormation 模板中指定这些实例类型。使用 Amazon EKS 优化版加速型 AMI，即表明您同意 NVIDIA 的最终用户许可协议 (EULA)。
- Amazon EKS 优化版加速型 AMI 以前称为带 GPU 支持的 Amazon EKS 优化版 AMI。
- 以前版本的 Amazon EKS 优化加速 AMI 安装 `nvidia-docker` 存储库。Amazon EKS AMI 版本 `v20200529` 及更高版本中不再包含此存储库。

启用基于 GPU 的工作负载的步骤

以下步骤介绍如何使用 Amazon EKS 优化版加速型 AMI 在基于 GPU 的实例上运行工作负载。有关使用基于 Inferentia 的工作负载的更多信息，请参阅 使用 Amazon Inferentia 进行的 Machine Learning 推理 (p. 350)。

1. 当 GPU 节点加入集群后，您必须通过以下命令对集群应用适用于 Kubernetes 的 NVIDIA 设备插件作为 DaemonSet。

   ```
   kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/nvidia-device-plugin.yml
   ```

2. 您可以使用以下命令验证节点是否具有可分配 GPU。

```
kubectl get nodes "-o=custom-columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia
\.com/gpu"
```

部署 pod 以测试是否已正确配置 GPU 节点的步骤

1. 使用以下内容创建名为 nvidia-smi.yaml 的文件。此清单会启动一个 Cuda 容器，该容器在一个节点上运行 nvidia-smi。

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: nvidia/cuda:9.2-devel
    args:
    - "nvidia-smi"
    resources:
      limits:
        nvidia.com/gpu: 1
```

2. 使用下面的命令应用清单。

```
kubectl apply -f nvidia-smi.yaml
```

3. pod 运行完成后，使用下面的命令查看其日志。

```
kubectl logs nvidia-smi
```

输出如下所示。

```
Mon Aug  6 20:23:31 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.26                 Driver Version: 396.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:1C.0 Off |                    0 |
| N/A   46C    P0    47W / 300W |      0MiB / 16160MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Amazon EKS 优化版 Arm Amazon Linux AMI

Arm 实例可以为横向扩展和基于 Arm 的应用程序（例如 Web 服务器、容器化微服务、缓存队列以及分布式数据存储）节省大量成本。当将 Arm 节点添加到集群时，请查看以下注意事项。

注意事项

- 如果您的集群是在 2020 年 8 月 17 日之前部署的，则必须对关键集群附加组件清单进行一次性升级。这样，Kubernetes 可以为集群中使用的每个硬件架构提取正确映像。有关更新集群附加组件的更多信息，请参阅 更新 Amazon EKS 集群的 Kubernetes 版本 (p. 28)。如果您在 2020 年 8 月 17 日或之后部署了集群，则您的 CoreDNS、`kube-proxy` 和 Amazon VPC CNI plugin for Kubernetes 附加组件已经具备多架构功能。
- 部署到 Arm 节点的应用程序必须针对 Arm 进行编译。
- 您不能将 Amazon FSx for Lustre CSI 驱动程序 (p. 220) 与 Arm 一起使用。
- 如果您在现有集群中部署了 DaemonSets，或者希望将它们部署到同时要在其中部署 Arm 节点的新集群中，请验证您的 DaemonSet 是否可以在集群中的所有硬件架构上运行。
- 您可以在同一集群中运行 Arm 节点组和 x86 节点组。如果您这样操作，请考虑将多架构容器镜像部署到容器存储库（如 Amazon Elastic Container Registry），然后将节点选择器添加到清单中，以便 Kubernetes 知道可将 pod 部署到哪个硬件架构上。有关更多信息，请参阅 Amazon ECR 用户指南中的推送多架构映像和 Amazon ECR 的多架构容器映像简介博客文章。

Amazon EKS 优化版 Amazon Linux AMI 版本

Amazon EKS 优化版 Amazon Linux AMI 按 Kubernetes 版本和 AMI 的发布日期进行版本控制，格式如下：

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

过去，此页面包含有关每个 AMI 版本发布的详细信息。详细信息包括 `kubelet`、Docker、Linux 内核、`containerd` 的版本，以及每个新 AMI 版本中提供的 NVIDIA 驱动程序。现在，您可以在 GitHub 上的更改日志中找到该信息。

检索 Amazon EKS 优化版 Amazon Linux AMI ID

您可以使用编程方式，通过查询 Amazon Systems Manager Parameter Store API 来检索 Amazon EKS 优化版 AMI 的 Amazon Machine Image (AMI) ID。此参数使您无需手动查找 Amazon EKS 优化版 AMI ID。有关 Systems Manager Parameter Store API 的更多信息，请参阅 GetParameter。您的用户账户必须具有 `ssm:GetParameter` IAM 权限才能检索 Amazon EKS 优化版 AMI 元数据。

您可以使用 Amazon CLI 或 Amazon Web Services Management Console 检索 AMI ID。

- Amazon CLI – 您可以使用子参数 `image_id`，通过以下命令检索最新推荐的 Amazon EKS 优化版 Amazon Linux AMI 的镜像 ID。将 *`1.23`* 替换为支持的版本 (p. 67)。将 *`region-code`* 替换为集群所在的 Amazon Web Services 区域。将 *`amazon-linux-2`* 替换为 `amazon-linux-2-gpu`，查看加速 AMI ID，替换为 `amazon-linux-2-arm64` 以查看 Arm ID。

  ```
  aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.23/amazon-linux-2/
  recommended/image_id --region region-code --query "Parameter.Value" --output text
  ```

  输出示例如下。

  ```
  ami-1234567890abcdef0
  ```

- Amazon Web Services Management Console – 您可以使用 URL 查询推荐的 Amazon EKS 优化版 AMI ID。该 URL 使用参数的 ID 的值打开 Amazon EC2 Systems Manager 控制台。在以下 URL 中，将 *`1.23`* 替换为支持的版本 (p. 67)，并将 *`region-code`* 替换为需要 AMI ID 的 Amazon EKS 支持的区域。将 *`amazon-linux-2`* 替换为 `amazon-linux-2-gpu`，查看加速 AMI ID，替换为 `amazon-linux-2-arm64` 以查看 Arm ID。

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/eks/optimized-
ami/1.23/amazon-linux-2/recommended/image_id/description?region=region-code
```

Amazon EKS 优化版 Amazon Linux AMI 生成脚本

Amazon Elastic Kubernetes Service（Amazon EKS）具有用于构建 Amazon EKS 优化版 AMI 的开源脚本。GitHub 上提供这些生成脚本。

Amazon EKS 优化版 Amazon Linux AMI 基于 Amazon Linux 2 而构建，专门用作 Amazon EKS 集群中的节点。您可以使用此存储库查看有关 Amazon EKS 团队如何配置 `kubelet`、Docker、适用于 Kubernetes 的 Amazon IAM 身份验证器等项目的详细信息。

生成脚本存储库包含 HashiCorp Packer 模板和生成脚本以生成 AMI。这些脚本是 Amazon EKS 优化版 AMI 生成的可信来源，因此您可关注 GitHub 存储库以监控对 AMI 所做的更改。例如，您可能希望自己的 AMI 使用 Amazon EKS 团队用于正式 AMI 的同一版本的 Docker。

GitHub 存储库还包含在启动时运行的专用引导脚本，以配置实例的证书数据、控制面板端点、集群名称等内容。

此外，GitHub 存储库还包含我们的 Amazon EKS 节点 Amazon CloudFormation 模板。利用这些模板，可以更轻松地运转正在运行 Amazon EKS 优化版 AMI 的实例，并将其注册到集群。

有关更多信息，请参阅 GitHub 上的存储库 (https://github.com/awslabs/amazon-eks-ami)。

Amazon EKS 优化的 Amazon Linux 2 包含可选的 bootstrap 标志，用于启用 `containerd` 运行时。当在适用于 `1.21` 版本的 Amazon EKS 优化版加速型 Amazon Linux AMI 中引导启动时，AmazonInferentia 工作负载不受支持。

为自定义 Amazon Linux AMI 配置 VT1

Amazon EKS 中的自定义 Amazon Linux AMI 可以支持 Amazon Linux 2、Ubuntu 18 和 Ubuntu 20 的 VT1 视频转码实例系列。VT1 支持具有加速 H.264/AVC 和 H.265/HEVC 编解码器的 Xilinx U30 媒体转码卡。要获得这些加速实例的好处，您必须按照以下步骤操作：

1. 从 Amazon Linux 2、Ubuntu 18 或 Ubuntu 20 创建并启动基本 AMI。
2. 基本 AMI 启动后，请安装 XRT 驱动程序和节点上的运行时。
3. 创建集群。
4. 在您的集群上安装 Kubernetes FPGA 插件。

```
kubectl apply -f fpga-device-plugin.yml
```

该插件现在将在 Amazon EKS 集群上为每个节点宣传 Xilinx U30 设备。您可以使用 FFMPEG Docker 镜像在 Amazon EKS 集群上运行示例视频转码工作负载。

为自定义 Amazon Linux 2 AMI 配置 DL1

Amazon EKS 中的自定义 Amazon Linux 2 AMI 可以通过额外的配置和 Kubernetes 附加组件大规模支持深度学习工作负载。本文档介绍了为本地设置或作为较大云配置中的基准设置通用 Kubernetes 解决方案所需的组件。要支持此函数，您必须在自定义环境中执行以下步骤：

- SynapaseAI® Software 驱动程序加载到系统中：这些驱动程序包含在 Github 上可用的 AMI。

  Habana 设备插件 — 一个 Daemonset，允许您在 Kubernetes 集群中自动启用 Habana 设备注册并跟踪设备运行状况。

- Kubernetes 版本介于 `1.19` 至 `1.21` 之间（通过测试确认）。
- Helm 3.x
- 用于安装 MPI Operator 的 Helm Chart。
- MPI Operator

1. 从 Amazon Linux 2、Ubuntu 18 或 Ubuntu 20 创建并启动基本 AMI。
2. 跟随这些说明以为 DL1 设置环境。

Amazon EKS 优化版 Ubuntu Linux AMI

> **Important**
>
> 此功能在 Amazon Web Services 区域（中国）不可用。

Canonical 还与 Amazon EKS 合作创建了可在您的集群中使用的节点 AMI。

Canonical 提供专用的 Kubernetes 节点操作系统映像。这是针对 Amazon EKS 优化的最小化 Ubuntu 镜像，包含与 Amazon 联合开发的自定义 Amazon 内核。有关更多信息，请参阅 Ubuntu 和 Amazon Elastic Kubernetes Service 以及 Ubuntu`18.04` 上对于 Amazon EKS 的优化支持。

Amazon EKS 优化版 Bottlerocket AMI

Amazon EKS 优化版 Bottlerocket AMI 基于 Bottlerocket 构建。Bottlerocket 是一个基于 Linux 的开源操作系统，由 Amazon 专门为在虚拟机或裸机主机上运行容器而打造。该系统安全，只包含运行容器所需的最低限度软件包。这样会减少攻击面和漏洞的影响，同时，也更容易满足节点合规性要求。有关 Bottlerocket 的更多信息，请参阅 GitHub 上的文档和发布。

AMI 配置为与 Amazon EKS 搭配使用，它包含 Containerd 和 `kubelet`。

以下几点是对您的 AMI 类型使用 Bottlerocket 时的一些注意事项：

- 您可以部署到具有 x86 或 Arm 处理器的 Amazon EC2 实例，但不能部署到具有 Inferentia 芯片的实例。
- 没有 Amazon CloudFormation 模板用来部署节点。
- Bottlerocket 图像不附带 SSH 服务器或 shell。您可以使用带外访问方法来允许 SSH 启用管理员容器，并执行一些带有用户数据的引导启动配置步骤。有关更多信息，请参阅 GitHub 上的 bottlerocket README.md：
  - Exploration (探索)
  - 管理员容器
  - Kubernetes 设置
- 预设情况下，Bottlerocket 启用了一个控制容器。该容器运行 Amazon Systems Manager 代理，您可以用它在 Amazon EC2 Bottlerocket 实例上运行命令或启动 shell 会话。有关更多信息，请参阅 Amazon Systems Manager 用户指南中的设置会话管理器。
- 如果创建节点组时提供 SSH 密钥，Bottlerocket 还启用了一个管理员容器。我们建议仅将管理员容器用于开发和测试场景。我们建议不要在生产环境中使用此模式。

从以下的一个表中选择一个链接，以查看某一区域和 Kubernetes 版本的最新 Amazon EKS 优化版 Bottlerocket AMI ID。您也可以使用不同工具，通过 Amazon Systems Manager 参数检索 ID。有关更多信息，请参阅检索 Amazon EKS 优化版 Bottlerocket AMI ID (p. 168)。

1.23

Kubernetes 版本 **1.23**

Amazon Web Services 区域	x86（标准）	x86（NVIDIA）	Arm（标准）	Arm（NVIDIA）
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	不适用	查看 AMI ID	不适用
中国（宁夏）(cn-northwest-1)	查看 AMI ID	不适用	查看 AMI ID	不适用
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID

1.22

Kubernetes 版本 **1.22**

Amazon Web Services 区域	x86（标准）	x86（NVIDIA）	Arm（标准）	Arm（NVIDIA）
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	不适用	查看 AMI ID	不适用
中国（宁夏）(cn-northwest-1)	查看 AMI ID	不适用	查看 AMI ID	不适用
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID

1.21

Kubernetes 版本 **1.21**

Amazon Web Services 区域	x86（标准）	x86（NVIDIA）	Arm（标准）	Arm（NVIDIA）
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	不适用	查看 AMI ID	不适用
中国（宁夏）(cn-northwest-1)	查看 AMI ID	不适用	查看 AMI ID	不适用
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID	查看 AMI ID	查看 AMI ID

1.20

Kubernetes 版本 **1.20**

Amazon Web Services 区域	x86	Arm
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID

1.19

Kubernetes 版本 **1.19**

Amazon Web Services 区域	x86	Arm
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID

1.18

Kubernetes 版本 **1.18**

Amazon Web Services 区域	x86	Arm
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3）	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID

检索 Amazon EKS 优化版 Bottlerocket AMI ID

您可以通过查询 Amazon Systems Manager Parameter Store API 来检索 Amazon EKS 优化版 AMI 的 Amazon Machine Image (AMI) ID。使用此参数，您无需手动查找 Amazon EKS 优化版 AMI ID。有关 Systems Manager Parameter Store API 的更多信息，请参阅 GetParameter。您的用户账户必须具有 `ssm:GetParameter` IAM 权限才能检索 Amazon EKS 优化版 AMI 元数据。

您可以使用 Amazon CLI 或 Amazon Web Services Management Console 检索 AMI ID。

- Amazon CLI – 您可以使用子参数 `image_id`，通过以下 Amazon CLI 命令检索推荐的最新的 Amazon EKS 优化版 Bottlerocket AMI 的镜像 ID。请将 *1.23* 替换为支持的版本 (p. 67)，并将 *region-code* 替换为您需要 AMI ID 的 Amazon EKS 支持的区域。

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.23/x86_64/latest/
image_id --region region-code --query "Parameter.Value" --output text
```

输出示例如下。

```
ami-1234567890abcdef0
```

- Amazon Web Services Management Console – 您可以使用 Amazon Web Services Management Console 中的 URL 查询推荐的 Amazon EKS 优化版 AMI ID。该 URL 使用参数的 ID 的值打开 Amazon EC2 Systems Manager 控制台。在以下 URL 中，将 *1.23* 替换为支持的版本 (p. 67)，并将 *region-code* 替换为需要 AMI ID 的 Amazon EKS 支持的区域。

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-
k8s-1.23/x86_64/latest/image_id/description?region=region-code
```

Amazon EKS 优化版 Windows AMI

Windows Amazon EKS 优化版 AMI 基于 Windows Server 2019 构建。它们被配置作为 Amazon EKS 节点的基本映像。默认情况下，AMI 包括以下组件：

- `kubelet`
- `kube-proxy`
- Amazon IAM 身份验证器
- CSI 代理
- Docker（Amazon EKS 版本 `1.22` 及更早版本）
- Containerd（Amazon EKS 版本 `1.21` 及更高版本）

> **Note**
>
> 您可以使用 Microsoft 安全更新指南跟踪 Windows Server 的安全或隐私事件。

Amazon EKS 提供了已针对 Windows 容器进行了优化的 AMI，包括以下变体：

- Amazon EKS 优化版 Windows Server 2019 Core AMI
- Amazon EKS 优化版 Windows Server 2019 Full AMI

> **Important**
>
> Amazon EKS 优化版 Windows Server 20H2 Core AMI 已弃用。不会发布此 AMI 的任何新版本。

以下各表中显示最新的 Amazon EKS 优化版 AMI ID。您也可以使用不同工具，通过 Amazon Systems Manager 参数检索 ID。有关更多信息，请参阅检索 Amazon EKS 优化版 Windows AMI ID (p. 187)。

Windows Server 2019 是长期服务渠道 (LTSC, Long-Term Servicing Channel) 版本，20H2 版是半年渠道 (SAC, Semi-Annual Channel) 版本。我们不再支持 SAC 版本。有关这些版本类型的详细信息，请参阅 Windows 文档中的 Microsoft 服务器版本信息。有关 Windows 操作系统版本支持的更多信息，请参阅 Kubernetes 中的 Windows 支持简介。

1.23

Kubernetes 版本 **1.23**

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3） Note 如果 SSM 参数当前不可用，您可以从 Amazon EC2 控制台搜索中对 AMI ID 进行硬编码（基于 Windows/Kube 变体）。在 Launch an instance（启动实例）页面上，搜索“**eks windows**”，然后切换到 Community AMIs（社区 AMI）选项卡。	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID

1.22

Kubernetes 版本 **1.22**

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3） Note 如果 SSM 参数当前不可用，您可以从 Amazon EC2 控制台搜索中对 AMI ID 进行硬编码（基于 Windows/Kube 变体）。在 Launch an instance（启动实例）页面上，搜索“**eks windows**”，然后切换到 Community AMIs（社区 AMI）选项卡。	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID

1.21

Kubernetes 版本 **1.21**

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3） Note 如果 SSM 参数当前不可用，您可以从 Amazon EC2 控制台搜索中对 AMI ID 进行硬编码（基于 Windows/Kube 变体）。在 Launch an instance（启动实例）页面上，搜索“**eks windows**”，然	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
后切换到 Community AMIs（社区 AMI）选项卡。		
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID

1.20

Kubernetes 版本 **1.20**

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3） Note 如果 SSM 参数当前不可用，您可以从 Amazon EC2 控制台搜索中对 AMI ID 进行硬编码（基于 Windows/Kube 变体）。在 Launch an instance（启动实例）页面上，搜索“**eks windows**”，然后切换到 Community AMIs（社区 AMI）选项卡。	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID

1.19

Kubernetes 版本 **1.19**

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3） Note 如果 SSM 参数当前不可用，您可以从 Amazon EC2 控	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
制台搜索中对 AMI ID 进行硬编码（基于 Windows/Kube 变体）。在 Launch an instance（启动实例）页面上，搜索“**eks windows**”，然后切换到 Community AMIs（社区 AMI）选项卡。		
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID

1.18

Kubernetes 版本 **1.18**

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
美国东部（俄亥俄）(us-east-2)	查看 AMI ID	查看 AMI ID
美国东部（弗吉尼亚北部）(us-east-1)	查看 AMI ID	查看 AMI ID
美国西部（俄勒冈）(us-west-2)	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
美国西部（加利福尼亚北部）(us-west-1)	查看 AMI ID	查看 AMI ID
非洲（开普敦）(af-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（香港）(ap-east-1)	查看 AMI ID	查看 AMI ID
亚太地区（孟买）(ap-south-1)	查看 AMI ID	查看 AMI ID
亚太地区（东京）(ap-northeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（首尔）(ap-northeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（大阪）(ap-northeast-3)	查看 AMI ID	查看 AMI ID
亚太地区（新加坡）(ap-southeast-1)	查看 AMI ID	查看 AMI ID
亚太地区（悉尼）(ap-southeast-2)	查看 AMI ID	查看 AMI ID
亚太地区（雅加达）（ap-southeast-3） Note 如果 SSM 参数当前不可用，您可以从 Amazon EC2 控制台搜索中对 AMI ID 进行硬编码（基于 Windows/Kube 变体）。在 Launch an instance（启动实例）页面上，搜索“**eks windows**”，然后切换到 Community AMIs（社区 AMI）选项卡。	查看 AMI ID	查看 AMI ID
加拿大（中部）(ca-central-1)	查看 AMI ID	查看 AMI ID
中国（北京）(cn-north-1)	查看 AMI ID	查看 AMI ID
中国（宁夏）(cn-northwest-1)	查看 AMI ID	查看 AMI ID
欧洲（法兰克福）(eu-central-1)	查看 AMI ID	查看 AMI ID
欧洲（爱尔兰）(eu-west-1)	查看 AMI ID	查看 AMI ID
欧洲（伦敦）(eu-west-2)	查看 AMI ID	查看 AMI ID

Amazon Web Services 区域	Windows Server 2019 Core	Windows Server 2019 Full
欧洲（米兰）(eu-south-1)	查看 AMI ID	查看 AMI ID
欧洲（巴黎）(eu-west-3)	查看 AMI ID	查看 AMI ID
欧洲（斯德哥尔摩）(eu-north-1)	查看 AMI ID	查看 AMI ID
中东（巴林）(me-south-1)	查看 AMI ID	查看 AMI ID
南美洲（圣保罗）(sa-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国东部）(us-gov-east-1)	查看 AMI ID	查看 AMI ID
Amazon GovCloud（美国西部）(us-gov-west-1)	查看 AMI ID	查看 AMI ID

Amazon EKS Windows AMI 发布日历

下表列出了 Amazon EKS 上的 Windows 版本的发布日期和支持终止日期。如果终止日期为空，则是因为相应版本仍受支持。

Windows 版本	Amazon EKS 版本	Amazon EKS 支持终止
Windows Server 20H2 Core	8/12/2021	8/9/2022
Windows Server 2004 Core	8/19/2020	12/14/2021
Windows Server 2019 Core	10/7/2019	
Windows Server 2019 Full	10/7/2019	
Windows Server 1909 Core	10/7/2019	12/8/2020

引导脚本配置参数

创建 Windows 节点时，节点上有一个允许配置不同参数的脚本。根据您的设置，可以在节点上类似于以下位置：`C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1` 找到此脚本。此脚本包含以下参数：

- `-EKSClusterName`：指定此 Worker 节点要加入的 Amazon EKS 集群名称。
- `-KubeletExtraArgs`：为 `kubelet` 指定额外的参数（可选）。
- `-KubeProxyExtraArgs`：为 `kube-proxy` 指定额外的参数（可选）。
- `-APIServerEndpoint`：指定 Amazon EKS 集群 API 服务器端点（可选）。仅在与 `-Base64ClusterCA` 一起使用时才有效。绕过调用 `Get-EKSCluster`。
- `-Base64ClusterCA`：指定 base64 编码的集群 CA 内容（可选）。仅在与 `-APIServerEndpoint` 一起使用时才有效。绕过调用 `Get-EKSCluster`。
- `-DNSClusterIP`：覆盖用于集群内 DNS 查询的 IP 地址（可选）。基于主接口的 IP 地址，默认值为 `10.100.0.10` 或 `172.20.0.10`。
- `-ContainerRuntime` – 指定要在节点上使用的容器运行时。

启用 containerd 运行时间引导标记

Amazon EKS 优化版 Windows AMI 包含可选的引导标记，用于启用 containerd 运行时间。该功能为您提供了迁移到 containerd 的清晰路径。Amazon EKS 将从 Kubernetes 版本 1.24 发布起，结束对 Docker 的支持。有关更多信息，请参阅Amazon EKS 即将结束对 Dockershim 的支持 (p. 147)。

在 Kubernetes 版本 1.24 之前，容器运行时支持的值为 docker 和 containerd，在使用 eksctl 或 Amazon Web Services Management Console 启动 Windows 节点时指定。

- 如果指定的值为 docker 且 Amazon EKS 版本为 1.23 或更早版本，则将 Docker 作为节点上的运行时。
- 如果指定的值为 containerd 且 Amazon EKS 版本高于 1.20，则选中 containerd 作为运行时。如果 Amazon EKS 版本低于 1.21，则引导会失败，节点无法加入集群。
- 如果指定了任何其他值，则引导程序将失败，节点无法加入集群。
- 如果未指定此标记，则选择容器运行时的默认值。对于 Amazon EKS 版本 1.23 和更早版本，默认值为 Docker。对于 1.24 和更高版本的集群，默认值为 containerd。

在启动 Amazon EKS 集群中的 Windows 节点时，请按照 启动自行管理的 Windows 节点 (p. 114) 中的步骤操作，使用 containerd 运行时的 Windows 自行管理的节点可以使用 eksctl 或 Amazon Web Services Management Console 启动。

eksctl

要使用 **eksctl** 启用 **containerd** 运行时

对于 Windows 自行管理的节点，可以在创建新节点组时在配置中指定容器运行时。您可以使用以下 **test-windows-with-containerd**.yaml 作为参考。

> Note
>
> 您必须使用 eksctl 版本 0.95.0 或更高版本以在配置文件中使用 containerRuntime 设置。

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-containerd-cluster
  region: us-west-2
  version: '1.21'

nodeGroups:
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2019FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
    containerRuntime: containerd
  - name: linux-ng
    amiFamily: AmazonLinux2
    minSize: 2
    maxSize: 3
```

然后，可以使用以下命令创建节点组。

```
eksctl create cluster -f test-windows-with-containerd.yaml
```

或者，您也可以将 EKS_CONTAINER_RUNTIME 环境变量指定为 eksctl 配置文件中的预引导命令。

```
preBootstrapCommands:
  - Invoke-Expression -Command
 '[Environment]::SetEnvironmentVariable("EKS_CONTAINER_RUNTIME", "containerd",
 [System.EnvironmentVariableTarget]::Machine)'
```

有关更多信息，请参阅 `eksctl` 文档中的从配置文件创建节点组、定义 containerd 运行时和配置文件架构。

Amazon Web Services Management Console

要使用 Amazon Web Services Management Console 启用 **`containerd`** 运行时

在 Amazon CloudFormation 模板中，有一个名为 `BootstrapArguments` 的参数，它可用于将额外的参数传递给引导脚本。名为 `ContainerRuntime` 的参数可用于在节点上选择特定的运行时间。

在 `BootstrapArguments` 中指定以下内容以启用 `containerd` 运行时。

```
-ContainerRuntime containerd
```

Amazon ECS 优化版 Windows AMI 版本

本主题列出了 Amazon EKS 优化版 Windows AMI 的版本及其 `kubelet`、Docker、containerd 和 `csi-proxy` 的相应版本。

每个变体的 Amazon EKS 优化版 AMI 元数据（包括 AMI ID）均可通过编程方式检索。有关更多信息，请参阅检索 Amazon EKS 优化版 Windows AMI ID (p. 187)。

AMI 按 Kubernetes 版本和 AMI 的发布日期进行版本控制，格式如下：

```
k8s_major_version.k8s_minor_version-release_date
```

Amazon EKS 优化版 Windows Server 2019 Core AMI

以下各表列出了 Amazon EKS 优化版 Windows AMI 的当前版本和以前版本。

Kubernetes version 1.23

Kubernetes 版本 **`1.23`**

AMI 版本	**`kubelet`** 版本	Docker 版本	**`containerd`** 版本	**`csi-proxy`** 版本
`1.23-2022.08.09`	`1.23.7`	`20.10.9`	`1.6.2`	`1.1.1`

Kubernetes version 1.22

Kubernetes 版本 **`1.22`**

AMI 版本	**`kubelet`** 版本	Docker 版本	**`containerd`** 版本	**`csi-proxy`** 版本
`1.22-2022.08.09`	`1.22.6`	`20.10.9`	`1.6.2`	`1.1.1`
`1.22-2022.07.20`	`1.22.6`	`20.10.9`	`1.6.2`	`1.1.1`
`1.22-2022.06.17`	`1.22.6`	`20.10.9`	`1.6.2`	`1.1.1`

AMI 版本	**kubelet** 版本	Docker 版本	**containerd** 版本	**csi-proxy** 版本
1.22-2022.05.16	1.22.6	20.10.9	1.6.2	1.1.1
1.22-2022.04.14	1.22.6	20.10.9	1.6.2	1.1.1

Kubernetes version 1.21

Kubernetes 版本 **1.21**

AMI 版本	**kubelet** 版本	Docker 版本	**containerd** 版本	**csi-proxy** 版本
1.21-2022.08.09	1.21.5	20.10.9	1.6.2	1.1.1
1.21-2022.07.20	1.21.5	20.10.9	1.6.2	1.1.1
1.21-2022.06.17	1.21.5	20.10.9	1.6.2	1.1.1
1.21-2022.05.16	1.21.5	20.10.9	1.6.2	1.1.1
1.21-2022.04.14	1.21.5	20.10.9	1.6.2	1.1.1
1.21.2021.03.10	1.21.5	20.10.9	1.6.1	N/A
1.21.2022.02.23	1.21.5	20.10.9	N/A	N/A
1.21.2022.01.18	1.21.5	20.10.9	N/A	N/A
1.21.2021.12.21	1.21.5	20.10.8	N/A	N/A
1.21-2021.10.14	1.21.4	20.10.7	N/A	N/A
1.21-2021.11.10	1.21.4	20.10.7	N/A	N/A
1.21-2021.09.16	1.21.2	20.10.7	N/A	N/A
1.21-2021.08.12	1.21.2	20.10.6	N/A	N/A

Kubernetes version 1.20

Kubernetes 版本 **1.20**

AMI 版本	**kubelet** 版本	Docker 版本	**csi-proxy** 版本
1.20-2022.08.09	1.20.11	20.10.9	1.1.1
1.20-2022.07.20	1.20.11	20.10.9	1.1.1
1.20-2022.06.17	1.20.11	20.10.9	1.1.1
1.20-2022.05.16	1.20.11	20.10.9	1.1.1
1.20-2022.04.14	1.20.11	20.10.9	1.1.1
1.20-2022.03.10	1.20.11	20.10.9	N/A
1.20-2022.02.23	1.20.11	20.10.9	N/A

AMI 版本	**kubelet** 版本	Docker 版本	**csi-proxy** 版本
1.20-2022.01.8	1.20.11	20.10.9	N/A
1.20-2021.12.21	1.20.11	20.10.8	N/A
1.20-2021.11.10	1.20.10	20.10.7	N/A
1.20-2021.10.14	1.20.10	20.10.7	N/A
1.20-2021.09.16	1.20.7	20.10.7	N/A
1.20-2021.08.12	1.20.4	20.10.6	N/A
1.20-2021.07.14	1.20.4	20.10.6	N/A
1.20-2021.06.16	1.20.4	20.10.5	N/A
1.20-2021.05.18	1.20.4	20.10.4	N/A

Kubernetes version 1.19

Kubernetes 版本 **1.19**

AMI 版本	**kubelet** 版本	Docker 版本	**csi-proxy** 版本
1.19-2022.06.17	1.19.15	20.10.9	1.1.1
1.19-2022.05.16	1.19.15	20.10.9	1.1.1
1.19-2022.04.14	1.19.15	20.10.9	1.1.1
1.19-2022.03.10	1.19.15	20.10.9	N/A
1.19-2022.02.23	1.19.15	20.10.9	N/A
1.19-2022.01.18	1.19.15	20.10.9	N/A
1.19-2021.12.21	1.19.15	20.10.8	N/A
1.19-2021.11.10	1.19.14	20.10.7	N/A
1.19-2021.10.14	1.19.14	20.10.7	N/A
1.19-2021.09.16	1.19.13	20.10.7	N/A
1.19-2021.08.12	1.19.8	20.10.6	N/A
1.19-2021.07.14	1.19.8	20.10.6	N/A
1.19-2021.06.16	1.19.8	20.10.5	N/A
1.19-2021.05.18	1.19.8	20.10.4	N/A
1.19-2021.04.14	1.19.6	20.10.0	N/A
1.19-2021.03.10	1.19.6	19.03.14	N/A
1.19-2021.02.18	1.19.6	19.03.14	N/A

Kubernetes version 1.18

Kubernetes 版本 **1.18**

AMI 版本	**kubelet** 版本	Docker 版本	**csi-proxy** 版本
1.18-2022.03.10	1.18.20	20.10.9	1.1.1
1.18-2022.02.23	1.18.20	20.10.9	N/A
1.18-2022.01.18	1.18.20	20.10.9	N/A
1.18-2021.12.21	1.18.20	20.10.8	N/A
1.18-2021.11.10	1.18.20	20.10.7	N/A
1.18-2021.10.14	1.18.20	20.10.7	N/A
1.18-2021.09.16	1.18.20	20.10.7	N/A
1.18-2021.08.12	1.18.16	20.10.6	N/A
1.18-2021.07.14	1.18.16	20.10.6	N/A
1.18-2021.06.16	1.18.16	20.10.5	N/A
1.18-2021.05.18	1.18.16	20.10.4	N/A
1.18-2021.04.14	1.18.9	20.10.0	N/A
1.18-2021.03.10	1.18.9	19.03.14	N/A
1.18-2021.02.10	1.18.9	19.03.14	N/A
1.18-2021.01.13	1.18.9	19.03.14	N/A
1.18-2020.12.11	1.18.9	19.03.13	N/A
1.18-2020.11.12	1.18.9	18.09.7	N/A
1.18-2020.10.29	1.18.8	18.09.7	N/A
1.18-2020.10.08	1.18.8	18.09.7	N/A

Amazon EKS 优化版 Windows Server 2019 Full AMI

以下各表列出了 Amazon EKS 优化版 Windows AMI 的当前版本和以前版本。

Kubernetes version 1.23

Kubernetes 版本 **1.23**

AMI 版本	**kubelet** 版本	Docker 版本	**containerd** 版本	**csi-proxy** 版本
1.23-2022.08.09	1.23.7	20.10.9	1.6.2	1.1.1

Kubernetes version 1.22

Kubernetes 版本 **1.22**

AMI 版本	**kubelet** 版本	Docker 版本	**containerd** 版本	**csi-proxy** 版本
1.22-2022.08.09	1.22.6	20.10.9	1.6.2	1.1.1
1.22-2022.07.20	1.22.6	20.10.9	1.6.2	1.1.1
1.22-2022.06.17	1.22.6	20.10.9	1.6.2	1.1.1
1.22-2022.05.16	1.22.6	20.10.9	1.6.2	1.1.1
1.22-2022.04.14	1.22.6	20.10.9	1.6.2	1.1.1

Kubernetes version 1.21

Kubernetes 版本 **1.21**

AMI 版本	**kubelet** 版本	Docker 版本	**containerd** 版本	**csi-proxy** 版本
1.21.2022.08.11	1.21.5	20.10.9	1.6.2	1.1.1
1.21.2022.07.20	1.21.5	20.10.9	1.6.2	1.1.1
1.21.2022.06.17	1.21.5	20.10.9	1.6.2	1.1.1
1.21.2022.05.16	1.21.5	20.10.9	1.6.2	1.1.1
1.21.2022.04.14	1.21.5	20.10.9	1.6.2	1.1.1
1.21.2022.03.10	1.21.5	20.10.9	1.6.1	N/A
1.21.2022.02.23	1.21.5	20.10.9	N/A	N/A
1.21.2022.01.18	1.21.5	20.10.9	N/A	N/A
1.21.2021.12.21	1.21.5	20.10.8	N/A	N/A
1.21-2021.11.10	1.21.4	20.10.7	N/A	N/A
1.21-2021.10.14	1.21.4	20.10.7	N/A	N/A
1.21-2021.09.16	1.21.2	20.10.7	N/A	N/A
1.21-2021.08.12	1.21.2	20.10.6	N/A	N/A

Kubernetes version 1.20

Kubernetes 版本 **1.20**

AMI 版本	**kubelet** 版本	Docker 版本	**csi-proxy** 版本
1.20-2022.08.09	1.20.11	20.10.9	1.1.1
1.20-2022.07.20	1.20.11	20.10.9	1.1.1

AMI 版本	**kubelet** 版本	Docker 版本	**csi-proxy** 版本
1.20-2022.06.17	1.20.11	20.10.9	1.1.1
1.20-2022.05.16	1.20.11	20.10.9	1.1.1
1.20-2022.04.14	1.20.11	20.10.9	1.1.1
1.20-2022.03.10	1.20.11	20.10.9	N/A
1.20-2022.02.23	1.20.11	20.10.9	N/A
1.20-2022.01.8	1.20.11	20.10.9	N/A
1.20-2021.12.21	1.20.11	20.10.8	N/A
1.20-2021.11.10	1.20.10	20.10.7	N/A
1.20-2021.10.14	1.20.10	20.10.7	N/A
1.20-2021.09.16	1.20.7	20.10.7	N/A
1.20-2021.08.12	1.20.4	20.10.6	N/A
1.20-2021.07.14	1.20.4	20.10.6	N/A
1.20-2021.06.16	1.20.4	20.10.5	N/A
1.20-2021.05.18	1.20.4	20.10.4	N/A

Kubernetes version 1.19

Kubernetes 版本 **1.19**

AMI 版本	**kubelet** 版本	Docker 版本	**csi-proxy** 版本
1.19-2022.06.17	1.19.15	20.10.9	1.1.1
1.19-2022.05.16	1.19.15	20.10.9	1.1.1
1.19-2022.04.14	1.19.15	20.10.9	1.1.1
1.19-2022.03.10	1.19.15	20.10.9	N/A
1.19-2022.02.23	1.19.15	20.10.9	N/A
1.19-2022.01.18	1.19.15	20.10.9	N/A
1.19-2021.12.21	1.19.15	20.10.8	N/A
1.19-2021.11.10	1.19.14	20.10.7	N/A
1.19-2021.10.14	1.19.14	20.10.7	N/A
1.19-2021.09.16	1.19.13	20.10.7	N/A
1.19-2021.08.12	1.19.8	20.10.6	N/A
1.19-2021.07.14	1.19.8	20.10.6	N/A
1.19-2021.06.16	1.19.8	20.10.5	N/A

AMI 版本	**kubelet** 版本	Docker 版本	**csi-proxy** 版本
1.19-2021.05.18	1.19.8	20.10.4	N/A
1.19-2021.04.14	1.19.6	20.10.0	N/A
1.19-2021.03.10	1.19.6	19.03.14	N/A
1.19-2021.02.18	1.19.6	19.03.14	N/A

Kubernetes version 1.18

Kubernetes 版本 **1.18**

AMI 版本	**kubelet** 版本	Docker 版本	**csi-proxy** 版本
1.18-2022.03.10	1.18.20	20.10.9	1.1.1
1.18-2022.02.23	1.18.20	20.10.9	N/A
1.18-2022.01.18	1.18.20	20.10.9	N/A
1.18-2021.12.21	1.18.20	20.10.8	N/A
1.18-2021.11.10	1.18.20	20.10.7	N/A
1.18-2021.10.14	1.18.20	20.10.7	N/A
1.18-2021.09.16	1.18.20	20.10.7	N/A
1.18-2021.08.12	1.18.16	20.10.6	N/A
1.18-2021.07.14	1.18.16	20.10.6	N/A
1.18-2021.06.16	1.18.16	20.10.5	N/A
1.18-2021.05.18	1.18.16	20.10.4	N/A
1.18-2021.04.14	1.18.9	20.10.0	N/A
1.18-2021.03.10	1.18.9	19.03.14	N/A
1.18-2021.02.10	1.18.9	19.03.14	N/A
1.18-2021.01.13	1.18.9	19.03.14	N/A
1.18-2020.12.11	1.18.9	19.03.13	N/A
1.18-2020.11.12	1.18.9	18.09.7	N/A
1.18-2020.10.29	1.18.8	18.09.7	N/A
1.18-2020.10.08	1.18.8	18.09.7	N/A

检索 Amazon EKS 优化版 Windows AMI ID

您可以使用编程方式，通过查询 Amazon Systems Manager Parameter Store API 来检索 Amazon EKS 优化版 AMI 的 Amazon Machine Image (AMI) ID。此参数使您无需手动查找 Amazon EKS 优化版 AMI ID。有关 Systems Manager Parameter Store API 的更多信息，请参阅 GetParameter。您的用户账户必须具有 `ssm:GetParameter` IAM 权限才能检索 Amazon EKS 优化版 AMI 元数据。

您可以使用 Amazon CLI 或 Amazon Web Services Management Console 检索 AMI ID。

- Amazon CLI – 您可以使用子参数 `image_id`，通过以下命令检索推荐的最新 Amazon EKS 优化版 Windows AMI 的镜像 ID。您可以将 *`1.23`* 替换为任何受支持的 Amazon EKS 版本，将 *`region-code`* 替换为需要 AMI ID 的 Amazon EKS 支持的区域。请将 *`Core`* 替换为 `Full`，以查看 Windows Server 的完整 AMI ID。

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2019-
English-Core-EKS_Optimized-1.23/image_id --region region-code --query "Parameter.Value"
 --output text
```

  输出示例如下。

```
ami-1234567890abcdef0
```

- Amazon Web Services Management Console – 您可以使用 URL 查询推荐的 Amazon EKS 优化版 AMI ID。该 URL 使用参数的 ID 的值打开 Amazon EC2 Systems Manager 控制台。在以下 URL 中，您可以将 *`1.22`* 替换为任何受支持的 Amazon EKS 版本，可将 *`region-code`* 替换为需要 AMI ID 的 Amazon EKS 支持的区域。请将 *`Core`* 替换为 `Full`，以查看 Windows Server 的完整 AMI ID。

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/ami-windows-latest/
Windows_Server-2019-English-Core-EKS_Optimized-1.23/image_id/description?region=region-
code
```

Amazon EKS 优化版 Windows AMI

您可以使用 Amazon EC2 Image Builder 创建自定义 Amazon EKS 优化版 Windows AMI。您必须创建自己的 Image Builder 配方。有关更多信息，请参阅 EC2 Image Builder 用户指南中的创建镜像配方和版本。在创建配方并选择 Source image（源镜像）时，您可使用以下选项：

- Select managed images（选择托管式镜像）– 如果选择此选项，则可以为 Image origin（镜像原点）选择以下选项之一。
  - Quick start (Amazon-managed)（快速入门（Amazon 托管））– 在 Image name（镜像名称）下拉列表中，选择一个 Amazon EKS 支持的 Windows Server 版本 (p. 169)。
  - Images owned by me（我拥有的镜像）– 对于 Image name（镜像名称），请选择具有自带许可证的自有镜像的 ARN。您提供的镜像尚未安装 Amazon EKS 组件。
- Enter custom AMI ID（输入自定义 AMI ID）– 对于 AMI ID，请输入具有自带许可证的 AMI 的 ID。您提供的镜像尚未安装 Amazon EKS 组件。

在 Build components - Windows（构建组件 – Windows）下的搜索框中，在下拉列表中选择 Amazon-managed（Amazon 托管），然后搜索 `eks`。选择 eks-optimized-ami-windows 搜索结果，即使返回的结果可能不是您想要的版本。在 Selected components（选定的组件）下，选择 Versioning options（版本控制选项），然后选择 Specify component version（指定组件版本）。输入 *`version`*`.x`，以将 *`version`* 替换为支持的 Kubernetes 版本 (p. 59)。如果您输入 `1.23.x` 作为组件版本，则您的 Image Builder 管道将使用最新的 `1.23.x` `kubelet` 版本构建 AMI。

如果要确定哪些 `kubelet` 和 Docker 版本随组件一起安装，请在左侧导航窗格中选择 Components（组件）。在 Components（组件）下，将 Owned by me（我拥有的）更改为 Quick start (Amazon-managed)（快速入门（Amazon 托管））。在 Find components by name（按名称查找组件）框中，输入 `eks`。搜索结果显示的是针对每个支持的 Kubernetes 版本返回的组件中的 `kubelet` 和 Docker 版本。这些组件在 Amazon EKS 支持的 Windows 版本上进行功能测试。不支持任何其他 Windows 版本，并且可能与相应组件不兼容。

还包括操作系统最新 Windows 补丁的 `update-windows` 组件。

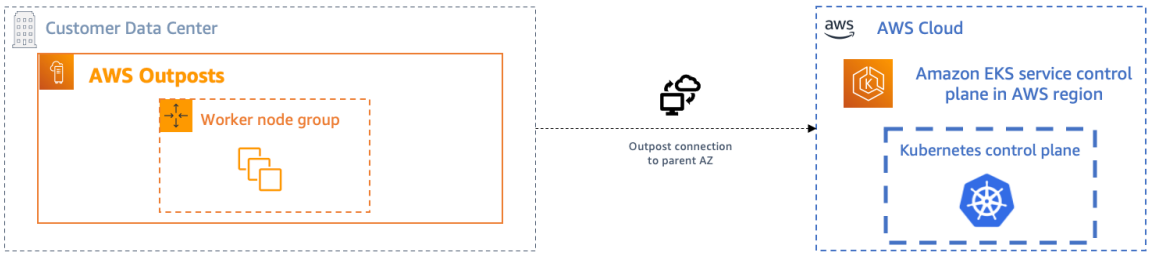
Amazon Outposts 上的 Amazon EKS 节点

Important

此功能在 Amazon Web Services 区域（中国）不可用。

您可以在 Amazon Outposts 上创建和运行 Amazon EKS 节点。Amazon Outposts 在本地设施中支持本机 Amazon 服务、基础设施和操作模型，以满足低延迟、本地数据处理和数据驻留需求。在 Amazon Outposts 环境中，您可以使用与 Amazon 云中相同的 Amazon API、工具和基础设施。有关 Amazon Outposts 的更多信息，请参阅《Amazon Outposts 用户指南》。

您可以使用 Amazon EKS 通过 Amazon Outposts 在本地运行 Kubernetes 应用程序。Amazon Outposts 上的 Amazon EKS 支持扩展集群，其 Kubernetes 控制面板在父级 Amazon 区域中运行，Worker 节点在 Amazon Outposts 上运行。Kubernetes 控制面板完全由 Amazon 托管，您可以使用相同的 Amazon EKS API、工具和控制台在 Amazon Outposts 上创建和运行 Amazon EKS Worker 节点。



先决条件

以下是在 Amazon Outposts 上使用 Amazon EKS 节点的先决条件：

- 您必须已在本地数据中心中安装并配置了 Outpost。有关更多信息，请参阅 Amazon Outposts 用户指南中的创建 Outpost 并订购 Outpost 容量。
- Outpost 与其父级 Amazon Web Services 区域 之间必须具有可靠的网络连接。我们建议在 Outpost 与父级 Amazon Web Services 区域 之间提供高度可用的低延迟连接。有关更多信息，请参阅 Amazon Outposts 用户指南中的与本地网络的 Outpost 连接。
- Outpost 的 Amazon Web Services 区域必须支持 Amazon EKS。有关支持的 Amazon Web Services 区域的列表，请参阅 Amazon 一般参考中的 Amazon EKS 服务端点。

Outpost 考虑因素

架构

- Amazon Outposts 提供多种外形规格，包括 1U 和 2U Outposts 服务器和 42U Outposts 机架。Amazon EKS 仅在 42U Outposts 机架上受支持。
- 单个子网不能跨越多个逻辑 Outposts，Outposts 间的流量必须使用客户拥有的 IP 地址并遍历本地网络。因此，建议对每个逻辑 Outpost 运行一个 Amazon EKS 集群。

- 运行于 Amazon Outposts 的 Amazon EKS Worker 节点到 Amazon 区域中的控制面板的流量保留在您的 VPC 内并遍历服务链接连接。您可以使用私有或公有连接进行服务链接连接。有关服务链接的更多信息，请参阅《Amazon Outposts 用户指南》中的到 Amazon 区域的 Outposts 连接。
- 如果 Outpost 与其父级Amazon区域之间的网络连接丢失，Amazon EKS Worker 节点将继续运行。但是，在恢复连接之前，您无法创建新节点或对现有部署执行转换管理操作。在网络断开连接期间，建议的操作方法是尝试按照网络连接检查表重新连接您的 Amazon Outposts 到父级 Amazon 区域。如果实例在网络

连接断开期间出现故障，则不会自动替换该实例。Amazon EKS Kubernetes 控制面板在父级 Amazon 区域中运行，且缺少 kubelet 检测信号可能会导致以下情况：

- Amazon Outposts 上的容器组（pod）被标记为运行状况不佳
- 节点状态超时
- 容器组（pod）将被标记为驱逐

有关更多信息，请参阅 Kubernetes 文档中的节点控制器。

正常运行

- 创建 Amazon EKS 集群时，必须使用在 Amazon 区域中运行的子网，因为这些子网用于创建 Amazon EKS Kubernetes 控制面板。在创建集群过程中，请勿使用在您的 Amazon Outposts 上运行的子网。
- 您只能在 Amazon Outposts 上运行自行管理的节点，不支持托管节点和 Fargate。创建自我管理的节点组时，必须传递已存在于您的 Amazon Outposts 上的子网。
- Amazon EKS Worker 节点的 VPC 必须与 LGW 路由表关联，才能通过本地网络访问在 Worker 节点上运行的服务。如果您使用的是客户拥有的 IP，则运行 Worker 节点的子网必须有通往 LGW 的路由。
- `IPv6` 不能用于服务链接或 Amazon Outposts 上的本地网络流量。如果您计划在 Amazon Outposts 上运行 Amazon EKS Worker 节点，则不应使用 `IPv6` IP 系列创建 Amazon EKS 集群。
- Amazon EKS Worker 节点的 Kubernetes 组件是从父级 Amazon 区域的 Amazon Elastic Container Registry（Amazon ECR）中提取的。在将新的 Amazon EC2 实例引导到 Amazon EKS 集群时，预计服务链接连接的流量会增加。
- 您可以在父级 Amazon 区域中使用 Amazon ECR 托管应用程序容器映像。应用程序容器映像的大小将影响服务链接带宽使用情况和部署尚未缓存的新容器组（pod）时的启动时间。如果需要缩短应用程序部署时间，请考虑托管应用程序容器映像的本地容器注册表或缓存。如果您的“隔离”子网没有通往互联网的路径，则必须为 Amazon S3 和 Amazon ECR 设置 VPC 端点，以在 Amazon Outposts 上创建 Amazon EKS Worker 节点。
- 创建 Amazon EKS Worker 节点时，您必须使用 `gp2` 卷类型。

应用程序

- 使用 Amazon Outposts，您可以在本地运行一些 Amazon 服务，可以连接到父级 Amazon 区域中提供的各种服务。本地可用的 Amazon 服务与在父级 Amazon 区域中运行的服务相比，其延迟响应时间降低。您在 Amazon 区域中使用的 Amazon 服务在父级 Amazon 区域的网络连接断开时不可用。有关在本地提供的 Amazon 服务的信息，请参阅《Amazon Outposts 用户指南》。
- 您可以使用 Kubernetes 服务和入口的普通方法通过您的本地网络连接到在 Amazon Outposts 上的 Amazon EKS 上运行的应用程序。应用程序负载均衡器 (ALB, Application Load Balancer) 可用于 Outposts 机架上的 Kubernetes 入口。网络负载均衡器在 Amazon Outposts 上不可用。对于每个 Kubernetes 服务，节省 Amazon Outposts 上容量的常见做法是将单个 ALB 部署与基于路径的路由结合使用。有关更多信息，请参阅应用程序负载均衡器文档。
- Amazon EBS 卷不会跨越物理或逻辑 Amazon Outposts，类似于 Amazon 区域中的多可用区行为。例如，如果在物理 Outpost A 上运行的节点移动到物理 Outpost B，EBS 卷将不会随之移动。如果您使用 EBS 卷支持的容器组（pod）在 Amazon EKS 上运行有状态的工作负载，请考虑在应用程序层实施双写入操作，或者如果您的应用程序及其数据在单机架故障期间必须保持可用，则使用替代存储机制。

在 Amazon Outposts 上部署包含 Worker 节点的 Amazon EKS 集群

本部分介绍如何创建 Amazon EKS 集群及使用 `eksctl` 和 Amazon CLI 在 Amazon Outposts 在上部署 Amazon EKS Worker 节点。您必须具有在 Amazon Outposts 上创建和管理子网的权限。点击此处

了解可共享 Outpost 资源的更多信息 https://docs.amazonaws.cn/outposts/latest/userguide/sharing-outposts.html#sharing-resources。

1. 使用 YAML 文件创建公有 Amazon EKS 集群：

```
eksctl create cluster -f cluster-config.yaml
```

以下是 YAML 文件的示例：

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-outposts-cluster
  version: 1.21
  region: region-code

cloudWatch:
  clusterLogging:
    enableTypes: [ "api", "audit", "authenticator", "controllerManager", "scheduler" ]

iam:
  withOIDC: true

addons:
  - name: vpc-cni
  - name: coredns
  - name: kube-proxy
```

要创建完全私有的集群，请将以下内容添加到您的配置 YAML 文件中：

```
privateCluster:
  enabled: true
```

输出中的最后一行应如下所示：

```
EKS cluster "my-outposts-cluster" in "region-code" region is ready
```

您可以确认您的集群已在 Amazon EKS Amazon Web Services Management Console 中或使用命令创建：

```
eksctl get cluster --name my-outposts-cluster
```

2. 确定使用新集群创建的 VPC。此 VPC 将托管包含您的 Worker 节点的子网。您可以使用命令找到您的 `vpc-id`：

```
eksctl get cluster --name my-outposts-cluster
```

3. 确定 Outpost 的子网和 CIDR。CIDR 不应与本地网络上使用的其他 IP 地址发生冲突。此操作可以通过使用子网计算器并确定适合您的环境的适当设置来完成。计算 CIDR 后，您可以在下一步中使用此值。您的 `availability-zone` 在 Amazon Web Services Management Console 中可见。
4. 使用 Amazon CLI 在 Amazon Outposts 上创建子网以托管 Worker 节点。在以下命令中使用您在上一步中检索到的 `vpc-id`：

```
aws ec2 create-subnet \
    --region region-code \
    --availability-zone region-code \
```

```
    --outpost-arn my-outpost-arn \
    --vpc-id my-vpc-id \
    --cidr-block my-cidr-block
```

请注意创建后在消息中显示的子网 ID。输出中的行应如下所示：

您的输出应如下所示：

```
"SubnetId": "subnet-1234567890abcdef0",
```

您可以在 Amazon Web Services Management Console 中的集群详细信息下验证子网创建。

5. 使用 YAML 文件创建 Worker 节点。将 `vpc-id` 替换为步骤 1 中创建的值和步骤 2 中创建的 `subnet-id`。`security-group-id` 在步骤 1 中创建，可以使用命令检索：

```
eksctl get cluster --name my-outposts-cluster
```

可以通过使用以下命令使用 YAML 文件创建节点：

```
eksctl create nodegroup -f worker-nodes-config.yaml
```

以下是要包含在此步骤中的示例 YAML 文件。对于 `instanceType`，选择以下实例大小：

- 在您的 Outpost 上设置插槽
- 适用于您的工作负载

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-outposts-cluster
  region: region-code

nodeGroups:
  - name: outpost-worker-nodes
    instanceType: m5.large
    desiredCapacity: 1
    minSize: 1
    maxSize: 1
    volumeSize: 50
    volumeType: gp2
    volumeEncrypted: true
    subnets:
      - subnet-042e2c531a5713a5b
    privateNetworking: true
```

要创建私有 Worker 节点，请将其添加到 YAML 文件的末尾：

```
subnets:
      - outpost-subnet
    volumeType: gp2
    privateNetworking: true
```

输出中的最后一行应如下所示：

```
created 1 nodegroup(s) in cluster "my-outposts-cluster"
```

```
created 0 managed nodegroup(s) in cluster "my-outposts-cluster"
checking security group configuration for all nodegroups
```

6. （可选）要通过本地网络公开 Worker 节点，请参阅本地网关的工作原理。

现在，您已有使用 Amazon Outposts 上运行的 Worker 节点配置了 Amazon EKS 集群，那么就可以安装附加组件并将应用程序部署到您的集群。有关如何扩展集群功能的更多信息，请参阅以下内容：

- 创建集群的 IAM 实体（用户或角色）是唯一可以使用 `kubectl` 调用 API 服务器的 IAM 用户。要向其他用户或角色授予访问权限，请参阅 将 `aws-authConfigMap` 应用到集群 (p. 361)。
- 至您的集群的 部署示例应用程序 (p. 315)。
- 了解重要的 集群管理 (p. 374) 工具。
- 在 Amazon Outposts 上部署容器化应用程序。
- 使用 Terraform 示例在 Amazon Outposts 上构建现代应用程序。

存储

本章介绍 Amazon EKS 集群的存储选项。

存储类 (p. 194) 主题使用树内 Amazon EBS 存储预配置程序。

> Note
>
> 现有树内 Amazon EBS 插件仍受支持，但通过使用 CSI 驱动程序，您可以从 Kubernetes 上游发行周期与 CSI 驱动程序发行周期的分离中获益。最终将弃用树内插件，改为使用 CSI 驱动程序。但是，Fargate 不支持 CSI 驱动程序。

主题

- 存储类 (p. 194)
- Amazon EBS CSI 驱动程序 (p. 195)
- Amazon EFS CSI 驱动程序 (p. 210)
- Amazon FSx for Lustre CSI 驱动程序 (p. 220)
- Amazon FSx for NetApp ONTAP CSI 驱动程序 (p. 225)

存储类

在 Kubernetes 版本 `1.11` 之前创建的 Amazon EKS 集群在创建时不会包含任何存储类。您必须为要使用的集群定义存储类，并且应为持久卷声明定义默认存储类。有关更多信息，请参阅 Kubernetes 文档中的存储类。

> Note
>
> 本主题使用树内 Amazon EBS 存储预配置程序。现有树内 Amazon EBS 插件仍受支持，但通过使用 CSI 驱动程序，您可以从 Kubernetes 上游发行周期与 CSI 驱动程序发行周期的分离中获益。最终将弃用树内插件，改为使用 CSI 驱动程序。但是，Fargate 不支持 CSI 驱动程序。

为 Amazon EKS 集群创建Amazon存储类

1. 确定您的集群已有哪些存储类别。

```
kubectl get storageclass
```

输出示例如下。

```
NAME            PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE
 ALLOWVOLUMEEXPANSION   AGE
gp2 (default)   kubernetes.io/aws-ebs   Delete          WaitForFirstConsumer   false
               34m
```

如果您的集群返回上面的输出，则说明它已在其余步骤中定义了存储类。您可以使用 存储 (p. 194) 一章中用于部署任何 CSI 驱动程序的步骤，来定义其他存储类。部署后，您可以将其中一个存储类设置为默认 (p. 195)存储类。

2. 为存储类创建Amazon存储类清单文件。下列 `gp2-storage-class.yaml` 示例定义一个名为 `gp2` 的存储类，该存储类使用 Amazon EBS `gp2` 卷类型。

有关 Amazon 存储类的可用选项的更多信息，请参阅 Kubernetes 文档中的 Amazon EBS。

```
kind: StorageClass
```

```
apiVersion: storage.k8s.io/v1
metadata:
  name: gp2
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
  fsType: ext4
```

3. 使用 kubectl 从清单文件创建存储类。

```
kubectl create -f gp2-storage-class.yaml
```

输出示例如下。

```
storageclass "gp2" created
```

定义默认存储类

1. 列出集群的现有存储类。必须先定义存储类，然后才能将它设置为默认存储类。

```
kubectl get storageclass
```

输出示例如下。

```
NAME      PROVISIONER             AGE
gp2       kubernetes.io/aws-ebs   8m
```

2. 选择一个存储类并通过设置 storageclass.kubernetes.io/is-default-class=true 注释来将该存储类设置为默认存储类。

```
kubectl annotate storageclass gp2 storageclass.kubernetes.io/is-default-class=true
```

输出示例如下。

```
storageclass "gp2" patched
```

3. 确认存储类现已设置为默认存储类。

```
kubectl get storageclass
```

输出示例如下。

```
gp2 (default)   kubernetes.io/aws-ebs   12m
```

Amazon EBS CSI 驱动程序

Amazon Elastic Block Store (Amazon EBS) Container Storage Interface (CSI) 驱动程序允许 Amazon Elastic Kubernetes Service (Amazon EKS) 集群管理持久性卷的 Amazon EBS 卷的生命周期。

以下是使用 Amazon EBS CSI 驱动程序时要考虑的一些事项。

- Amazon EBS CSI 插件需要 IAM 权限才能代表您调用 Amazon API。有关更多信息，请参阅为服务账户创建 Amazon EBS CSI 驱动程序 IAM 角色 (p. 196)。
- Amazon EKS 集群不支持 Amazon EBS CSI 驱动程序的 Alpha 功能。
- Important

  如果您当前运行 pods 的 `1.22` 或更早版本的集群使用 Amazon EBS 卷，并且您的集群上当前没有安装此驱动程序，请务必在将集群更新到 `1.23` 之前将此驱动程序安装到您的集群中。如果未在将集群更新到 `1.23` 之前安装此驱动程序，则可能会中断您的工作负载。有关更多信息或要获取有关此要求的常见问题解答，请参阅 the section called “Kubernetes 1.23” (p. 59) 或 the section called “CSI 迁移常见问题” (p. 208)。

首次创建集群时，不安装 Amazon EBS CSI 驱动程序。要使用该驱动程序，您必须将其添加为 Amazon EKS 附加组件或自行管理的附加组件。

- 有关如何将其添加为 Amazon EKS 附加组件的说明，请参阅 将 Amazon EBS CSI 驱动程序作为 Amazon EKS 附加组件管理 (p. 201)。
- 有关如何将其添加为自行管理的附加组件的说明，请参阅 GitHub 上的 Amazon EBS 容器存储接口 (CSI) 驱动程序项目。

使用任何一种方法安装 CSI 驱动程序后，都可以使用示例应用程序测试功能。有关更多信息，请参阅部署示例应用程序并验证 CSI 驱动程序是否正常运行 (p. 206)。

为服务账户创建 Amazon EBS CSI 驱动程序 IAM 角色

Amazon EBS CSI 插件需要 IAM 权限才能代表您调用 Amazon API。有关更多信息，请参阅 GitHub 上的设置驱动程序权限。

部署插件后，创建并配置来使用名为 `ebs-csi-controller-sa` 的服务账户。服务账户绑定到被分配了所需的 Kubernetes 权限的 Kubernetes `clusterrole`。

Note

无论您是否将 Amazon EBS CSI 插件配置为将 IAM 角色用于服务账户，pods 都可以访问分配到 IAM 角色的权限。这种情况下，除非您阻止对IMD的访问。有关更多信息，请参阅Amazon EKS 的安全最佳实践 (p. 453)。

先决条件

- 现有 `1.18` 版或更高版本的集群。
  - 1.18 要求使用 eks.9 或更高版本。
  - 1.19 要求使用 eks.7 或更高版本。
  - 1.20 要求使用 eks.3 或更高版本。
  - 1.21 要求使用 eks.3 或更高版本。
- 集群现有 Amazon Identity and Access Management (IAM) OpenID Connect (OIDC) 提供程序。要确定您是否已经拥有一个或是否要创建一个，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。

创建 IAM 角色并向其附加所需的 Amazon 托管策略。您可以使用 `eksctl`、Amazon Web Services Management Console或 Amazon CLI。

eksctl

使用 **`eksctl`** 创建 Amazon EBS CSI 插件 IAM 角色

1. 使用以下命令创建 IAM 角色并附加所需的 Amazon 托管策略。将 *`my-cluster`* 替换为您的集群名称。命令会部署一个创建了 IAM 角色的 Amazon CloudFormation 堆栈，向其附加 IAM policy，并

使用 IAM 角色的 Amazon 资源名称（ARN）对现有 ebs-csi-controller-sa 服务账户添加注释。

```
eksctl create iamserviceaccount \
  --name ebs-csi-controller-sa \
  --namespace kube-system \
  --cluster my-cluster \
  --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
 \
  --approve \
  --role-only \
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

2. 如果在 Amazon EBS 卷上使用自定义 KMS 密钥进行加密，请根据需要自定义 IAM 角色。例如，执行以下操作：

   a. 复制并在新的 *kms-key-for-encryption-on-ebs*.json 文件中粘贴以下代码。将 *custom-key-id* 替换为自定义 KMS 密钥 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-id"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-id"]
    }
  ]
}
```

   b. 创建策略。您可以将 *KMS_Key_For_Encryption_On_EBS_Policy* 更改为其他名称。但是，如果更改了名称，请确保在后面的步骤中也进行相应的更改。

```
aws iam create-policy \
  --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
  --policy-document file://kms-key-for-encryption-on-ebs.json
```

   c. 使用以下命令将 IAM policy 附加到该角色。请将 *111122223333* 替换为您的账户 ID。

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/KMS_Key_For_Encryption_On_EBS_Policy \
```

```
    --role-name AmazonEKS_EBS_CSI_DriverRole
```

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 创建 Amazon EBS CSI 插件 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Roles（角色）。
3. 在 Roles（角色）页面上，选择 Create role（创建角色）。
4. 在 Select trusted entity（选择受信任的实体）页面上，请执行以下操作：
   a. 在 Trusted entity type（受信任的实体类型）部分中，选择 Web identity（Web 身份）。
   b. 对于 Identity provider（身份提供商），为您的集群选择 OpenID Connect provider URL（提供商 URL）（如 Amazon EKS 中的 Overview（概述）下所示）。
   c. 对于 Audience (受众)，请选择 `sts.amazonaws.com.cn`。
   d. 选择 Next (下一步)。
5. 在 Add permissions（添加权限）页面上，请执行以下操作：
   a. 在 Filter policies (筛选器策略) 框中，输入 *AmazonEBSCSIDriverPolicy*。
   b. 选中搜索返回的 *AmazonEBSCSIDriverPolicy* 左侧的复选框。
   c. 选择 Next (下一步)。
6. 在 Name, review, and create（命名、查看和创建）页面中，请执行以下操作：
   a. 对于 Role name（角色名称），请为角色输入唯一名称，例如 ***AmazonEKS_EBS_CSI_DriverRole***。
   b. 在 Add tags (Optional)（添加标签(可选)）下，通过以键值对的形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 IAM 用户指南中的标记 IAM 实体。
   c. 选择 Create role（创建角色）。
7. 创建角色后，在控制台中选择角色以将其打开进行编辑。
8. 选择 Trust relationships（信任关系）选项卡，然后选择 Edit trust policy（编辑信任策略）。
9. 该行看起来类似于以下行：

   ```
   "oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
    "sts.amazonaws.com.cn"
   ```

   在上一行末尾添加逗号，然后在前一行后面添加以下行。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 替换为集群的 OIDC 提供商 ID。

   ```
   "oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
    "system:serviceaccount:kube-system:ebs-csi-controller-sa"
   ```

10. 选择 Update policy（更新策略）以完成操作。
11. 如果在 Amazon EBS 卷上使用自定义 KMS 密钥进行加密，请根据需要自定义 IAM 角色。例如，执行以下操作：
    a. 在左侧导航窗格中，选择 Policies (策略)。
    b. 在策略页面上，选择 Create a policy (创建策略)。
    c. 在创建策略页面上，选择 JSON 选项卡。
    d. 将以下代码复制并粘贴到编辑器中，从而将 *custom-key-id* 替换为自定义 KMS 密钥 ID：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-id"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-id"]
    }
  ]
}
```

e. 请选择下一步：标签。

f. 在 Add tags (Optional)（添加标签（可选））页面上，选择 Next: Review（下一步：审核）。

g. 对于 Name（名称），请为您的策略输入唯一的名称（例如 ***KMS_Key_For_Encryption_On_EBS_Policy***）。

h. 选择 Create policy（创建策略）。

i. 在左侧导航窗格中，选择 Roles（角色）。

j. 在控制台中选择 ***AmazonEKS_EBS_CSI_DriverRole*** 以打开它进行编辑。

k. 从 Add permissions（添加权限）下拉列表中，选择 Attach policies（附加策略）。

l. 在 Filter policies (筛选器策略) 框中，输入 *KMS_Key_For_Encryption_On_EBS_Policy*。

m. 选中搜索中返回的 *KMS_Key_For_Encryption_On_EBS_Policy* 左侧的复选框。

n. 选择 Attach policies（附上策略）。

Amazon CLI

使用 Amazon CLI 创建 Amazon EBS CSI 插件 IAM 角色

1. 查看集群的 OIDC 提供商 URL。将 *my-cluster* 替换为您的集群名称。如果命令的输出为 `None`，请查看先决条件。

```
aws eks describe-cluster \
  --name my-cluster \
  --query "cluster.identity.oidc.issuer" \
  --output text
```

输出示例如下。

```
https://oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. 创建 IAM 角色。

   a. 将以下内容复制到名为 *aws-ebs-csi-driver-trust-policy*.json 的文件中。请将 *111122223333* 替换为您的账户 ID，将 *region-code* 替换为您的 Amazon Web Services 区域，并将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 替换为上一步骤中返回的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com.cn/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
          "oidc.eks.region-code.amazonaws.com.cn/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:ebs-csi-controller-sa"
        }
      }
    }
  ]
}
```

   b. 创建角色。您可以将 *AmazonEKS_EBS_CSI_DriverRole* 更改为其他名称。如果更改了名称，请确保在后面的步骤中也进行相应的更改。

```
aws iam create-role \
  --role-name AmazonEKS_EBS_CSI_DriverRole \
  --assume-role-policy-document file://"aws-ebs-csi-driver-trust-policy.json"
```

3. 使用以下命令以将所需的 Amazon 托管策略附加到角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

4. 如果在 Amazon EBS 卷上使用自定义 KMS 密钥进行加密，请根据需要自定义 IAM 角色。例如，执行以下操作：

   a. 复制并在新的 *kms-key-for-encryption-on-ebs*.json 文件中粘贴以下代码。将 *custom-key-id* 替换为自定义 KMS 密钥 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-id"],
```

```
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-id"]
    }
  ]
}
```

b. 创建策略。您可以将 *KMS_Key_For_Encryption_On_EBS_Policy* 更改为其他名称。但是，如果更改了名称，请确保在后面的步骤中也进行相应的更改。

```
aws iam create-policy \
  --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
  --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. 使用以下命令将 IAM policy 附加到该角色。请将 *111122223333* 替换为您的账户 ID。

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/KMS_Key_For_Encryption_On_EBS_Policy \
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

将 Amazon EBS CSI 驱动程序作为 Amazon EKS 附加组件管理

要提高安全性并减少工作量，您可以将 Amazon EBS CSI 驱动程序作为 Amazon EKS 附加组件管理。有关 Amazon EKS 附加组件的信息，请参阅 Amazon EKS 附加组件 (p. 341)。您可以按照 添加 Amazon EBS CSI 附加组件 (p. 202) 中的步骤添加 Amazon EBS CSI 附加组件。

如果添加了 Amazon EBS CSI 附加组件，则可以按照 将 Amazon EBS CSI 驱动程序作为 Amazon EKS 附加组件更新 (p. 203) 和 删除 Amazon EBS CSI 附加组件 (p. 205) 部分的步骤进行管理。

先决条件

- 现有 `1.18` 版或更高版本的集群。要查看所需的平台版本，请运行以下命令。

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- 集群的现有 Amazon Identity and Access Management (IAM) OpenID Connect (OIDC) 提供程序。要确定您是否已经拥有一个或是否要创建一个，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。
- Amazon EBS CSI 驱动程序 IAM 角色。有关更多信息，请参阅为服务账户创建 Amazon EBS CSI 驱动程序 IAM 角色 (p. 196)。

Important

要使用 Amazon EBS CSI 驱动程序的快照功能，必须在安装附加组件之前安装外部快照程序。必须按以下顺序安装外部快照程序组件：

- 适用于 `volumesnapshotclasses`、`volumesnapshots` 和 `volumesnapshotcontents` 的 `CustomResourceDefinition`（CRD）
- RBAC（`ClusterRole`、`ClusterRoleBinding` 等）
- 控制器部署

有关更多信息，请参阅 GitHub 上的 CSI 快照程序。

添加 Amazon EBS CSI 附加组件

Important

在添加 Amazon EBS CSI 附加组件之前，请确认您不会自行管理Amazon EKS 将开始管理的任何设置。要确定 Amazon EKS 管理的设置，请参阅 Amazon EKS 附加组件配置 (p. 342)。

您可以使用 `eksctl`、Amazon Web Services Management Console 或 Amazon CLI 将 Amazon EBS CSI 附加组件添加到您的集群。

eksctl

要使用 **`eksctl`** 添加 Amazon EBS CSI 附加组件

运行以下 命令。请将 *`my-cluster`* 替换为集群名称，将 *`111122223333`* 替换为账户 ID，并将 *`AmazonEKS_EBS_CSI_DriverRole`* 替换为之前创建的角色名称。

```
eksctl create addon --name aws-ebs-csi-driver --cluster my-cluster --service-account-
role-arn arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole --force
```

如果删除 `--force` 选项并且与现有设置存在冲突，则命令将失败。您可以使用生成的错误消息对冲突进行故障排除。选择此选项之前，请确保 Amazon EKS 附加组件不会管理您需要自行管理的设置。这是因为该选项覆盖了这些设置。有关管理 Amazon EKS 附加组件的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

Amazon Web Services Management Console

要使用 Amazon Web Services Management Console 添加 Amazon EBS CSI 附加组件

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择集群。
3. 选择要为其配置 Amazon EBS CSI 附加组件的集群名称。
4. 选择 Add-ons（附加组件）选项卡。
5. 选择 Add new (新增)。
    a. 为 Name（名称）选择 Amazon EBS CSI Driver（Amazon EBS CSI 驱动程序）。
    b. 选择您想使用的 Version（版本）。
    c. 对于 Service account role（服务账户角色），选择要将 IAM policy 附加到的 IAM 角色。
    d. 如果选择 Override existing configuration for this add-on on the cluster（覆盖集群上此附加组件的现有配置），则可能用 Amazon EKS 附加组件设置覆盖现有附加组件的一个或多个设置。如果不启用此选项，并且与现有设置存在冲突，则操作将失败。您可以使用生成的错误消息对冲突进行故障排除。在选择此选项之前，请确保 Amazon EKS 附加组件不会管理您需要自行管理的设置。有关管理 Amazon EKS 附加组件的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。
    e. 选择 Add（添加）。

Amazon CLI

要使用 Amazon CLI 添加 Amazon EBS CSI 附加组件

运行以下 命令。将 *my-cluster* 替换为集群名称，将 *111122223333* 替换为账户 ID，将 *AmazonEKS_EBS_CSI_DriverRole* 替换为之前创建的角色名称。

```
aws eks create-addon \
  --cluster-name my-cluster \
  --addon-name aws-ebs-csi-driver \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole
```

将 Amazon EBS CSI 驱动程序作为 Amazon EKS 附加组件更新

新版本发布后或者将集群更新到 (p. 28)一个新的 Kubernetes 次要版本后，Amazon EKS 不会自动为集群更新 Amazon EBS CSI。要更新现有集群上的 Amazon EBS CSI，您必须启动更新，然后 Amazon EKS 会为您更新该附加组件。

Important

请先将您的集群和节点更新到新的 Kubernetes 次要版本，然后再将 Amazon EBS CSI 更新到相同的次要版本。

eksctl

要使用 **`eksctl`** 更新 Amazon EBS CSI 附加组件

1. 检查 Amazon EBS CSI 附加组件的当前版本。将 *my-cluster* 替换为您的集群名称。

   ```
   eksctl get addon --name aws-ebs-csi-driver --cluster my-cluster
   ```

   输出示例如下。

   ```
   NAME                    VERSION                     STATUS  ISSUES  IAMROLE UPDATE
    AVAILABLE
   aws-ebs-csi-driver      v1.4.0-eksbuild.preview     ACTIVE  0
     v1.5.1-eksbuild.1
   ```

2. 将附加组件更新为上一个步骤的输出中的 `UPDATE AVAILABLE` 下返回的版本。

   ```
   eksctl update addon \
     --name aws-ebs-csi-driver \
     --version v1.5.1-eksbuild.1 \
     --cluster my-cluster \
     --force
   ```

   如果删除 `--force` 选项并且与现有设置存在冲突，则命令将失败。您可以使用生成的错误消息对冲突进行故障排除。选择此选项之前，请确保 Amazon EKS 附加组件不会管理您需要自行管理的设置。这是因为该选项覆盖了这些设置。有关管理 Amazon EKS 附加组件的更多信息，请参阅Amazon EKS 附加组件配置 (p. 342)。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 更新 Amazon EBS CSI 附加组件

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。

2. 在左侧导航窗格中，选择集群。
3. 选择要为其更新 Amazon EBS CSI 附加组件的集群名称。
4. 选择 Add-ons（附加组件）选项卡。
5. 选择 aws-ebs-csi-driver 框右上角的单选按钮。
6. 选择 Edit（编辑）。
   a. 选择您要使用的 Amazon EKS 附加组件的 Version（版本）。
   b. 对于 Service account role（服务账户角色），选择要将 Amazon EBS CSI 驱动程序 IAM policy 附加到的 IAM 角色的名称。
   c. 如果选择 Override existing configuration for this add-on on the cluster（覆盖集群上此附加组件的现有配置），则可能用 Amazon EKS 附加组件设置覆盖现有附加组件的一个或多个设置。如果不启用此选项，并且与现有设置存在冲突，则操作将失败，并显示一条错误消息来帮助您解决冲突。在选择此选项之前，请确保 Amazon EKS 附加组件不会管理您需要自行管理的设置。有关管理 Amazon EKS 附加组件的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。
   d. Select Update（更新）。

Amazon CLI

使用 Amazon CLI 更新 Amazon EBS CSI 附加组件

1. 检查 Amazon EBS CSI 附加组件的当前版本。将 *my-cluster* 替换为您的集群名称。

```
aws eks describe-addon \
  --cluster-name my-cluster \
  --addon-name aws-ebs-csi-driver \
  --query "addon.addonVersion" \
  --output text
```

输出示例如下。

```
v1.4.0-eksbuild.preview
```

2. 确定哪些版本的 Amazon EBS CSI 附加组件可用于您的集群版本。

```
aws eks describe-addon-versions \
  --addon-name aws-ebs-csi-driver \
  --kubernetes-version 1.20 \
  --query "addons[].addonVersions[].[addonVersion,
 compatibilities[].defaultVersion]" \
  --output text
```

输出示例如下。

```
v1.5.1-eksbuild.1
False
v1.4.0-eksbuild.preview
True
```

下面带有 `True` 的版本是使用您指定的版本的新集群部署的默认版本。

3. 将附加组件更新为上一步输出中返回 `True` 的版本。如果在输出中返回，您也可以更新到更高版本。

```
aws eks update-addon \
```

```
--cluster-name my-cluster \
--addon-name aws-ebs-csi-driver \
--addon-version v1.4.0-eksbuild.preview \
--resolve-conflicts OVERWRITE
```

如果删除 `--resolve-conflicts OVERWRITE` 选项并且与现有设置存在冲突，则命令将失败，并显示错误消息来帮助您解决冲突。选择此选项之前，请确保 Amazon EKS 附加组件不会管理您需要自行管理的设置。这是因为该选项覆盖了这些设置。有关管理 Amazon EKS 附加组件的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

删除 Amazon EBS CSI 附加组件

移除 Amazon EKS 附加组件时，您有两种选择。

- 将附加组件保留在您的集群上 - 此方法将移除 Amazon EKS 对任何设置的管理。还移除 Amazon EKS 通知您更新以及在您启动更新后自动更新 Amazon EKS 附加组件的功能。但是，此方法会保留集群上的附加组件软件。此选项使附加组件成为自我管理的附加组件，而不是 Amazon EKS 附加组件。使用此方法，附加组件不停机。此过程中的命令使用此选项。
- 从集群中完全删除附加组件 - 我们建议只有当集群中没有资源依赖于附加组件时，才从集群移除 Amazon EKS 附加组件。要执行此选项，请从您在此过程中使用的命令中删除 `--preserve`。

如果附加组件有与其关联的 IAM 账户，则不会移除该 IAM 账户。

您可以使用 `eksctl`、Amazon Web Services Management Console 或 Amazon CLI 移除 Amazon EBS CSI 附加组件。

eksctl

要使用 **`eksctl`** 移除 Amazon EBS CSI 附加组件

请将 *my-cluster* 替换为您的集群名称，然后运行以下命令。

```
eksctl delete addon --cluster my-cluster --name aws-ebs-csi-driver --preserve
```

Amazon Web Services Management Console

要使用 Amazon Web Services Management Console 移除 Amazon EBS CSI 附加组件

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择集群。
3. 选择要为其移除 Amazon EBS CSI 附加组件的集群名称。
4. 选择 Add-ons（附加组件）选项卡。
5. 选择 aws-ebs-csi-driver 框右上角的单选按钮。
6. 选择 Remove。
7. 如果希望 Amazon EKS 停止管理附加组件设置，请选择 Preserve on cluster（在集群上保留）。如果要在集群上保留附加组件软件，请执行此操作。这样您可以自行管理附加组件的所有设置。
8. 输入 **`aws-ebs-csi-driver`**。
9. 选择 Remove（移除）。

Amazon CLI

要使用 Amazon CLI 移除 Amazon EBS CSI 附加组件

请将 *my-cluster* 替换为您的集群名称，然后运行以下命令。

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-ebs-csi-driver --
preserve
```

部署示例应用程序并验证 CSI 驱动程序是否正常运行

您可以使用示例应用程序测试 CSI 驱动程序功能。本主题展示了一个示例，但您还可以执行以下操作：

- 部署使用外部快照创建卷快照的示例应用程序。有关更多信息，请参阅 GitHub 上的卷快照。
- 部署使用卷大小的示例应用程序。有关更多信息，请参阅 GitHub 上的调整卷大小。

此过程利用来自 Amazon EBS 容器存储接口 (CSI) 驱动程序 GitHub 存储库的动态卷配置示例，使用动态预置的 Amazon EBS 卷。

1. 将 Amazon EBS 容器存储接口 (CSI) 驱动程序 GitHub 存储库克隆到您的本地系统。

   ```
   git clone https://github.com/kubernetes-sigs/aws-ebs-csi-driver/blob/master/deploy/
   kubernetes/overlays/stable/ecr/kustomization.yaml
   ```

2. 导航到 dynamic-provisioning 示例目录。

   ```
   cd aws-ebs-csi-driver/examples/kubernetes/dynamic-provisioning/
   ```

3. 从 manifests 目录部署 ebs-sc 存储类、ebs-claim 持久性卷声明和 app 示例应用程序。

   ```
   kubectl apply -f manifests/
   ```

4. 描述 ebs-sc 存储类。

   ```
   kubectl describe storageclass ebs-sc
   ```

   输出示例如下。

   ```
   Name:            ebs-sc
   IsDefaultClass:  No
   Annotations:     kubectl.kubernetes.io/last-applied-
   configuration={"apiVersion":"storage.k8s.io/v1","kind":"StorageClass","metadata":
   {"annotations":{},"name":"ebs-
   sc"},"provisioner":"ebs.csi.aws.com","volumeBindingMode":"WaitForFirstConsumer"}

   Provisioner:           ebs.csi.aws.com
   Parameters:            <none>
   AllowVolumeExpansion:  <unset>
   MountOptions:          <none>
   ReclaimPolicy:         Delete
   VolumeBindingMode:     WaitForFirstConsumer
   Events:                <none>
   ```

   Note

   存储类使用 WaitForFirstConsumer 卷绑定模式。这意味着，在 pod 进行持久性卷声明之前，不会动态预置卷。有关更多信息，请参阅 Kubernetes 文档中的卷绑定模式。

5. 查看默认命名空间中的 pods。几分钟后，app pod 的状态变为 Running。

   ```
   kubectl get pods --watch
   ```

输入 Ctrl+C 以返回到 Shell 提示符。

6. 列出默认命名空间中的持久性卷。查找具有 default/ebs-claim 声明的持久性卷。

```
kubectl get pv
```

输出示例如下。

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
 STATUS   CLAIM                 STORAGECLASS   REASON   AGE
pvc-37717cd6-d0dc-11e9-b17f-06fad4858a5a   4Gi        RWO            Delete
 Bound    default/ebs-claim   ebs-sc                  30s
```

7. 描述持久性卷。将 pvc-*37717cd6-d0dc-11e9-b17f-06fad4858a5a* 替换为上一步的输出值。

```
kubectl describe pv pvc-37717cd6-d0dc-11e9-b17f-06fad4858a5a
```

输出示例如下。

```
Name:              pvc-37717cd6-d0dc-11e9-b17f-06fad4858a5a
Labels:            <none>
Annotations:       pv.kubernetes.io/provisioned-by: ebs.csi.aws.com
Finalizers:        [kubernetes.io/pv-protection external-attacher/ebs-csi-aws-com]
StorageClass:      ebs-sc
Status:            Bound
Claim:             default/ebs-claim
Reclaim Policy:    Delete
Access Modes:      RWO
VolumeMode:        Filesystem
Capacity:          4Gi
Node Affinity:
  Required Terms:
    Term 0:        topology.ebs.csi.aws.com/zone in [region-code]
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            ebs.csi.aws.com
    VolumeHandle:      vol-0d651e157c6d93445
    ReadOnly:          false
    VolumeAttributes:      storage.kubernetes.io/
csiProvisionerIdentity=1567792483192-8081-ebs.csi.aws.com
Events:                <none>
```

Amazon EBS 卷 ID 是之前输出中 VolumeHandle 的值。

8. 验证 pod 是否将数据写入卷。

```
kubectl exec -it app -- cat /data/out.txt
```

输出示例如下。

```
Wed May 5 16:17:03 UTC 2021
Wed May 5 16:17:08 UTC 2021
Wed May 5 16:17:13 UTC 2021
Wed May 5 16:17:18 UTC 2021
...
```

9. 完成后，请删除此示例应用程序的资源。

```
kubectl delete -f manifests/
```

Amazon EBS CSI 迁移常见问题

默认情况下，Amazon EBS Container Storage Interface (CSI) 迁移功能在 Amazon EKS `1.23` 和更高版本的集群上处于启用状态。如果您在版本 **`1.22`** 或更早版本的集群上运行 pods，必须先安装 Amazon EBS 驱动程序 (p. 195)，然后再将集群更新到版本 **`1.23`**，以避免服务中断。选择任何常见问题以了解其答案。

什么是 CSI 驱动程序？

CSI 驱动程序具有以下功能：

- 替换存在于 Kubernetes 项目源代码中的 Kubernetes“树内”存储驱动程序。
- 与存储提供商（如 Amazon EBS）合作。
- 提供简化的插件模型，使 Amazon 这样的存储提供商更容易发布功能并保持支持，而无需依赖 Kubernetes 发布周期。

有关更多信息，请参阅 Kubernetes CSI 文档中的介绍。

什么是 CSI 迁移？

Kubernetes CSI 迁移功能将处理存储操作的责任从现有的树内存储插件（例如 `kubernetes.io/aws-ebs`）移至相应的 CSI 驱动程序。只要安装了相应的 CSI 驱动程序，现有的 `StorageClass`、`PersistentVolume` 和 `PersistentVolumeClaim` (PVC) 对象便会继续工作。启用该功能后：

- 利用 PVC 的现有工作负载将一如既往地继续运行。
- Kubernetes 将所有存储管理操作的控制权交给 CSI 驱动程序。

有关更多信息，请参阅 Kubernetes 博客上的 Kubernetes`1.23`: Kubernetes In-Tree to CSI Volume Migration Status Update（树内插件至 CSI 卷迁移状态更新）。

为帮助您从树内插件迁移到 CSI 驱动程序，默认情况下在 Amazon EKS 版本 `1.23` 以及更高版本的集群上启用 `CSIMigration` 和 `CSIMigrationAWS` 标记。这些标记让您的集群能够将树内 API 转换为其等效的 CSI API。这些标记设置在由 Amazon EKS 管理的 Kubernetes 控制面板，以及在 Amazon EKS 优化版 AMI 中配置的 `kubelet` 设置中。如果您在集群中有使用 Amazon EBS 卷的 pods，请在将集群更新为版本 **`1.23`** 之前，安装 Amazon EBS CSI 驱动程序。否则，预置和挂载等卷操作可能无法按预期工作。有关更多信息，请参阅 Amazon EBS CSI 驱动程序 (p. 195)。

> **Note**
>
> 树内 `StorageClass` 置备程序命名为 `kubernetes.io/aws-ebs`。Amazon EBS CSI `StorageClass` 置备程序命名为 `ebs.csi.aws.com`。

我能否在版本 `1.23` 和更高版本的集群中挂载 `kubernetes.io/aws-ebs`StorageClass 卷？

是的，只要安装了 Amazon EBS CSI 驱动程序 (p. 195)即可执行此操作。对于新创建的版本 `1.23` 和更高版本的集群，我们建议您在集群创建过程中安装 Amazon EBS CSI 驱动程序。我们还建议只使用基于 `ebs.csi.aws.com` 置备程序的 `StorageClasses`。

如果您已将集群控制面板更新为版本 `1.23` 并且尚未将您的节点更新为 `1.23`，则 `CSIMigration` 和 `CSIMigrationAWS` `kubelet` 标记未启用。在这种情况下，使用树内驱动程序来挂载基于 `kubernetes.io/aws-ebs` 的卷。但是，仍必须安装 Amazon EBS CSI 驱动程序，以确保可以计划使用基于 `kubernetes.io/aws-ebs` 的卷的 pods。其他卷操作也需要驱动程序才能成功。

我是否可以在 Amazon EKS 1.23 和更高版本的集群上预置 kubernetes.io/aws-ebsStorageClass 卷？

是的，只要安装了 Amazon EBS CSI 驱动程序 (p. 195)即可执行此操作。

kubernetes.io/aws-ebsStorageClass 置备程序是否会从 Amazon EKS 中删除？

不会。StorageClass 置备程序 kubernetes.io/aws-ebs 和卷类型 awsElasticBlockStore 是 Kubernetes API 的一部分。这些资源不再受支持，但没有删除它们的计划。

如何安装 Amazon EBS CSI 驱动程序？

如果您想自己管理驱动程序，可以使用开源的 Helm 图表进行安装。或者，您可以安装 Amazon EBS CSI 驱动程序 Amazon EKS 附加组件 (p. 195)。如需更新 Amazon EKS 附加组件，您需启动更新，然后 Amazon EKS 会为您更新附加组件。

> Important
>
> Kubernetes 树内 Amazon EBS 驱动程序在 Kubernetes 控制面板上运行。它使用分配给 the section called “集群 IAM 角色” (p. 425) 的 IAM 权限来预置 Amazon EBS 卷。Amazon EBS CSI 驱动程序在节点上运行。该驱动程序需要 IAM 权限才能预置卷。有关更多信息，请参阅the section called “创建 IAM policy 和角色” (p. 196)。

如何检查我的集群中是否安装了 Amazon EBS CSI 驱动程序？

使用 helm list -A 确定是否安装了开源的 Helm 图表。使用 aws eks list-addons 确定是否安装了 Amazon EKS 附加组件。

如果我尚未安装 Amazon EBS CSI 驱动程序，Amazon EKS 是否会阻止集群更新到版本 1.23？

否。

如果我在将集群更新到 1.23 版本之前忘记安装 Amazon EBS CSI 驱动程序，该怎么办？ 我可以在更新集群后安装驱动程序吗？

可以，但需要 Amazon EBS CSI 驱动程序的卷操作将在集群更新后失败，直到安装驱动程序为止。

在新创建的 Amazon EKS 1.23 版本和更高版本集群中应用的默认 StorageClass 是什么？

默认 StorageClass 行为将保持不变。对于每个新集群，Amazon EKS 都会应用一个名为 gp2 的基于 kubernetes.io/aws-ebs 的 StorageClass。我们不打算从新创建的集群中删除该 StorageClass。尽管可能会发生变化，但我们目前以 1.25 版本为目标，以便从应用于新创建集群的 StorageClass 中删除默认注释。与集群默认 StorageClass 独立，如果您在不指定卷类型的情况下创建基于 ebs.csi.aws.com 的 StorageClass，则 Amazon EBS CSI 驱动程序将默认为使用 gp3。

当我将集群更新到 1.23 版本时，Amazon EKS 是否会对已存在于现有集群中的 StorageClasses 进行更改？

否。

如何使用快照将持久卷从 kubernetes.io/aws-ebsStorageClass 迁移到 ebs.csi.aws.com？

要迁移持久卷，请参阅 Amazon 博客上的将 Amazon EKS 集群从 gp2 迁移到 gp3 EBS 卷。

Windows 工作负载是否支持迁移？

是。如果您正在使用开源的 Helm 图表安装 Amazon EBS CSI 驱动程序，请将 `node.enableWindows` 设置为 `true`。如果将 Amazon EBS CSI 驱动程序作为 Amazon EKS 附加组件安装，该设置为默认设置。创建 `StorageClasses` 时，将 `fsType` 设置为 Windows 文件系统，例如 `ntfs`。随即将 Windows 工作负载的卷操作迁移到 Amazon EBS CSI 驱动程序，就像迁移 Linux 工作负载的卷操作一样。一个例外情况是文件系统大小调整操作，这些操作在使用 Windows 文件系统的 `kubernetes.io/aws-ebs` 持久卷上不受支持。

Amazon EFS CSI 驱动程序

Amazon EFS Container Storage Interface（CSI）驱动程序提供了一个 CSI 接口，允许在 Amazon 上运行的 Kubernetes 集群管理 Amazon EFS 文件系统的生命周期。

本主题介绍了如何部署 Amazon EFS CSI 驱动程序到您的 Amazon EKS 集群，并验证它是否正常工作。

> **Note**
>
> Amazon EKS 集群 不支持 Amazon EFS CSI 驱动程序的 Alpha 功能。

有关可用参数的详细说明和演示驱动程序功能的完整示例，请参阅 GitHub 上的 Amazon EFS 容器存储接口 (CSI) 驱动程序项目。

注意事项

- Amazon EFS CSI 驱动程序与基于 Windows 的容器映像不兼容。
- 您不能将动态持久性卷预置与 Fargate 节点结合使用，但可以使用静态预置。
- 动态预置需要 `1.2` 或更高版本的驱动程序。您可以在任何受支持的 Amazon EKS 集群版本 (p. 59)上使用 `1.1` 版本的驱动程序静态预置持久性卷。
- 此驱动程序的 `1.3.2` 版或更高版本支持 Arm64 架构，包括基于 Amazon EC2 Graviton 的实例。

先决条件

- 集群的现有 Amazon Identity and Access Management (IAM) OpenID Connect (OIDC) 提供程序。要确定您是否已经拥有一个或是否要创建一个，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。
- 您的设备或 Amazon CloudShell 上安装并配置了 `2.7.21` 版或更高版本，或 `1.25.46` 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 `aws configure` 进行快速配置。
- 您的设备或 Amazon CloudShell 上安装了 `kubectl` 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 `1.22`，则可以将 `kubectl` 版本 `1.21`、`1.22` 或 `1.23` 用于它。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl` (p. 4)。

> **Note**
>
> Amazon Fargate 上运行的 pod 会自动挂载 Amazon EFS 文件系统，无需此页面上所述的手动驱动程序安装步骤。

创建 IAM policy和角色

创建 IAM policy 并将其分配给 IAM 角色。该策略将允许 Amazon EFS 驱动程序与您的文件系统交互。

将 Amazon EFS CSI 驱动程序部署到 Amazon EKS 集群

1. 创建允许 CSI 驱动程序的服务账户代表您调用 Amazon API 的 IAM policy。

   a. 从 GitHub 下载 IAM policy 文档。您还可以查看策略文档。

```
curl -o iam-policy-example.json https://raw.githubusercontent.com/kubernetes-sigs/
aws-efs-csi-driver/master/docs/iam-policy-example.json
```

b. 创建策略。您可以将 *AmazonEKS_EFS_CSI_Driver_Policy* 更改为其他名称，但如果更改，请确保在后续步骤中也做出相应更改。

```
aws iam create-policy \
    --policy-name AmazonEKS_EFS_CSI_Driver_Policy \
    --policy-document file://iam-policy-example.json
```

2. 创建 IAM 角色并向其附加此 IAM policy。用 IAM 角色 ARN 对 Kubernetes 服务账户进行注释，用 Kubernetes 服务账户名称对 IAM 角色进行注释。您可以使用 `eksctl` 或 Amazon CLI 创建角色。

   eksctl

   运行下面的命令，以创建 IAM 角色和 Kubernetes 服务账户。它还将策略附加到角色，用 IAM 角色 ARN 对 Kubernetes 服务账户添加注释，并将 Kubernetes 服务账户名称添加到 IAM 角色的信任策略中。请将 *my-cluster* 替换为您的集群名称，并将 *111122223333* 替换为您的账户 ID。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

```
eksctl create iamserviceaccount \
    --cluster my-cluster \
    --namespace kube-system \
    --name efs-csi-controller-sa \
    --attach-policy-arn
 arn:aws:iam::111122223333:policy/AmazonEKS_EFS_CSI_Driver_Policy \
    --approve \
    --region region-code
```

   Amazon CLI

   1. 确定集群的 OIDC 提供商 URL。将 *my-cluster* 替换为您的集群名称。如果命令的输出为 `None`，请查看先决条件。

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer"
 --output text
```

      输出示例如下。

```
https://oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

   2. 创建 IAM 角色，从而向 Kubernetes 服务账户授予 `AssumeRoleWithWebIdentity` 操作权限。
      a. 将以下内容复制到名为 *trust-policy*.json 的文件中。请将 *111122223333* 替换为您的账户 ID。将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 和 *region-code* 替换为在上一步中返回的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
```

```
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com.cn/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:efs-csi-controller-sa"
        }
      }
    }
  ]
}
```

b. 创建角色。您可以将 *AmazonEKS_EFS_CSI_DriverRole* 更改为其他名称，但如果更改，请确保在后续步骤中也做出相应更改。

```
aws iam create-role \
  --role-name AmazonEKS_EFS_CSI_DriverRole \
  --assume-role-policy-document file://"trust-policy.json"
```

3. 使用以下命令将 IAM policy 附加到该角色。请将 *111122223333* 替换为您的账户 ID。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_EFS_CSI_Driver_Policy \
  --role-name AmazonEKS_EFS_CSI_DriverRole
```

4. 创建一个使用您创建的 IAM 角色的 ARN 进行注释的 Kubernetes 服务账户。

a. 将以下内容保存到名为 *efs-service-account*.yaml 的文件中。请将 *111122223333* 替换为您的账户 ID。

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: aws-efs-csi-driver
  name: efs-csi-controller-sa
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn:
 arn:aws:iam::111122223333:role/AmazonEKS_EFS_CSI_DriverRole
```

b. 在集群上创建 Kubernetes 服务账户。使用您创建的名为 *AmazonEKS_EFS_CSI_DriverRole* 的 IAM 角色注释名为 `efs-csi-controller-sa` 的 Kubernetes 服务账户。

```
kubectl apply -f efs-service-account.yaml
```

安装 Amazon EFS 驱动程序

使用清单安装 Amazon EFS CSI 驱动程序。

> Important
>
> 预设情况下，使用 TLS 对传输中的数据进行加密。使用传输中加密，数据在通过网络传输到 Amazon EFS 服务期间进行加密。要禁用它并使用 NFSv4 挂载卷，请在您的持久性卷清单中将 `volumeAttributes` 字段 `encryptInTransit` 设置为 `"false"`。有关示例清单，请参阅 GitHub 上的传输中加密示例。

Manifest (private registry)

如果您想下载带清单的镜像，我们建议首先尝试以下步骤，以从私有 Amazon ECR 注册表中提取受保护的镜像。

使用私有 Amazon ECR 注册表中存储的镜像来安装驱动程序

1. 下载清单。

```
kubectl kustomize \
    "github.com/kubernetes-sigs/aws-efs-csi-driver/deploy/kubernetes/overlays/stable/ecr?ref=release-" > private-ecr-driver.yaml
```

Note

如果您遇到无法通过添加 IAM 权限来解决的问题，请改为尝试“清单（公有注册表）”步骤。

2. 运行命令将文件中的 `us-west-2` 替换为您的 Amazon Web Services 区域。

```
sed -i.bak -e 's|us-west-2|region-code|' private-ecr-driver.yaml
```

3. 运行命令以替换文件中的 `602401143452`。

```
sed -i.bak -e 's|602401143452||' private-ecr-driver.yaml
```

4. 编辑 `private-ecr-driver.yaml` 文件并删除用于创建 Kubernetes 服务账户的以下行。由于服务账户是在上一步骤中创建的，因此无需执行此操作。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: aws-efs-csi-driver
  name: efs-csi-controller-sa
  namespace: kube-system
---
```

5. 运用该清单。

```
kubectl apply -f private-ecr-driver.yaml
```

创建 Amazon EFS 文件系统

Amazon EFS CSI 驱动程序支持 Amazon EFS 接入点，这些接入点是 Amazon EFS 文件系统中特定于应用程序的入口点，便于更轻松地在多个 pods 之间共享文件系统。接入点可以为通过接入点发出的所有文件系统请求强制执行用户身份，并为每个 pod 强制执行根目录。有关更多信息，请参阅 GitHub 上的 Amazon EFS 接入点。

Important

您必须在同一个终端中完成以下步骤，因为这些步骤中设置并使用了变量。

为您的 Amazon EKS 集群创建 Amazon EFS 文件系统

1. 检索您的集群所在的 VPC ID，并将其存储在变量中，以便在后续步骤中使用。将 *my-cluster* 替换为您的集群名称。

```
vpc_id=$(aws eks describe-cluster \
    --name my-cluster \
    --query "cluster.resourcesVpcConfig.vpcId" \
    --output text)
```

2. 检索您的集群的 VPC 的 CIDR 范围，并将其存储在变量中，以便在后续步骤中使用。

```
cidr_range=$(aws ec2 describe-vpcs \
    --vpc-ids $vpc_id \
    --query "Vpcs[].CidrBlock" \
    --output text)
```

3. 创建一个安全组，该安全组包含一条允许您的 Amazon EFS 挂载点的入站 NFS 流量的入站规则。
   a. 创建安全组。将 *example values* 替换为您自己的值。

```
security_group_id=$(aws ec2 create-security-group \
    --group-name MyEfsSecurityGroup \
    --description "My EFS security group" \
    --vpc-id $vpc_id \
    --output text)
```

   b. 创建一条入站规则，该入站规则允许来自您的集群 VPC 的 CIDR 的入站 NFS 流量。

```
aws ec2 authorize-security-group-ingress \
    --group-id $security_group_id \
    --protocol tcp \
    --port 2049 \
    --cidr $cidr_range
```

   Important

   要进一步限制对文件系统的访问，您可以将 CIDR 用于子网，而不是 VPC。

4. 为您的 Amazon EKS 集群创建 Amazon EFS 文件系统。
   a. 创建文件系统。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

```
file_system_id=$(aws efs create-file-system \
    --region region-code \
    --performance-mode generalPurpose \
    --query 'FileSystemId' \
    --output text)
```

   b. 创建挂载目标。
      i. 确定您的集群节点的 IP 地址。

```
kubectl get nodes
```

      输出示例如下。

```
NAME                                             STATUS   ROLES    AGE   VERSION
ip-192-168-56-0.region-code.compute.internal   Ready    <none>   19m   v1.19.6-
eks-49a6c0
```

      ii. 确定 VPC 中子网的 ID 以及子网所在的可用区。

```
aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$vpc_id" \
```

```
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

输出示例如下。

```
|                          DescribeSubnets                           |
+------------------+--------------------+----------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId           |
+------------------+--------------------+----------------------------+
|  region-codec    |  192.168.128.0/19  |  subnet-EXAMPLE6e421a0e97  |
|  region-codeb    |  192.168.96.0/19   |  subnet-EXAMPLEd0503db0ec  |
|  region-codec    |  192.168.32.0/19   |  subnet-EXAMPLEe2ba886490  |
|  region-codeb    |  192.168.0.0/19    |  subnet-EXAMPLE123c7c5182  |
|  region-codea    |  192.168.160.0/19  |  subnet-EXAMPLE0416ce588p  |
+------------------+--------------------+----------------------------+
```

iii. 为节点所在的子网添加挂载目标。从前两个步骤的输出来看，集群有一个 IP 地址为 `192.168.56.0` 的节点。该 IP 地址位于 ID 为 `subnet-EXAMPLEe2ba886490` 的子网的 `CidrBlock` 内。因此，以下命令会为节点所在的子网创建一个挂载目标。如果集群中有更多节点，那么您应该为每个 AZ 中有节点的子网运行一次命令，将 `subnet-EXAMPLEe2ba886490` 替换为相应的子网 ID。

```
aws efs create-mount-target \
    --file-system-id $file_system_id \
    --subnet-id subnet-EXAMPLEe2ba886490 \
    --security-groups $security_group_id
```

部署示例应用程序

您可以部署动态创建持久性卷的示例应用程序，也可以手动创建持久性卷。您可以用其他应用程序替换本节中给出的示例。

Dynamic

> **Important**
>
> 不能将动态预置与 Fargate 节点一起使用。

先决条件

您必须使用版本 `1.2x` 或更高版本的 Amazon EFS CSI 驱动程序。

部署使用控制器所创建的持久性卷的示例应用程序

此过程利用来自 Amazon EFS 容器存储接口 (CSI) 驱动程序 GitHub 存储库的动态预置示例。它通过 Amazon EFS 接入点和 pod 使用的持久性卷声明 (PVC, Persistent Volume Claim) 动态创建一个持久性卷。

1. 为 EFS 创建存储类。有关所有参数和配置选项，请参阅 GitHub 上的 Amazon EFS CSI 驱动程序。
   a. 检索您的 Amazon EFS 文件系统 ID。您可以在 Amazon EFS 控制台中查找此信息，或者使用以下 Amazon CLI 命令。

   ```
   aws efs describe-file-systems --query "FileSystems[*].FileSystemId" --output text
   ```

   输出示例如下。

   ```
   fs-582a03f3
   ```

b. 下载 Amazon EFS 的 `StorageClass` 清单。

```
curl -o storageclass.yaml https://raw.githubusercontent.com/kubernetes-sigs/
aws-efs-csi-driver/master/examples/kubernetes/dynamic_provisioning/specs/
storageclass.yaml
```

c. 编辑 文件。请找到以下行，然后将 `fileSystemId` 的值替换为文件系统 ID。

```
fileSystemId: fs-582a03f3
```

d. 部署存储类。

```
kubectl apply -f storageclass.yaml
```

2. 通过部署利用 `PersistentVolumeClaim` 的 Pod 来测试自动预置：

a. 下载一个清单，该清单将部署一个 pod 和一个 PersistentVolumeClaim。

```
curl -o pod.yaml https://raw.githubusercontent.com/kubernetes-sigs/aws-efs-csi-
driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

b. 使用示例应用程序和 pod 使用的 PersistentVolumeClaimm 来部署 pod。

```
kubectl apply -f pod.yaml
```

3. 确定运行控制器的 pods 的名称。

```
kubectl get pods -n kube-system | grep efs-csi-controller
```

输出示例如下。

```
efs-csi-controller-74ccf9f566-q5989   3/3     Running   0          40m
efs-csi-controller-74ccf9f566-wswg9   3/3     Running   0          40m
```

4. 几秒钟后，您可以观察到控制器开始接受更改（已编辑，旨在提高可读性）。将 *74ccf9f566-q5989* 替换为来自上一个命令输出中的一个 pods 的值。

```
kubectl logs efs-csi-controller-74ccf9f566-q5989 \
    -n kube-system \
    -c csi-provisioner \
    --tail 10
```

输出示例如下。

```
...
1 controller.go:737] successfully created PV pvc-5983ffec-96cf-40c1-9cd6-e5686ca84eca
 for PVC efs-claim and csi volume name fs-95bcec92::fsap-02a88145b865d3a87
```

如果未看到上一个输出，请使用其他控制器 pods 之一运行上一个命令。

5. 确认已创建状态为 `Bound` 至 `PersistentVolumeClaim` 的持久性卷：

```
kubectl get pv
```

输出示例如下。

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
 STATUS   CLAIM               STORAGECLASS   REASON   AGE
pvc-5983ffec-96cf-40c1-9cd6-e5686ca84eca   20Gi       RWX            Delete
 Bound    default/efs-claim   efs-sc                  7m57s
```

6. 查看有关所创建的 PersistentVolumeClaim 的详细信息。

```
kubectl get pvc
```

输出示例如下。

```
NAME        STATUS   VOLUME                                     CAPACITY   ACCESS
 MODES   STORAGECLASS   AGE
efs-claim   Bound    pvc-5983ffec-96cf-40c1-9cd6-e5686ca84eca   20Gi       RWX
     efs-sc         9m7s
```

7. 请查看示例应用程序 pod 的状态，直到 STATUS 变成 Running。

```
kubectl get pods -o wide
```

输出示例如下。

```
NAME          READY   STATUS    RESTARTS   AGE   IP               NODE
                              NOMINATED NODE   READINESS GATES
efs-example   1/1     Running   0          10m   192.168.78.156
  ip-192-168-73-191.region-code.compute.internal   <none>           <none>
```

Note

如果 pod 没有列出 IP 地址，请确保为节点所在的子网添加了挂载目标（如 创建 Amazon EFS 文件系统 (p. 213) 末尾所述）。否则该 pod 不会退出 ContainerCreating 状态。列出 IP 地址后，pod 可能需要几分钟才能达到 Running 状态。

8. 确认数据已写入到卷。

```
kubectl exec efs-app -- bash -c "cat data/out"
```

输出示例如下。

```
...
Tue Mar 23 14:29:16 UTC 2021
Tue Mar 23 14:29:21 UTC 2021
Tue Mar 23 14:29:26 UTC 2021
Tue Mar 23 14:29:31 UTC 2021
...
```

9. （可选）终止运行 pod 的 Amazon EKS 节点并等待重新安排运行 pod。或者，您也可以删除 pod 并重新部署它。再次完成上一步，确认输出包含先前的输出。

Static

部署使用您创建的持久性卷的示例应用程序

此过程利用来自 Amazon EFS 容器存储接口 (CSI) 驱动程序 GitHub 存储库的多个容器组（pod）读写许多示例来使用静态预置的 Amazon EFS 持久性卷，并使用 ReadWriteMany 访问模式从多个 pods 访问它。

1. 将 Amazon EFS 容器存储接口 (CSI) 驱动程序 GitHub 存储库克隆到您的本地系统。

```
git clone https://github.com/kubernetes-sigs/aws-efs-csi-driver.git
```

2. 导航到 multiple_pods 示例目录。

```
cd aws-efs-csi-driver/examples/kubernetes/multiple_pods/
```

3. 检索您的 Amazon EFS 文件系统 ID。您可以在 Amazon EFS 控制台中查找此信息，或者使用以下 Amazon CLI 命令。

```
aws efs describe-file-systems --query "FileSystems[*].FileSystemId" --output text
```

输出示例如下。

```
fs-582a03f3
```

4. 编辑 specs/pv.yaml 文件并将 volumeHandle 值替换为您的 Amazon EFS 文件系统 ID。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: efs-pv
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: efs-sc
  csi:
    driver: efs.csi.aws.com
    volumeHandle: fs-582a03f3
```

> **Note**
>
> Amazon EFS 是弹性文件系统，不会强制实施任何文件系统容量限制。创建文件系统时，不使用持久性卷和持久性卷声明中的实际存储容量值。但是，由于存储容量是 Kubernetes 的必填字段，您必须指定一个有效值，例如此示例中的 5Gi。此值不会限制 Amazon EFS 文件系统的大小。

5. 从 specs 目录部署 efs-sc 存储类、efs-claim 持久性卷声明以及 efs-pv 持久性卷。

```
kubectl apply -f specs/pv.yaml
kubectl apply -f specs/claim.yaml
kubectl apply -f specs/storageclass.yaml
```

6. 列出默认命名空间中的持久性卷。查找具有 default/efs-claim 声明的持久性卷。

```
kubectl get pv -w
```

输出示例如下。

```
NAME     CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
 STORAGECLASS   REASON   AGE
efs-pv   5Gi        RWX            Retain           Bound    default/efs-claim
 efs-sc                 2m50s
```

在 STATUS 变为 Bound 之前，请勿继续执行下一步。

7. 从 specs 目录部署 app1 和 app2 示例应用程序。

```
kubectl apply -f specs/pod1.yaml
kubectl apply -f specs/pod2.yaml
```

8. 查看默认命名空间中的 pods 并等待 app1 和 app2 pods 的 STATUS 变为 Running 状态。

```
kubectl get pods --watch
```

Note

可能需要几分钟 pods 才能达到 Running 状态。

9. 描述持久性卷。

```
kubectl describe pv efs-pv
```

输出示例如下。

```
Name:            efs-pv
Labels:          none
Annotations:     kubectl.kubernetes.io/last-applied-configuration:
                   {"apiVersion":"v1","kind":"PersistentVolume","metadata":
{"annotations":{},"name":"efs-pv"},"spec":{"accessModes":
["ReadWriteMany"],"capaci...
                 pv.kubernetes.io/bound-by-controller: yes
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    efs-sc
Status:          Bound
Claim:           default/efs-claim
Reclaim Policy:  Retain
Access Modes:    RWX
VolumeMode:      Filesystem
Capacity:        5Gi
Node Affinity:   none
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            efs.csi.aws.com
    VolumeHandle:      fs-582a03f3
    ReadOnly:          false
    VolumeAttributes:  none
Events:                none
```

Amazon EFS 文件系统 ID 将作为 VolumeHandle 列出。

10. 验证 app1 pod 是否成功将数据写入卷。

```
kubectl exec -ti app1 -- tail /data/out1.txt
```

输出示例如下。

```
...
Mon Mar 22 18:18:22 UTC 2021
Mon Mar 22 18:18:27 UTC 2021
Mon Mar 22 18:18:32 UTC 2021
Mon Mar 22 18:18:37 UTC 2021
```

```
...
```

11. 验证 app2 pod 在卷中显示的数据与 app1 写入卷的数据相同。

```
kubectl exec -ti app2 -- tail /data/out1.txt
```

输出示例如下。

```
...
Mon Mar 22 18:18:22 UTC 2021
Mon Mar 22 18:18:27 UTC 2021
Mon Mar 22 18:18:32 UTC 2021
Mon Mar 22 18:18:37 UTC 2021
...
```

12. 完成试验时，请删除此示例应用程序的资源以进行清理。

```
kubectl delete -f specs/
```

您还可以手动删除您创建的文件系统和安全组。

Amazon FSx for Lustre CSI 驱动程序

FSx for Lustre Container Storage Interface (CSI) 驱动程序提供了一个 CSI 接口，允许 Amazon EKS 集群管理 FSx for Lustre 文件系统的生命周期。有关更多信息，请参阅《FSx for Lustre 用户指南》。

本主题介绍了如何部署 FSx for Lustre CSI 驱动程序到您的 Amazon EKS 集群，并验证它是否正常工作。我们建议使用 0.4.0 版本的驱动程序。

> Note
>
> 此驱动程序在 Kubernetes 版本 1.23 及更高版本的 Amazon EKS 集群和节点上受支持。Fargate 不支持该驱动程序。Amazon EKS 集群上不支持 FSx for Lustre CSI 驱动程序的 Alpha 功能。驱动程序处于 Beta 版本。它经过了严格测试且 Amazon EKS 支持将它用于生产。不会删除对此驱动程序的支持，尽管详细信息可能会发生变化。如果驱动程序的架构或原理图发生变化，则将提供迁移到下一版本的说明。

有关可用参数的详细说明和演示驱动程序功能的完整示例，请参阅 GitHub 上的 FSx for Lustre 容器存储接口 (CSI) 驱动程序项目。

先决条件

您必须：

- 您的设备或 Amazon CloudShell 上安装并配置了 2.7.21 版或更高版本，或 1.25.46 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 aws configure 进行快速配置。
- 您的设备或 0.110.0 上安装了 eksctl 版或更高版本的 Amazon CloudShell 命令行工具。要安装或更新 eksctl，请参阅 安装或更新 eksctl (p. 10)。
- 您的设备或 Amazon CloudShell 上安装了 kubectl 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 1.22，则可以将 kubectl 版本 1.21、1.22 或 1.23 用于它。要安装或升级 kubectl，请参阅 安装或更新 kubectl (p. 4)。

以下步骤帮助您创建简单的测试集群并为其配置自定义联网，以便您可以了解它是如何工作的。我们不建议将测试集群用于生产工作负载。在本教程中，我们建议使用 *example values*，除非注意到要替换它们。

您可以在完成生成集群的步骤时替换任何 *example value*。我们建议您在同一个终端中完成所有步骤，因为这些步骤中设置并使用了变量，而且这些变量不会存在于不同的终端中。

将 FSx for Lustre CSI 驱动程序部署到 Amazon EKS 集群

1. 设置一些要在其余步骤中使用的变量。将 *my-csi-fsx-cluster* 替换为您要创建的测试集群的名称，并将 *region-code* 替换为您要在其中创建测试集群的 Amazon Web Services 区域。

```
export cluster_name=my-csi-fsx-cluster
export region_code=region-code
```

2. 创建测试集群。

```
eksctl create cluster \
  --name $cluster_name \
  --region $region_code \
  --with-oidc \
  --ssh-access \
  --ssh-public-key my-key
```

集群预配置需要几分钟时间。在集群创建过程中，您将看到几行输出。输出的最后一行类似于以下示例行。

```
[#]  EKS cluster "my-csi-fsx-cluster" in "region-code" region is ready
```

3. 使用以下命令为驱动程序创建一个 Kubernetes 服务账户，并将 `AmazonFSxFullAccess` Amazon 托管策略附加到该服务账户。

```
eksctl create iamserviceaccount \
    --name fsx-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --attach-policy-arn arn:aws:iam::aws:policy/AmazonFSxFullAccess \
    --approve \
    --role-name AmazonEKSFSxLustreCSIDriverFullAccess \
    --region $region_code
```

输出示例如下。

创建服务账户时，您将看到几行输出。输出的最后一行类似于以下行。

```
[#]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/fsx-csi-controller-sa",
        create serviceaccount "kube-system/fsx-csi-controller-sa",
    } }
[#]  building iamserviceaccount stack "eksctl-my-csi-fsx-cluster-addon-
iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#]  deploying stack "eksctl-my-csi-fsx-cluster-addon-iamserviceaccount-kube-
system-fsx-csi-controller-sa"
[#]  waiting for CloudFormation stack "eksctl-my-csi-fsx-cluster-addon-
iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#]  created serviceaccount "kube-system/fsx-csi-controller-sa"
```

记录已部署的 Amazon CloudFormation 堆栈的名称。在上面的示例输出中，堆栈的名称为 eksctl-*my-csi-fsx-cluster*-addon-iamserviceaccount-kube-system-*fsx-csi-controller-sa*。

4. 使用以下命令部署驱动程序。

Note

您可以在 GitHub 上查看应用于 aws-fsx-csi-driver 中的内容。

```
kubectl apply -k "github.com/kubernetes-sigs/aws-fsx-csi-driver/deploy/kubernetes/
overlays/stable/?ref=master"
```

输出示例如下。

```
serviceaccount/fsx-csi-controller-sa created
serviceaccount/fsx-csi-node-sa created
clusterrole.rbac.authorization.k8s.io/fsx-csi-external-provisioner-role created
clusterrole.rbac.authorization.k8s.io/fsx-external-resizer-role created
clusterrolebinding.rbac.authorization.k8s.io/fsx-csi-external-provisioner-binding
 created
clusterrolebinding.rbac.authorization.k8s.io/fsx-csi-resizer-binding created
deployment.apps/fsx-csi-controller created
daemonset.apps/fsx-csi-node created
csidriver.storage.k8s.io/fsx.csi.aws.com created
```

5. 记录所创建角色的 ARN。如果您早些时候没有注意到它并且没有在 Amazon CLI 输出中再提供它，您可以执行以下操作以在 Amazon Web Services Management Console 中查看它。
   a. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
   b. 确保将控制台设置为您在其中创建 IAM 角色的 Amazon Web Services 区域，然后选择 Stacks（堆栈）。
   c. 选择名为 eksctl-*my-csi-fsx-cluster*-addon-iamserviceaccount-kube-system-*fsx-csi-controller-sa* 的堆栈。
   d. 选择 Outputs（输出）选项卡。Role1 ARN 列于 Outputs（1）页面上。
6. 使用以下命令修补驱动程序部署以添加之前创建的服务账户。将 ARN 替换为您记下的 ARN。请将 *111122223333* 替换为您的账户 ID。

```
kubectl annotate serviceaccount -n kube-system fsx-csi-controller-sa \
 eks.amazonaws.com/role-
arn=arn:aws:iam::111122223333:role/AmazonEKSFSxLustreCSIDriverFullAccess --
overwrite=true
```

输出示例如下。

```
serviceaccount/fsx-csi-controller-sa annotated
```

部署 Kubernetes 存储类、持久卷注册和示例应用程序，以验证 CSI 驱动程序是否正常工作

此过程利用 FSx for Lustre 容器存储接口 (CSI) 驱动程序 GitHub 存储库来使用动态预置的 FSx for Lustre 卷。

1. 注意集群的安全组。您可以在 Amazon Web Services Management Console 中的 Networking（联网）部分下或通过使用以下 Amazon CLI 命令来查看它。

```
aws eks describe-cluster --name $cluster_name --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

2. 根据《Amazon FSx for Lustre 用户指南》中的 Amazon VPC 安全组显示的标准，为您的 Amazon FSx 文件系统创建安全组。对于 VPC，选择 Networking（联网）部分下显示的集群的 VPC。对于“与 Lustre 客户端关联的安全组”，请使用您的集群安全组。您可以单独保留出站规则以允许所有流量。

3. 使用下面的命令下载存储类清单。

```
curl -o storageclass.yaml https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
```

4. 编辑 storageclass.yaml 文件的参数部分。请将每个 *example value* 替换为您自己的值。

```
parameters:
  subnetId: subnet-0eabfaa81fb22bcaf
  securityGroupIds: sg-068000ccf82dfba88
  deploymentType: PERSISTENT_1
  automaticBackupRetentionDays: "1"
  dailyAutomaticBackupStartTime: "00:00"
  copyTagsToBackups: "true"
  perUnitStorageThroughput: "200"
  dataCompressionType: "NONE"
  weeklyMaintenanceStartTime: "7:09:00"
  fileSystemTypeVersion: "2.12"
```

   - **subnetId** – 应在其中创建 Amazon FSx for Lustre 文件系统的子网 ID。并非所有可用区都支持 Amazon FSx for Lustre。打开 https://console.aws.amazon.com/fsx/ 中 Amazon FSx for Lustre 控制台，确认您要使用的子网是否位于支持的可用区中。该子网可以包含您的节点，也可以是不同的子网或 VPC：
     - 您可以通过在 Compute（计算）部分下选择节点组来在 Amazon Web Services Management Console 中检查节点子网。
     - 如果您指定的子网不是节点所在的子网，则必须已连接 VPC，并且必须确保已在您的安全组中打开必要的端口。
   - **securityGroupIds** – 您为文件系统创建的安全组的 ID。
   - **deploymentType**（可选）– 文件系统部署类型。有效值为 SCRATCH_1、SCRATCH_2、PERSISTENT_1 和 PERSISTENT_2。有关部署类型的更多信息，请参阅创建 Amazon FSx for Lustre 文件系统。
   - 其他参数（可选）– 有关其他参数的信息，请参阅 GitHub 上的编辑 StorageClass。

5. 创建存储类清单。

```
kubectl apply -f storageclass.yaml
```

   输出示例如下。

```
storageclass.storage.k8s.io/fsx-sc created
```

6. 下载持久卷注册清单。

```
curl -o claim.yaml https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. （可选）编辑 claim.yaml 文件。根据您的存储需求和上一步选择的 deploymentType，将 *1200Gi* 更改为下面列出的增量值之一。

```
storage: 1200Gi
```

   - SCRATCH_2 和 PERSISTENT – **1.2 TiB**、**2.4 TiB**，或 2.4TiB 之上 2.4TiB 的增量。
   - SCRATCH_1 和 **1.2 TiB** – **2.4 TiB**、**3.6 TiB**，或 3.6TiB 之上 3.6TiB 的增量。

8. 创建持久卷注册。

```
kubectl apply -f claim.yaml
```

输出示例如下。

```
persistentvolumeclaim/fsx-claim created
```

9. 确认已预配置文件系统。

```
kubectl describe pvc
```

输出示例如下。

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
...
```

> **Note**
>
> Status 可能会在 5-10 分钟内显示为 Pending，然后才会更改为 Bound。请勿继续下一步，直到 Status 变成 Bound。如果 Status 显示 Pending 10 分钟以上，使用 Events 中的警告消息作为解决任何问题的参考。

10. 部署示例应用程序。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/
master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11. 验证示例应用程序正在运行。

```
kubectl get pods
```

输出示例如下。

```
NAME      READY   STATUS              RESTARTS   AGE
fsx-app   1/1     Running             0          8s
```

12. 验证应用程序是否正确挂载了文件系统。

```
kubectl exec -ti fsx-app -- df -h
```

输出示例如下。

```
Filesystem                  Size  Used Avail Use% Mounted on
overlay                      80G  4.0G   77G   5% /
tmpfs                        64M     0   64M   0% /dev
tmpfs                       3.8G     0  3.8G   0% /sys/fs/cgroup
192.0.2.0@tcp:/abcdef01  1.1T  7.8M  1.1T   1% /data
/dev/nvme0n1p1               80G  4.0G   77G   5% /etc/hosts
shm                          64M     0   64M   0% /dev/shm
tmpfs                       6.9G   12K  6.9G   1% /run/secrets/kubernetes.io/
serviceaccount
tmpfs                       3.8G     0  3.8G   0% /proc/acpi
tmpfs                       3.8G     0  3.8G   0% /sys/firmware
```

13. 验证示例应用程序已将数据写入 FSx for Lustre 文件系统。

```
kubectl exec -it fsx-app -- ls /data
```

输出示例如下。

```
out.txt
```

此示例输出显示示例应用程序成功编写了 `out.txt` 文件到文件系统。

Note

删除集群之前，请务必删除 FSx for Lustre 文件系统。有关更多信息，请参阅《FSx for Lustre 用户指南》中的清理资源。

Amazon FSx for NetApp ONTAP CSI 驱动程序

NetApp 的 Astra Trident 使用符合容器存储接口（CSI）的驱动程序提供动态存储编排。该服务允许 Amazon EKS 集群管理 Amazon FSx for NetApp ONTAP 文件系统提供支持的持久卷（PV）的生命周期。要开始使用，请参阅 Astra Trident 文档中的将 Astra Trident 与 Amazon FSx for NetApp ONTAP 结合使用。

Amazon FSx for NetApp ONTAP 是一种存储服务，允许您在云中启动和运行完全托管的 ONTAP 文件系统。ONTAP 是 NetApp 的文件系统技术，它提供了一套广泛采用的数据访问和数据管理功能。Amazon FSx for NetApp ONTAP 提供本地 NetApp 文件系统的功能、性能和 API，具有完全托管式 Amazon 服务的灵活性、可扩展性和简单性。有关更多信息，请参阅《FSx for ONTAP 用户指南》。

Amazon EKS 联网

您的 Amazon EKS 集群是在 VPC 中创建的。容器组（pod）联网由 Amazon VPC 容器网络接口（CNI）插件提供。本章包含可以进一步了解集群联网的以下主题。

主题

- Amazon EKS VPC 和子网要求和注意事项 (p. 226)
- 为 Amazon EKS 集群创建 VPC (p. 228)
- Amazon EKS 安全组要求和注意事项 (p. 231)
- Amazon EKS 中使用适用于 Kubernetes 的 Amazon VPC CNI 插件的 Pod 联网 (p. 233)
- 安装 Amazon Load Balancer Controller 附加组件 (p. 287)
- 管理 CoreDNS 附加组件 (p. 294)
- 管理 kube-proxy 附加组件 (p. 300)
- 安装 Calico 附加组件 (p. 306)

Amazon EKS VPC 和子网要求和注意事项

在创建集群时，您将指定 VPC 以及位于不同可用区中的至少两个子网。本主题概述了 Amazon EKS 对于您用于集群的 VPC 和子网的特定要求和注意事项。如果您没有与 Amazon EKS 结合使用的 VPC，则可以使用 Amazon EKS 提供的 Amazon CloudFormation 模板创建一个 (p. 228)。

VPC 要求和注意事项

在创建集群时，您指定的 VPC 必须满足以下要求和注意事项：

- VPC 必须有足够数量的 IP 地址来供集群、任何节点和您想要创建的其他 Kubernetes 资源使用。如果要使用的 VPC 没有足够数量的 IP 地址，请尝试增加可用 IP 地址的数量。您可以通过关联其他无类别域间路由（CIDR）块与 VPC 来执行此操作。您可以在创建集群之前或之后将私有（RFC 1918）和公有（非 RFC 1918）CIDR 块与您的 VPC 关联。集群识别与 VPC 关联的 CIDR 块最多可能需要五个小时的时间。

  您可以通过将中转网关与共享服务 VPC 结合使用来节省 IP 地址的使用。有关更多信息，请参阅具有共享服务的隔离 VPC 和混合网络中的 Amazon EKS VPC 可路由 IP 地址保护模式。
- 如果您想让 Kubernetes 分配 `IPv6` 地址给 pods 和服务，请将 `IPv6` CIDR 块与您的 VPC 关联。有关更多信息，请参阅《Amazon VPC 用户指南》中的将 `IPv6` CIDR 块与您的 VPC 关联。
- VPC 必须具有 `DNS` 主机名和 `DNS` 解析支持。否则，节点无法注册到集群。有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 的 DNS 属性。
- VPC 可能需要使用 Amazon PrivateLink 的 VPC 节点。有关更多信息，请参阅the section called “子网要求和注意事项” (p. 227)。

如果您使用 Kubernetes `1.14` 或更早版本创建集群，Amazon EKS 会将以下标签添加到您的 VPC 中：

密钥	值
`kubernetes.io/cluster/`*`my-cluster`*	`owned`

此标签仅由 Amazon EKS 使用。您可以在不影响服务的情况下删除标签。它不适用于版本 `1.15` 或更高版本的集群。

子网要求和注意事项

在创建集群时，Amazon EKS 将在您指定的子网中创建 2–4 个弹性网络接口。这些网络接口可实现集群和 VPC 之间的通信。这些网络接口还支持 Kubernetes 功能，例如 `kubectl exec` 和 `kubectl logs`。每个 Amazon EKS 创建的网络接口的描述中都有 `Amazon EKS cluster-name` 文本。

当您创建集群时，Amazon EKS 可以在您指定的任何子网中创建其网络接口。创建集群后，您无法更改 Amazon EKS 在哪些子网中创建其网络接口。当您更新集群的 Kubernetes 版本时，Amazon EKS 会删除其创建的原始网络接口，然后创建新的网络接口。这些网络接口可以在与原始网络接口相同的子网中创建，也可以在与原始网络接口不同的子网中创建。要控制在哪些子网中创建网络接口，可以在创建集群时将指定的子网数限制为只有两个子网。

您在创建集群时指定的子网必须满足以下要求：

- 每个子网必须至少有六个 IP 地址以供 Amazon EKS 使用。但是，我们建议至少使用 16 个 IP 地址。
- 子网不能驻留在 Amazon Outposts、Amazon Wavelength 或 Amazon 本地区域中。但是，如果您的 VPC 中有这些子网，则可以向这些类型的子网部署自行管理的节点 (p. 106)和 Kubernetes 资源。
- 子网必须使用基于 IP 地址的命名。Amazon EKS 不支持 Amazon EC2 基于资源的名称。
- 子网可以是公有子网或私有子网。但是，如果可能，我们建议您指定私有子网。公有子网是指包含路由表的子网，这个路由表中包含指向互联网网关的路由，而私有子网是指具有一个其中不包含指向互联网网关的路由的路由表的子网。

您可以将节点和 Kubernetes 资源部署到您在创建集群时指定的相同子网。但是，这并不是必要的。这是因为，您还可以将节点和 Kubernetes 资源部署到您在创建集群时未指定的子网。如果您将节点部署到不同的子网，Amazon EKS 不会在这些子网中创建集群网络接口。您向其中部署节点和 Kubernetes 资源的任何子网必须满足以下要求：

- 子网必须有足够的可用 IP 地址才能将所有节点和 Kubernetes 资源部署到其中。
- 子网必须使用基于 IP 地址的命名。Amazon EKS 不支持 Amazon EC2 基于资源的名称。
- 如果您想让 Kubernetes 分配 `IPv6` 地址到 pods 和服务，您必须拥有一个 `IPv6` CIDR 块和一个与您的子网相关联的 `IPv4` CIDR 块。有关更多信息，请参阅《Amazon VPC 用户指南》中的将 `IPv6` CIDR 块与您的子网关联。与子网关联的路由表必须包含到 `IPv4` 和 `IPv6` 地址的路由。有关更多信息，请参阅《Amazon VPC 用户指南》中的路由。容器组（pod）只分配了 `IPv6` 地址。但是，Amazon EKS 为您的集群和您的节点创建的网络接口被分配了 `IPv4` 和 `IPv6` 地址。
- 如果您需要从互联网入站访问您的 pods，请确保至少有一个公有子网具有足够的可用 IP 地址，以便将负载均衡器和入口部署到其中。您可以将负载均衡器部署到公有子网。负载均衡器可以对私有或公有子网中的 pods 进行负载均衡。我们建议尽可能将节点部署到私有子网。
- 如果您计划将节点部署到公有子网，则子网必须自动分配 `IPv4` 公有地址或 `IPv6` 地址。如果将节点部署到具有关联的 `IPv6` CIDR 块的私有子网，私有子网还必须自动分配 `IPv6` 地址。如果您使用 Amazon EKS Amazon CloudFormation 模板 (p. 228)在 2020 年 3 月 26 日之后部署您的 VPC，此设置已启用。如果您在此日期之前使用模板部署 VPC，或者您使用自己的 VPC，则必须手动启用此设置。有关更多信息，请参阅《Amazon VPC 用户指南》中的修改您的子网的公有 `IPv4` 寻址属性和修改您的子网的 `IPv6` 寻址属性。
- 如果您将节点部署到的子网是私有子网，且其路由表不包含到网络地址转换（NAT）设备（`IPv4`）或仅限出口的网关（`IPv6`）的路由，则添加使用 Amazon PrivateLink 的 VPC 端点到您的 VPC。您的节点和 pods 需要与之通信的所有 Amazon Web Services 都需要 VPC 端点。示例包括 Amazon ECR、Elastic Load Balancing、Amazon CloudWatch、Amazon Security Token Service 和 Amazon Simple Storage Service（Amazon S3）。端点必须包括节点所在的子网。并非所有 Amazon Web Services 都支持 VPC 端点。有关更多信息，请参阅什么是 Amazon PrivateLink？和与 Amazon PrivateLink 集成的 Amazon 服务。有关更多 Amazon EKS 要求的列表，请参阅 the section called “私有集群要求” (p. 56)。
- 如果要将负载均衡器部署到子网，则子网必须具有以下标签：

- 私有子网

密钥	值
kubernetes.io/role/internal-elb	1

- 公有子网

密钥	值
kubernetes.io/role/elb	1

当创建版本为 1.18 和更早版本的 Kubernetes 集群时，Amazon EKS 会向您在创建集群时指定的所有子网添加以下标签。

密钥	值
kubernetes.io/cluster/*my-cluster*	shared

当您创建 1.19 版本和更高版本的 Kubernetes 集群时，Amazon EKS 不会将标签添加到您的子网中。如果在早于 1.19 版本的集群所用的子网上存在标签，而且您已将集群更新到 1.19 或更高版本，则不会从子网中删除该标签。但是，在更新到 1.19 完成后，您可以在不中断服务的情况下将其删除。但如果您使用版本 2.1.1 或更早版本的 Amazon Load Balancer Controller (p. 287)，您无法删除标签。这些版本的 Amazon 负载均衡器控制器还需要此标签。

如果您使用 eksctl 或者任一 Amazon EKS Amazon CloudFormation VPC 模板部署了 VPC，则以下情况适用：

- 2020 年 3 月 26 日或之后 – 公有子网将公有 IPv4 地址自动分配给部署到公有子网的新节点。

此更改通过以下方式影响部署到公有子网的新节点组：

- 托管节点组 (p. 90) – 如果节点组在 2020 年 4 月 22 日或之后部署到公有子网，则该公有子网必须启用自动分配公有 IP 地址。有关更多信息，请参阅修改子网的公有 IPv4 寻址属性。
- Linux (p. 107)、Windows (p. 114) 或 Arm (p. 159) 自行管理节点组 – 如果节点组在 2020 年 3 月 26 日或之后部署到公有子网，则该公有子网必须启用自动分配公有 IP 地址。或者，节点必须使用公有 IP 地址启动。有关更多信息，请参阅修改子网的公有 IPv4 寻址属性或在实例启动期间分配公有 IPv4 地址。

为 Amazon EKS 集群创建 VPC

您可以使用 Amazon Virtual Private Cloud（Amazon VPC）将Amazon资源启动到您定义的虚拟网络中。此虚拟网络与您在自己的数据中心中运行的传统网络极为相似。但是，它带有使用亚马逊云科技的可扩展基础设施的优势。有关更多信息，请参阅《Amazon VPC 用户指南》

Amazon EKS 集群、节点和 Kubernetes 资源均部署到 VPC 中。如果您想要将现有 VPC 与 Amazon EKS 一起使用，则该 VPC 必须满足 the section called “VPC 和子网要求” (p. 226) 中所述的要求。本主题介绍如何使用 Amazon EKS 提供的 Amazon CloudFormation 模板创建符合 Amazon EKS 要求的 VPC。

先决条件

要为 Amazon EKS 创建 VPC，您必须拥有必要的 IAM 权限才能创建 Amazon VPC 资源。这些资源包括 VPC、子网、安全组、路由表和路由以及互联网和 NAT 网关。有关更多信息，请参阅《Amazon VPC 用户

指南》中的创建带有公有子网示例策略的 VPC 和《服务授权参考》中的 Amazon EC2 的操作、资源和条件键的完整列表。

您可以创建带有公有子网和私有子网、仅公有子网或仅私有子网的 VPC。

Public and private subnets

此 VPC 有两个公有子网和两个私有子网。公有子网的关联路由表具有指向互联网网关的路由。但是，私有子网的路由表没有指向互联网网关的路由。一个公有子网和一个私有子网部署到同一个可用区。其他公有子网和私有子网部署到同一 Amazon Web Services 区域的另一个可用区。我们建议大多数部署使用此选项。

使用此选项，您可以将节点部署到私有子网。此选项允许 Kubernetes 将负载均衡器部署到公有子网，这些负载均衡器可以对在私有子网的节点上运行的 pods 进行流量负载均衡。系统自动向部署到公有子网的节点分配公有 `IPv4` 地址，但公有 `IPv4` 地址不会分配给部署到私有子网的节点。

您还可以将 `IPv6` 地址分配给公有子网和私有子网中的节点。私有子网中的节点可以与集群和其他 Amazon Web Services 通信。Pods 可以通过使用 `IPv4` 地址的 NAT 网关或使用部署在各个可用区的 `IPv6` 地址的仅出站互联网网关与互联网通信。部署的安全组包含拒绝来自集群或节点以外的来源的所有入站流量但允许所有出站流量的规则。对子网进行标记，以便 Kubernetes 可以向它们部署负载均衡器。

要创建您的 VPC

1. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
2. 从导航栏中，选择一个支持 Amazon EKS 的 Amazon Web Services 区域。
3. 依次选择创建堆栈和使用新资源（标准）。
4. 在 Prerequisite - Prepare template（先决条件 - 准备模板）下，请确保选中 Template is ready（模板就绪），然后在 Specify template（指定模板）下选择 Amazon S3 URL。
5. 您可以创建仅支持 `IPv4` 的 VPC，或支持 `IPv4` 和 `IPv6` 的 VPC。请将以下 URL 粘贴到 Amazon S3 URL 下方的文本区域，然后选择 Next（下一步）：

   - `IPv4`

     ```
     https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/
     cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
     ```

6. 在 Specify stack details（指定堆栈详细信息）页面上，修改参数，然后选择 Next（下一步）。

   - 堆栈名称：为 Amazon CloudFormation 堆栈选择堆栈名称。例如，可以使用上一步中使用的模板名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。
   - VpcBlock：为您的 VPC 选择 `IPv4` CIDR 范围。您部署的每个节点、容器组（pod）和负载均衡器都会分配有此块中一个 `IPv4` 地址。默认 `IPv4` 值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。有关更多信息，请参阅 Amazon VPC 用户指南中的 VPC 和子网大小调整。您还可以在 VPC 创建后向其添加额外的 CIDR 块。如果您要创建 `IPv6` VPC，则系统从 Amazon 的 Global Unicast Address 空间自动为您分配 `IPv6` CIDR 范围。
   - PublicSubnet01Block：指定公有子网 1 的 `IPv4` CIDR 块。原定设置值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。如果您要创建 `IPv6` VPC，则在模板中为您指定此区块。
   - PublicSubnet02Block：指定公有子网 2 的 `IPv4` CIDR 块。原定设置值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。如果您要创建 `IPv6` VPC，则在模板中为您指定此区块。
   - PrivateSubnet01Block：指定私有子网 1 的 `IPv4` CIDR 块。原定设置值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。如果您要创建 `IPv6` VPC，则在模板中为您指定此区块。

- PrivateSubnet02Block：指定私有子网 2 的 IPv4 CIDR 块。原定设置值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。如果您要创建 IPv6 VPC，则在模板中为您指定此区块。

7. （可选）在 Configure stack options（配置堆栈选项）页面上，为堆栈资源添加标签，然后选择 Next（下一步）。
8. 在 Review（查看）页面中，请选择 Create stack（创建堆栈）。
9. 创建堆栈后，在控制台中选中它，然后选择 Outputs (输出)。
10. 记录所创建 VPC 的 VpcId。在您创建和节点时需要此功能。
11. 记录已创建子网的 SubnetIds，以及您将其作为公有子网还是私有子网创建。在您创建和节点时至少需要其中的两个功能。

Only public subnets

此 VPC 有三个部署到 Amazon Web Services 区域 中的不同可用区的公有子网。所有节点都会自动分配公有 IPv4 地址，并且可以通过互联网网关发送和接收互联网流量。部署了安全组，拒绝所有入站流量并允许所有出站流量。对子网进行标记，以便 Kubernetes 可以向它们部署负载均衡器。

要创建您的 VPC

1. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
2. 从导航栏中，选择一个支持 Amazon EKS 的 Amazon Web Services 区域。
3. 依次选择创建堆栈和使用新资源（标准）。
4. 在 Prepare template（准备模板）下，确保选中 Template is ready（模板已就绪），然后在 Template source（模板源）下选择 Amazon S3 URL。
5. 将以下 URL 粘贴到 Amazon S3 URL 下方的文本区域，然后选择 Next（下一步）：

```
https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

6. 在 Specify Details（指定详细信息）页面上，输入参数，然后选择 Next（下一步）。

   - 堆栈名称：为 Amazon CloudFormation 堆栈选择堆栈名称。例如，您可以将其命名为 ***amazon-eks-vpc-sample***。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。
   - VpcBlock：为您的 VPC 选择 CIDR 块。您部署的每个节点、容器组（pod）和负载均衡器都会分配有此块中一个 IPv4 地址。默认 IPv4 值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。有关更多信息，请参阅 Amazon VPC 用户指南中的 VPC 和子网大小调整。您还可以在 VPC 创建后向其添加额外的 CIDR 块。
   - Subnet01Block：指定子网 1 的 CIDR 块。原定设置值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。
   - Subnet02Block：指定子网 2 的 CIDR 块。原定设置值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。
   - Subnet03Block：指定子网 3 的 CIDR 块。原定设置值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。

7. （可选）在 Options (选项) 页面上，为您的堆栈资源添加标签。选择 Next (下一步)。
8. 在 Review 页面上，选择 Create 。
9. 创建堆栈后，在控制台中选中它，然后选择 Outputs (输出)。
10. 记录所创建 VPC 的 VpcId。在您创建和节点时需要此功能。
11. 记录已创建的子网的 SubnetIds。在您创建和节点时至少需要其中的两个功能。

Only private subnets

此 VPC 有三个部署到 Amazon Web Services 区域 中的不同可用区的私有子网。部署到子网的资源无法访问互联网，互联网也无法访问子网中的资源。该模板使用 Amazon PrivateLink 为节点通常需要访问的几个 Amazon Web Services 创建 VPC 端点。如果您的节点需要出站互联网访问，您可以在创建 VPC 后在每个子网的可用区中添加公有 NAT 网关。安全组被创建的用于拒绝所有入站流量，但部署到子网的资源除外。安全组还允许所有出站流量。对子网进行标记，以便 Kubernetes 可以向它们部署内部负载均衡器。如果您正在使用此配置创建 VPC，请参阅 the section called “私有集群要求” (p. 56) 以了解其他要求和考虑因素。

要创建您的 VPC

1. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
2. 从导航栏中，选择一个支持 Amazon EKS 的 Amazon Web Services 区域。
3. 依次选择创建堆栈和使用新资源（标准）。
4. 在 Prepare template（准备模板）下，确保选中 Template is ready（模板已就绪），然后在 Template source（模板源）下选择 Amazon S3 URL。
5. 将以下 URL 粘贴到 Amazon S3 URL 下方的文本区域，然后选择 Next（下一步）：

   ```
   https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/cloudformation/2020-10-29/amazon-eks-fully-private-vpc.yaml
   ```

6. 在 Specify Details（指定详细信息）页面上，输入参数，然后选择 Next（下一步）。
   - 堆栈名称：为 Amazon CloudFormation 堆栈选择堆栈名称。例如，您可以将其命名为 ***amazon-eks-fully-private-vpc***。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。
   - VpcBlock：为您的 VPC 选择 CIDR 块。您部署的每个节点、容器组（pod）和负载均衡器都会分配有此块中一个 `IPv4` 地址。默认 `IPv4` 值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。有关更多信息，请参阅 Amazon VPC 用户指南中的 VPC 和子网大小调整。您还可以在 VPC 创建后向其添加额外的 CIDR 块。
   - PrivateSubnet01Block：指定子网 1 的 CIDR 块。原定设置值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。
   - PrivateSubnet02Block：指定子网 2 的 CIDR 块。原定设置值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。
   - PrivateSubnet03Block：指定子网 3 的 CIDR 块。原定设置值为大多数实现提供了足够的 IP 地址，但如果没有提供，那么您可以对其进行更改。
7. （可选）在 Options (选项) 页面上，为您的堆栈资源添加标签。选择 Next (下一步)。
8. 在 Review 页面上，选择 Create 。
9. 创建堆栈后，在控制台中选中它，然后选择 Outputs (输出)。
10. 记录所创建 VPC 的 VpcId。在您创建和节点时需要此功能。
11. 记录已创建的子网的 SubnetIds。在您创建和节点时至少需要其中的两个功能。

Amazon EKS 安全组要求和注意事项

本主题介绍 Amazon EKS 集群的安全组要求。

在创建集群时，Amazon EKS 将创建一个名为 `eks-cluster-sg-`*`my-cluster`*`-`*`uniqueID`* 的安全组。该安全组具有以下默认规则：

Rule type	协议	端口	源	目标
入站	全部	全部	自身	

Rule type	协议	端口	源	目标
出站	全部	全部		0.0.0.0/0

Amazon EKS 使用以下标签对此安全组进行标记：

密钥	值
kubernetes.io/cluster/*my-cluster*	owned
aws:eks:cluster-name	*my-cluster*

Amazon EKS 会自动将此安全组关联到它还创建的以下资源：

- 在您创建集群时创建的 2–4 个弹性网络接口（在本文档其余部分中称为网络接口）。
- 您创建的任何托管节点组中节点的网络接口。

默认规则允许所有流量在集群和节点之间自由流动，并允许所有出站流量到任何目的地。在创建集群时，您可以（选择性地）指定您自己的安全组。如果您这样做，那么 Amazon EKS 还会将您指定的安全组与它为集群创建的网络接口关联起来。但是，它不会将它们与您创建的任何节点组关联起来。

您可以在Amazon Web Services Management Console中集群的 Networking（联网）部分下，确定您的集群安全组的 ID。或者，您可以使用以下 Amazon CLI 命令进行这项操作。

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

限制集群流量

如果您需要限制集群与节点之间的开放端口，则可以删除默认规则并添加集群所需的以下最少规则：

Rule type	协议	端口	源	目标
入站	TCP	443	集群安全组	
入站	TCP	10250	集群安全组	
入站（CoreDNS）	TCP 和 UDP	53	集群安全组	
出站	TCP	443		集群安全组
出站	TCP	10250		集群安全组
出站（DNS）	TCP 和 UDP	53		集群安全组

您还必须为以下流量添加规则：

- 您预计节点要用于节点间通信的任何协议和端口。
- 出站互联网访问，以便节点可以访问 Amazon EKS API 以在启动时进行集群自检和节点注册。如果您的节点没有互联网访问权限，请查看 私有集群要求 (p. 56) 了解其他注意事项。
- 从它们需要从中拉取镜像的 Amazon ECR 或其他容器注册表 API（例如 DockerHub）中拉取容器镜像的节点访问权限。有关更多信息，请参阅Amazon一般参考中的Amazon IP 地址范围。
- 节点访问 Amazon S3。

- IPv4 和 IPv6 地址需要单独的规则。

如果您考虑限制规则，我们建议您彻底测试您的所有 pods，然后再将更改的规则应用于生产集群。

如果您最初使用 Kubernetes 1.14 和 eks.3 或更早版本的平台部署集群，则考虑以下事项：

- 您可能还拥有控制面板和节点安全组。创建这些组时，它们包括上表中列出的限制规则。这些安全组不再需要，可以删除。但是，您需要确保集群安全组包含这些组所包含的规则。
- 如果您直接使用 API 部署集群，或者使用 Amazon CLI 或 Amazon CloudFormation 之类的工具创建集群且在创建集群时您未指定安全组，则 VPC 的默认安全组应用到 Amazon EKS 创建的集群网络接口。

Amazon EKS 中使用适用于 Kubernetes 的 Amazon VPC CNI 插件的 Pod 联网

Amazon EKS 支持通过适用于 Kubernetes 的 Amazon VPC 容器网络接口 (CNI, Container Network Interface) 插件进行本机 Amazon VPC 联网。该插件可以：

- 创建弹性网络接口（网络接口）并将其附加到 Amazon EC2 节点。
- 从您的 VPC 将私有 IPv4 地址分配给每个 pod 和服务。

该插件是一个在 GitHub 上进行维护的开源项目。我们建议您通过阅读 GitHub 上的 amazon-vpc-cni-k8s 和提议：通过 Amazon VPC 进行 Kubernetes 联网的 CNI 插件来熟悉该插件。amazon-vpc-cni-k8s 中的几个 CNI 配置变量在 the section called “使用案例” (p. 247) 中进行扩展。该插件完全支持在 Amazon EKS 和 Amazon 上自行管理的 Kubernetes 集群上使用。

主题

- 管理 Amazon VPC CNI plugin for Kubernetes (p. 233)
- 配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)
- 选择 pod 联网使用案例 (p. 247)
- 指标帮助程序 (p. 283)
- 备选的兼容 CNI 插件 (p. 287)

管理 Amazon VPC CNI plugin for Kubernetes

Amazon EKS 支持通过适用于 Kubernetes 的 Amazon VPC 容器网络接口 (CNI, Container Network Interface) 插件进行本机 VPC 联网。使用此插件允许 Kubernetes pods 在 pod 内具有与其在 VPC 网络上相同的 IP 地址。有关更多信息，请参阅Amazon EKS 中使用适用于 Kubernetes 的 Amazon VPC CNI 插件的 Pod 联网 (p. 233)。

为每个集群版本推荐的 Amazon VPC CNI 附加组件版本

	1.23	1.22	1.21	1.20	1.19	1.18
附加组件版本	1.11.3-eksbuild.1	1.11.3-eksbuild.1	1.11.3-eksbuild.1	1.11.3-eksbuild.1	1.11.3-eksbuild.1	1.11.3-eksbuild.1

如果您使用 Amazon Web Services Management Console 创建了 1.18 版或更高版本的集群，则 Amazon EKS 已为您安装插件作为 Amazon EKS 附加组件。如果您使用任何工具创建了 1.17 版本或更低版本的集群，或者您使用 Amazon Web Services Management Console 以外的其他工具创建了 1.18 版本或更高版本的集群，则 Amazon EKS 将插件安装为自行管理的附加组件。您可以按照 添加 Amazon VPC CNI

Amazon EKS 附加组件 (p. 234) 中的步骤将自行管理的附加组件迁移到 Amazon EKS 附加组件。如果您已将 Amazon VPC CNI plugin for Kubernetes 附加组件添加到集群，则您可以按照 更新 Amazon VPC CNI plugin for Kubernetes 附加组件 (p. 236) 和 删除 Amazon VPC CNI plugin for Kubernetes 附加组件 (p. 237) 部分中的程序来管理该附加组件。有关 Amazon EKS 附加组件的更多信息，请参阅 Amazon EKS 附加组件 (p. 341)。

要更新 Amazon EKS 附加组件版本，请参阅 更新 Amazon VPC CNI plugin for Kubernetes 附加组件 (p. 236)。要使用 Amazon EKS Amazon Elastic Container Registry 或您自己的存储库中的容器镜像更新自行管理的附加组件版本，请参阅 更新 Amazon VPC CNI plugin for Kubernetes 自行管理的附加组件 (p. 238)。

> Important
>
> 创建集群时部署的附加组件版本可能早于推荐的版本。如果您使用清单更新了自行管理的附加组件，则该版本不包括 `-eksbuild.1`。

先决条件

- 现有 版 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS (p. 4)。
- 如果您的集群为 `1.21` 版本或更高版本，确保您的 `kube-proxy` 和 CoreDNS 附加组件为服务账户令牌 (p. 395)中列出的最低版本。
- 集群的现有 Amazon Identity and Access Management (IAM) OpenID Connect (OIDC) 提供程序。要确定您是否已经拥有一个或是否要创建一个，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。
- 附加有 AmazonEKS_CNI_Policy IAM policy的 IAM 角色。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。
- 如果使用是 `1.7.0` 版或更高版本的 CNI 插件，并且使用自定义 pod 安全策略，请参阅 the section called “删除默认策略” (p. 454)Pod 安全策略 (p. 453)。

添加 Amazon VPC CNI Amazon EKS 附加组件

您可以使用 `eksctl`、Amazon Web Services Management Console 或 Amazon CLI 将 Amazon VPC CNI Amazon EKS 附加组件添加到您的 `1.18` 或更高版本的集群。

> Important
>
> 在添加 Amazon VPC CNI Amazon EKS 附加组件之前，请确认您不会自行管理 Amazon EKS 将开始管理的任何设置。要确定 Amazon EKS 管理的设置，请参阅 Amazon EKS 附加组件配置 (p. 342)。

eksctl

使用 **`eksctl`** 添加推荐版本 (p. 233)的 Amazon EKS 附加组件

将 *`my-cluster`* 替换为您的集群名称，将 `arn:aws:iam::`*`111122223333`*`:role/`*`AmazonEKSVPCCNIRole`* 替换为现有的 IAM 角色（请参阅先决条件 (p. 234)）。

```
eksctl create addon \
    --name vpc-cni \
    --version v1.11.3-eksbuild.1 \
    --cluster my-cluster \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole \
    --force
```

如果任意 Amazon EKS 附加组件设置与自行管理附加组件的现有设置冲突，那么添加 Amazon EKS 附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。

如果您想添加其他版本的附加组件，则可以使用以下命令查看可用于附加组件的所有版本和集群的版本。将 *`1.23`* 替换为集群版本。

```
eksctl utils describe-addon-versions --name vpc-cni --kubernetes-version 1.23 | grep
 AddonVersion:
```

将 `create addon` 命令中的 `v1.11.3-eksbuild.1` 替换为您想要添加的输出中返回的版本，然后运行 `create addon` 命令。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 添加推荐版本 (p. 233)的 Amazon EKS 附加组件

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 在左侧导航窗格中，选择 Clusters（集群），然后选择要为其配置 Amazon VPC CNI plugin for Kubernetes 附加组件的集群的名称。
3. 选择 Add-ons（附加组件）选项卡。
4. 选择 Add new（新增）。
   - 对于 Name（名称），请选择 **vpc-cni**。
   - 选择您想使用的 Version（版本）。我们建议使用 **1.11.3-eksbuild.1** 版本，但如有必要，您可以选择另一个版本。
   - 对于 Service account role（服务账户角色），选择您附加了 AmazonEKS_CNI_Policy IAM policy 的 IAM 角色的名称（请参阅先决条件 (p. 234)）。
   - 选择 Override existing configuration for this add-on on the cluster.（覆盖集群上此附加组件的现有配置。）如果任意 Amazon EKS 附加组件设置与自行管理附加组件的现有设置冲突，那么添加 Amazon EKS 附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。
   - 选择 Add（添加）。

Amazon CLI

要使用 Amazon CLI 添加推荐版本 (p. 233)的 Amazon EKS 附加组件，请将 *my-cluster* 替换为您的集群名称，将 `arn:aws:iam::`*111122223333*`:role/`*AmazonEKSCNIRole* 替换为您已将 AmazonEKS_CNI_Policy IAM policy 添加到的 IAM 角色的 ARN（请参阅先决条件 (p. 234)），然后运行命令。

```
aws eks create-addon \
    --cluster-name my-cluster \
    --addon-name vpc-cni \
    --addon-version v1.11.3-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole \
    --resolve-conflicts OVERWRITE
```

如果任意 Amazon EKS 附加组件设置与自行管理附加组件的现有设置冲突，那么添加 Amazon EKS 附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。

如果您想添加其他版本的附加组件，则可以使用以下命令查看可用于附加组件的所有版本和集群的版本。将 *1.23* 替换为集群版本。

```
aws eks describe-addon-versions \
    --addon-name vpc-cni \
    --kubernetes-version 1.23 \
    --query "addons[].addonVersions[].[addonVersion, compatibilities[].Version]" \
    --output text
```

将 `create-addon` 命令中的 `v1.11.3-eksbuild.1` 替换为您想要添加的输出中返回的版本，然后运行 `create-addon` 命令。

更新 Amazon VPC CNI plugin for Kubernetes 附加组件

Important

在更新 Amazon VPC CNI plugin for Kubernetes 附加组件之前，请确认您不会自行管理 Amazon EKS 将会管理的任何设置。要确定 Amazon EKS 管理的设置，请参阅 Amazon EKS 附加组件配置 (p. 342)。

本程序用于更新 Amazon VPC CNI plugin for Kubernetes 附加组件。如果您尚未添加 Amazon VPC CNI plugin for Kubernetes 附加组件，请改为完成 更新 Amazon VPC CNI plugin for Kubernetes 自行管理的附加组件 (p. 238) 中的程序。当新版本发布之后或您将集群更新 (p. 28)到一个新的 Kubernetes 次要版本后，Amazon EKS 不会自动更新附加组件。要更新现有集群的附加组件，您必须启动更新，然后 Amazon EKS 会为您更新该附加组件。

建议您一次更新一个次要版本。例如，如果您的当前的次要版本为 `1.9`，并且您想要更新到 `1.11`，则应首先更新到最新的补丁版本 `1.10`，再更新到最新的补丁版本 `1.11`。

您可以使用 `eksctl`、Amazon Web Services Management Console 或 Amazon CLI 更新您的 `1.18` 或更高版本集群上的 Amazon VPC CNI plugin for Kubernetes 附加组件。

eksctl

使用 **`eksctl`** 将 Amazon EKS 附加组件更新为推荐版本 (p. 233)

1. 检查附加组件的当前版本。将 *`my-cluster`* 替换为您的集群名称。

```
eksctl get addon --name vpc-cni --cluster my-cluster
```

输出示例如下。

```
NAME    VERSION                 STATUS   ISSUES   IAMROLE
                              UPDATE AVAILABLE
vpc-cni v1.7.5-eksbuild.2       ACTIVE   0
 arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole      v1.11.3-eksbuild.1
```

2. 将附加组件更新为推荐版本 (p. 233)。

```
eksctl update addon \
    --name vpc-cni \
    --version 1.11.3-eksbuild.1 \
    --cluster my-cluster \
    --force
```

如果您想更新到其他版本的附加组件，则可以使用以下命令查看可用于附加组件的所有版本和集群的版本。将 *`1.23`* 替换为集群版本。

```
eksctl utils describe-addon-versions --name vpc-cni --kubernetes-version 1.23 |
 grep AddonVersion:
```

将 `update addon` 命令中的 `v1.11.3-eksbuild.1` 替换为您想要添加的输出中返回的版本，然后运行 `update addon` 命令。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 将 Amazon EKS 附加组件更新为推荐版本 (p. 233)

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。

2. 在左侧导航窗格中，选择 Clusters（集群），然后选择要为其更新 Amazon VPC CNI plugin for Kubernetes 附加组件的集群的名称。
3. 选择 Add-ons（附加组件）选项卡。
4. 选择 vpc-cni 框右上角中的框，然后选择 Edit（编辑）。
   - 选择您要使用的 Amazon EKS 附加组件的 Version（版本）。我们建议使用 **`1.11.3-eksbuild.1`** 版本，但如有必要，您可以选择另一个版本。
   - 对于 Service account role（服务账户角色），如果尚未进行选择，则选择您附加了 AmazonEKS_CNI_Policy IAM policy 的 IAM 角色的名称（请参阅先决条件 (p. 234)）。
   - 选择 Override existing configuration for this add-on on the cluster.（覆盖集群上此附加组件的现有配置。）
   - Select Update（更新）。

Amazon CLI

使用 Amazon CLI 将 Amazon EKS 附加组件更新为推荐版本 (p. 233)

1. 检查附加组件的当前版本。将 *my-cluster* 替换为您的集群名称。

```
aws eks describe-addon \
    --cluster-name my-cluster \
    --addon-name vpc-cni \
    --query "addon.addonVersion" \
    --output text
```

输出示例如下。

```
v1.7.5-eksbuild.2
```

返回的版本可能有所不同。

2. 将附加组件更新为最新的推荐版本。将 *my-cluster* 替换为您的集群名称。

```
aws eks update-addon \
    --cluster-name my-cluster \
    --addon-name vpc-cni \
    --addon-version v1.11.3-eksbuild.1 \
    --resolve-conflicts OVERWRITE
```

如果您想更新到其他版本的附加组件，则可以使用以下命令查看可用于附加组件的所有版本和集群的版本。将 *1.23* 替换为集群版本。

```
aws eks describe-addon-versions \
    --addon-name vpc-cni \
    --kubernetes-version 1.23 \
    --query "addons[].addonVersions[].[addonVersion, compatibilities[].Version]" \
    --output text
```

将 `update-addon` 命令中的 `v1.11.3-eksbuild.1` 替换为您想要添加的输出中返回的版本，然后运行 `update-addon` 命令。

删除 Amazon VPC CNI plugin for Kubernetes 附加组件

移除 Amazon EKS 附加组件时，您有两种选择：

- 在集群上保留附加组件的软件：此选项删除了 Amazon EKS 对任何设置和功能的管理，使 Amazon EKS 能够在您启动更新后向您发出更新通知并自动更新 Amazon EKS 附加组件，但会将附加组件的软件保留在您的集群上。此选项使附加组件成为自我管理的附加组件，而不是 Amazon EKS 附加组件。附加组件没有停机时间。
- 从集群中完全删除附加组件：只有在集群中没有资源依赖于附加组件提供的功能时，才应从集群中移除 Amazon EKS 附加组件。删除 Amazon EKS 附加组件后，如有需要，您可以再次添加它。

如果附加组件有与其关联的 IAM 账户，则不会移除该 IAM 账户。

您可以使用 `eksctl`、Amazon Web Services Management Console 或 Amazon CLI 从您的集群中删除 Amazon VPC CNI plugin for Kubernetes 附加组件。

eksctl

使用 **`eksctl`** 删除 Amazon EKS 附加组件

将 *`my-cluster`* 替换为集群名称，然后运行以下命令。移除 `--preserve` 会从您的集群中移除附加组件软件。

```
eksctl delete addon --cluster my-cluster --name vpc-cni --preserve
```

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 删除 Amazon EKS 附加组件

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择 Clusters（集群），然后选择要删除其 Amazon VPC CNI plugin for Kubernetes 附加组件的集群的名称。
3. 选择 Add-ons（附加组件）选项卡。
4. 选择 **`vpc-cni`** 框的右上角的复选框，然后选择 Remove（删除）。如果您希望 Amazon EKS 停止管理附加组件的设置，但希望在集群上保留附加组件，以便您可以自行管理附加组件的所有设置，请选择 Preserve on cluster（保留在集群上）。键入 **`vpc-cni`**，然后选择 Remove（删除）。

Amazon CLI

使用 Amazon CLI 删除 Amazon EKS 附加组件

将 *`my-cluster`* 替换为集群名称，然后运行以下命令。移除 `--preserve` 会从您的集群中移除附加组件软件。

```
aws eks delete-addon --cluster-name my-cluster --addon-name vpc-cni --preserve
```

更新 Amazon VPC CNI plugin for Kubernetes 自行管理的附加组件

如果您有一个集群，但尚未向其添加 Amazon VPC CNI plugin for Kubernetes 附加组件，或者需要自行管理附加组件，请完成以下步骤更新附加组件。如果您已添加了 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件，请改为按照 更新 Amazon VPC CNI plugin for Kubernetes 附加组件 (p. 236) 中的程序操作。

Important

- 版本指定为 `major-version.minor-version.patch-version`

- 您每次只更新一个次要版本。例如，如果您的当前次要版本为 1.9，并且您想要更新到 1.11，则应首先更新到 1.10，再更新到 1.11。
- 所有版本都适用于所有 Amazon EKS 支持的 Kubernetes 版本，但并非每个版本的所有功能都适用于所有 Kubernetes 版本。使用不同的 Amazon EKS 功能时，如果需要特定版本的附加组件，则会在功能文档中注明。
- 建议您更新为版本 1.11.3，尽管您可以在必要时更新到任何发布版本。

要更新自行管理的附加组件

1. 查看 GitHub 上的 releases，以查看可用版本并熟悉要更新到的版本中的更改。
2. 使用以下命令确定您集群的当前 Amazon VPC CNI plugin for Kubernetes 附加组件版本：

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/" -f 2
```

输出示例如下。

```
amazon-k8s-cni-init:v1.7.5-eksbuild.1
amazon-k8s-cni:v1.7.5-eksbuild.1
```

您的输出可能看起来与示例输出不同。在此示例输出中，Amazon VPC CNI plugin for Kubernetes 附加组件版本为 1.7.5-eksbuild.1。当前建议的版本为 1.11.3。Amazon EKS 最初与集群一起部署的版本与之前的输出类似。但是，如果您已经使用清单至少更新了一次附加组件，则输出可能不包括 -eksbuild.1。

3. 使用 Helm V3 (p. 384) 或更高版本或通过使用清单更新 DaemonSet。

Helm

a. 将 eks-charts 存储库添加到 Helm。

```
helm repo add eks https://aws.github.io/eks-charts
```

b. 更新您的本地存储库，以确保您拥有最新的图表。

```
helm repo update
```

c. 备份当前设置，以便在后续步骤中确定需要为哪些设置指定值。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

d. 如果您使用 Helm 安装了现有的 Amazon VPC CNI plugin for Kubernetes DaemonSet，则跳至下一步骤。

完成以下选项之一，以便 Helm 可以管理 DaemonSet 资源：

- 将 Helm 注释和标签添加到现有资源中。
  i. 将以下内容复制到您的设备。如果您想要使用另一个发布名称，请替换 *aws-vpc-cni*。运行该命令以创建 helm-cni.sh 文件。

```
cat >helm-cni.sh <<EOF
#!/usr/bin/env bash

set -euo pipefail

for kind in daemonSet clusterRole clusterRoleBinding serviceAccount; do
  echo "setting annotations and labels on $kind/aws-node"
```

```
  kubectl -n kube-system annotate --overwrite $kind aws-node meta.helm.sh/
release-name=aws-vpc-cni
  kubectl -n kube-system annotate --overwrite $kind aws-node meta.helm.sh/
release-namespace=kube-system
  kubectl -n kube-system label --overwrite $kind aws-node app.kubernetes.io/
managed-by=Helm
done
EOF
```

ii. 使脚本成为可执行文件。

```
chmod +x helm-cni.sh
```

iii. 运行脚本

```
./helm-cni.sh
```

- 删除现有的 `DaemonSet` 资源。

  Important

  从完成此步骤到下一步之间，您的集群将遇到停机时间。

```
kubectl delete serviceaccount aws-node -n kube-system
kubectl delete customresourcedefinition eniconfigs.crd.k8s.amazonaws.com
kubectl delete clusterrole aws-node
kubectl delete clusterrolebinding aws-node
kubectl delete daemonset aws-node -n kube-system
```

e. 使用以下选项之一安装图表。在运行安装之前，请检查您在上一步中对 `DaemonSet` 的设置所做的备份，然后查看配置设置以确定是否需要设置其中任何一个。

如果您有现有的 IAM 角色可用于 `DaemonSet`，则在后面的安装选项的结尾添加以下行。如果您没有关联 `aws-node` Kubernetes 服务账户的 IAM 角色，那么我们建议创建一个。将 *111122223333* 替换为您的账户 ID，并将 *AmazonEKSVPCCNIRole* 替换为您的角色名称。如需创建角色，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。

```
--set serviceAccount.annotations."eks\.amazonaws\.com\.cn/role-
arn"=arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
```

如果您在上一步中添加了 Helm 注释和标签，则将以下设置添加到以下任何选项中。

```
--set originalMatchLabels=true
--set crd.create=false
```

- 如果您的节点有权访问 Amazon EKS Amazon ECR 存储库，则用发布名称 `aws-vpc-cni` 和安装图表。请将 *918309763551* 替换为您的集群所在的 Amazon Web Services 区域 的 the section called “Amazon 容器镜像注册表” (p. 340) 中的值。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

```
helm upgrade -i aws-vpc-cni eks/aws-vpc-cni \
--namespace kube-system \
--set image.account=918309763551 \
--set image.region=region-code \
--set image.domain=amazonaws.com.cn \
--set image.tag=v1.11.3 \
--set init.image.account=918309763551
--set init.image.region=region-code \
```

```
--set init.image.domain=amazonaws.com.cn \
--set init.image.tag=v1.11.3
```

- 如果您的节点无法访问 Amazon EKS Amazon ECR 存储库

  1. 请提取以下容器镜像并将其推送到节点有权访问的存储库。有关如何提取、标记和推送镜像到您自己的存储库的更多信息，请参阅 将容器镜像从一个存储库复制到另一个存储库 (p. 338)。我们建议在以下命令中使用该版本，但是如果有必要，您可以将它替换为任何发布版本。将 *918309763551* 和 *region-code* 替换为您的集群所在的 Amazon Web Services 区域 的 the section called “Amazon 容器镜像注册表” (p. 340) 中的值。

     ```
     918309763551.dkr.ecr.region-code.amazonaws.com.cn/amazon-k8s-cni-init:v1.11.3
     918309763551.dkr.ecr.region-code.amazonaws.com.cn/amazon-k8s-cni:v1.11.3
     ```

  2. 使用发布名称 aws-vpc-cni 和默认配置安装图表。在运行安装之前，请检查您在上一步中对 DaemonSet 的设置所做的备份，然后查看配置设置以确定是否需要设置其中任何一个。将 *registry*/*repo*:*tag* 替换为您的注册表、存储库和标签。

     ```
     helm upgrade -i aws-vpc-cni eks/aws-vpc-cni \
         --namespace kube-system \
         --set image.override=registry/repo:tag \
         --set init.image.override=registry/repo:tag
     ```

Manifest

a. 如果您更改了当前 Amazon VPC CNI plugin for Kubernetes DaemonSet 的任何默认设置，或者您需要从自己的存储库中提取容器镜像才能更新 DaemonSet，或者您的集群位于 us-west-2 以外的区域，或者您需要更新到版本 1.7 或更早版本的特定修补程序版本，则跳到下一步。

   运行以下命令以更新您的 Amazon VPC CNI plugin for Kubernetes 附加组件。您可以将 *1.11.3* 更改为 1.7.0 或更高版本。无论您为 1.7 指定了什么修补程序版本，例如 1.7.5，都将提取镜像的最新修补程序版本（1.7.10）。

   ```
   kubectl apply -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/
   v1.11.3/config/master/aws-k8s-cni.yaml
   ```

   如果您需要更新到早于 1.7.0 的版本，则使用下面的 URL 提取清单。如有必要，您可以将 *1.6* 更改为更早的版本。清单会为您指定的版本提取镜像的最新修补程序版本。

   ```
   kubectl apply -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/
   release-1.6/config/v1.6/aws-k8s-cni.yaml
   ```

   跳至查看 DaemonSet 的状态 (p. 243)步骤。

b. 如果您的节点有权访问 Amazon EKS Amazon ECR 镜像存储库，请跳到下一步。

   请提取以下容器镜像并将其推送到节点有权访问的存储库。有关如何提取、标记和推送镜像到您自己的存储库的更多信息，请参阅 将容器镜像从一个存储库复制到另一个存储库 (p. 338)。我们建议在以下命令中使用该版本，但是如果有必要，您可以将它替换为任何发布版本。将 *918309763551* 和 *region-code* 替换为您的集群所在的 Amazon Web Services 区域 的 the section called “Amazon 容器镜像注册表” (p. 340) 中的值。

   ```
   918309763551.dkr.ecr.region-code.amazonaws.com.cn/amazon-k8s-cni-init:v1.11.3
   918309763551.dkr.ecr.region-code.amazonaws.com.cn/amazon-k8s-cni:v1.11.3
   ```

c. 如果您尚未更改 DaemonSet 的任何默认设置，请跳到下一步。

   备份当前设置，以便您可以将设置与新清单中的默认设置进行比较。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

d. 下载 Amazon VPC CNI plugin for Kubernetes 附加组件的清单。您可以将 *1.11.3* 更改为 1.7.0 或更高版本。无论您为 1.7 指定了什么修补程序版本，例如 1.7.5，都将提取镜像的最新修补程序版本（1.7.10）。

```
curl -o aws-k8s-cni.yaml https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.11.3/config/master/aws-k8s-cni.yaml
```

如果您需要更新到早于 1.7.0 的版本，则使用下面的 URL 提取清单。如有必要，您可以将 *1.6* 更改为更早的版本。清单会为您指定的版本提取镜像的最新修补程序版本。

```
curl -o aws-k8s-cni.yaml https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/release-1.6/config/v1.6/aws-k8s-cni.yaml
```

如果您需要 1.7 或更早版本的特定修补程序，请在文本编辑器中打开文件，并将以下两行中的 v*1.11.3* 更改为您想要的特定修补程序版本。根据您下载的文件版本，v*1.11.3* 可能是不同的版本号，也可能是 latest。进行更改后，保存文件。

```
image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/amazon-k8s-cni-init:1.11.3"
image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/amazon-k8s-cni:v1.11.3"
```

e. 如果您在上一步中没有将容器镜像复制到您自己的存储库，则跳至下一个步骤。

将文件中的注册表、存储库和标签替换为您自己的。

i. 请将以下命令中的 *your-registry* 替换为您的注册表，然后运行修改后的命令以替换文件中的 602401143452.dkr.ecr.us-west-2.amazonaws.com。

```
sed -i.bak -e 's|602401143452.dkr.ecr.us-west-2.amazonaws.com|your-registry|' aws-k8s-cni.yaml
```

ii. 将以下命令中的 *your-repository* 和 *tag* 替换为您的存储库和标签，然后运行修改后的命令以替换文件中的 amazon-k8s-cni-init:v1.11.3。将 *1.11.3* 替换为您下载的清单的版本。

```
sed -i.bak -e 's|amazon-k8s-cni-init:v1.11.3|your-repository:tag|' aws-k8s-cni.yaml
```

iii. 将以下命令中的 *your-repository* 和 *tag* 替换为您的存储库和标签，然后运行修改后的命令以替换文件中的 amazon-k8s-cni:v1.11.3。将 *1.11.3* 替换为您下载的清单的版本。

```
sed -i.bak -e 's|amazon-k8s-cni:v1.11.3|your-repository:tag|' aws-k8s-cni.yaml
```

iv. 跳至比较设置 (p. 243)步骤。

f. 运行以下命令，以将文件中的信息替换为您的集群所在的 Amazon Web Services 区域 的信息。

i. 运行以下命令将文件中的 us-west-2 替换为您的 Amazon Web Services 区域。

```
sed -i.bak -e 's|us-west-2|cn-northwest-1|' aws-k8s-cni.yaml
```

ii. 运行以下命令以将文件中的 602401143452 替换为您的集群所在的 Amazon Web Services 区域 的镜像的账户。

```
sed -i.bak -e 's|602401143452|961992271922|' aws-k8s-cni.yaml
```

iii. 运行以下命令以将 amazonaws.com 替换为您的集群所在的 Amazon Web Services 区域 的域。

```
sed -i.bak -e 's|amazonaws.com|amazonaws.com.cn|' aws-k8s-cni.yaml
```

g. 如果您更改了当前 Amazon VPC CNI plugin for Kubernetes DaemonSet 的任何默认设置，则将新清单中的设置与您在上一步中创建的备份文件进行比较。

```
diff aws-k8s-cni.yaml aws-k8s-cni-old.yaml -u
```

编辑新的清单文件并更改任何设置值，以便它们与备份文件中的设置匹配。

h. 将清单文件应用于集群。

```
kubectl apply -f aws-k8s-cni.yaml
```

4. 查看 DaemonSet 的状态。

```
kubectl get daemonset aws-node -n kube-system
```

输出示例如下。

```
NAME       DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
aws-node   2         2         2       2            2           <none>          4h39m
```

一旦 READY、UP-TO-DATE 和 AVAILABLE 列中的数字相同，您的更新即完成。您的数字可能与前一个输出中的数字不同。

5. 查看 DaemonSet 以确认您所做的更改。

```
kubectl get daemonset aws-node -n kube-system -o yaml
```

配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户

Amazon VPC CNI plugin for Kubernetes 是 Amazon EKS 集群中用于 pod 联网的联网插件。该插件负责向 Kubernetes 节点分配 VPC IP 地址并为每个节点上的 pods 配置所需网络。该插件：

- 需要 Amazon Identity and Access Management (IAM) 权限。则权限在 AmazonEKS_CNI_Policy Amazon 托管策略中指定。您可以将该策略附加到 Amazon EKS 节点 IAM 角色 (p. 427) 或单独 IAM 角色。我们建议您按照本主题中的详细说明，将其分配给一个单独的角色。
- 在部署时创建并配置为使用名为 aws-node 的 Kubernetes 服务账户。服务账户绑定到被分配了所需的 Kubernetes 权限的名为 aws-node 的 Kubernetes clusterrole。

Note

Amazon VPC CNI plugin for Kubernetes 的 pods 有权访问分配给 Amazon EKS 节点 IAM 角色 (p. 427) 的权限，除非您阻止对 IMDS 的访问。有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

先决条件

- 现有 版 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS (p. 4)。

- 集群的现有 Amazon Identity and Access Management IAM OpenID Connect (OIDC) 提供程序。要确定您是否已经拥有一个或是否要创建一个，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。

步骤 1：创建 Amazon VPC CNI plugin for Kubernetes IAM 角色

创建 IAM 角色

1. 创建 IAM 角色。您可以使用 `eksctl` 或 `kubectl` 和 Amazon CLI 以创建 IAM 角色。

   eksctl

   使用与您的集群的 IP 系列匹配的命令创建 IAM 角色并将 IAM policy 附加到该角色。此命令创建并部署一个创建 IAM 角色的 Amazon CloudFormation 堆栈，向其附加您指定的策略，并使用所创建 IAM 角色的 ARN 对现有 `aws-node` Kubernetes 服务账户添加注释。

   - 将 *my-cluster* 替换为您自己的值。

   ```
   eksctl create iamserviceaccount \
       --name aws-node \
       --namespace kube-system \
       --cluster my-cluster \
       --role-name "AmazonEKSVPCCNIRole" \
       --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
       --override-existing-serviceaccounts \
       --approve
   ```

   kubectl and the Amazon CLI

   1. 查看集群的 OIDC 提供商 URL。

      ```
      aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer"
       --output text
      ```

      输出示例如下。

      ```
      https://oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE
      ```

      如果没有返回输出，则您必须为集群创建 IAM OIDC 提供程序 (p. 396)。

   2. 将以下内容复制到名为 *vpc-cni-trust-policy.json* 的文件中。将 *111122223333* 替换为您的账户 ID，并将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 替换为上一步骤中返回的输出。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

      ```
      {
          "Version": "2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Federated": "arn:aws:iam::111122223333:oidc-provider/
      oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
                  },
                  "Action": "sts:AssumeRoleWithWebIdentity",
                  "Condition": {
                      "StringEquals": {
                          "oidc.eks.region-code.amazonaws.com.cn/
      id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
      ```

```
                    "oidc.eks.region-code.amazonaws.com.cn/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
node"
                }
            }
        }
    ]
}
```

3. 创建角色。您可以将 *AmazonEKSVPCCNIRole* 替换为您选择的任何名称。

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

4. 将所需的 IAM policy 附加到角色。

   - IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSVPCCNIRole
```

5. 运行以下命令以使用您以前创建的 IAM 角色的 ARN 对 aws-node 服务账户添加注释。将 *example values* 替换为您自己的值。

```
kubectl annotate serviceaccount \
    -n kube-system aws-node \
    eks.amazonaws.com/role-arn=arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
```

2. （可选）配置您的 Kubernetes 服务账户使用的 Amazon Security Token Service 端点类型。有关更多信息，请参阅配置服务账户的 Amazon Security Token Service 端点 (p. 405)。

步骤 2：重新部署 Amazon VPC CNI plugin for Kubernetes 容器组（pod）

1. 删除并重新创建任何与服务账户关联的现有 pods，以应用凭证环境变量。注释未应用于目前在没有注释的情况下运行的 pods。以下命令删除现有的 aws-node DaemonSet pods 并使用服务账户注释部署它们。

```
kubectl delete pods -n kube-system -l k8s-app=aws-node
```

2. 确认 pods 已全部重新启动。

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. 描述 pods 之一并确保 AWS_WEB_IDENTITY_TOKEN_FILE 和 AWS_ROLE_ARN 环境变量存在。将 *9rgzw* 替换为上一步输出中返回的其中一个 pods 的名称。

```
kubectl exec -n kube-system aws-node-9rgzw -c aws-node -- env | grep AWS
```

输出示例如下。

```
...
AWS_WEB_IDENTITY_TOKEN_FILE=/var/run/secrets/eks.amazonaws.com/serviceaccount/token
...
AWS_ROLE_ARN=arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
```

```
...
```

如果您的 pod 正在使用Amazon Web Services 区域端点，之前的输出中也将返回以下行。

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

步骤 3：从节点 IAM 角色中删除 CNI 策略

如果您的 Amazon EKS 节点 IAM 角色 (p. 427) 当前附加有 AmazonEKS_CNI_Policy IAM policy ，而且您已另行创建了一个 IAM 角色，并已将该策略附加到该 IAM 角色并将其分配给了 aws-node Kubernetes 服务账户，那么我们建议您使用与您的集群的 IP 系列匹配的 Amazon CLI 命令从节点角色中删除该策略。将 *AmazonEKSNodeRole* 替换为您的节点角色的名称。

- IPv4

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn
 arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

为使用 IPv6 系列的集群创建 IAM policy

Important

IPv6 在此 Amazon Web Services 区域 中不可用。

如果您创建了使用 IPv6 系列的 1.21 版或更高版本的集群，并且集群配置了 1.10.1 版或更高版本的 Amazon VPC CNI plugin for Kubernetes 附加组件，则需要创建一个 IAM policy，您可以将其分配给 IAM 角色。如果您现有的 1.21 版或更高版本的集群在创建时没有使用 IPv6 系列进行配置，则必须创建一个新集群才能使用 IPv6。有关集群使用 IPv6 的详细信息，请参阅 the section called “IPv6” (p. 248)。

1. 复制以下文本并将其保存到名为 *vpc-cni-ipv6-policy*.json 的文件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. 创建 IAM policy。

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document file://
vpc-cni-ipv6-policy.json
```

选择 pod 联网使用案例

Amazon VPC CNI 插件为 pods 提供了联网。下表可帮助您了解哪些联网使用案例可以共同使用，以及可用于不同 Amazon EKS 节点类型的功能和 Amazon VPC CNI 插件设置。表中的所有信息仅适用于 Linux `IPv4` 节点。

Amazon EKS 节点类型 (p. 84)	Amazon EC2			Fargate
教程：自定义联网 (p. 258)：分配来自不同于节点子网的子网的 IP 地址	是	是	是	是（通过 Fargate 配置文件控制的子网）
适用于 pods 的 SNAT (p. 258)	是（默认值为 `false`）	是（默认值为 `false`）	是（仅限 `true`）	是（仅限 `true`）
功能				
安全组范围	节点	节点	Pod	Pod
Amazon VPC 子网类型	公有和私有	公有和私有	仅限私有	仅限私有
网络策略 (p. 306) (Calico)	兼容	兼容	兼容 仅版本 `1.11.0` 或更高版本的 Amazon VPC 附加组件配置有 `POD_SECURITY_GROUP_ENFORCING_MODE=standard`	不支持
每个节点的容器组密度	中	高	低	第一
容器组启动时间	较好	最佳	好	中
Amazon VPC CNI 插件设置（有关各项设置的更多信息，请参阅 GitHub 上的 amazon-vpc-cni-k8s）				
`WARM_ENI_TARGET`	是	不适用	不适用	不适用
`WARM_IP_TARGET`	是	是	不适用	不适用
`MINIMUM_IP_TARGET`	是	是	不适用	不适用
`WARM_PREFIX_TARGET`	不适用	是	不适用	不适用

Note

- 流向和来自具有关联安全组的 pods 的流量不受 Calico 网络策略执行限制，并且仅受限于 Amazon VPC 安全组执行。

- IP 前缀和 IP 地址与标准 Amazon EC2 弹性网络接口相关联。需要特定安全组的容器组被分配给分支网络接口的主 IP 地址。您可以将获取 IP 地址或从 IP 前缀获取 IP 地址的 pods 与在同一节点上获取分支网络接口的 pods 混合使用。

Windows 个节点

每个 Windows 节点只支持 pods 的一个网络接口和辅助 `IPv4` 地址。因此，您无法将 IP 地址前缀或 `IPv6` 用于 Windows 节点。每个节点的最大 pods 数等于您可以分配给每个弹性网络接口的 IP 地址数减 1。Windows 支持 Calico 网络策略。有关更多信息，请参阅 Amazon EKS 上 Windows 容器的开源 Calico。您不能在 Windows 上使用 pods 的安全组 (p. 273)。

教程：将 `IPv6` 地址分配给 pods 和 services

> **Important**
>
> 此功能在您所在的 Amazon Web Services 区域不可用。

默认情况下，Kubernetes 将 `IPv4` 地址分配给您的 pods 和 services。您可以将集群配置为向其分配 `IPv6` 地址，而不是将 `IPv4` 地址分配给 pods 和 services。Amazon EKS 不支持双堆叠 pods 或 services，即使 Kubernetes 的 `1.23` 版本及更高版本中支持此类堆叠也是如此。因此，您无法将 `IPv4` 和 `IPv6` 地址同时分配给 pods 和 services。

创建集群时，您可以选择要用于集群的 IP 系列。集群在创建之后无法更改系列。

集群使用 **`IPv6`** 系列的注意事项：

- 您必须创建 `1.21` 版或更高版本的新集群，并指定该集群要使用 `IPv6` 系列。您无法为从前一版本更新的集群启用 `IPv6` 系列。有关如何创建新集群的说明，请参阅创建 Amazon EKS 集群 (p. 21)。
- 您部署到集群的 Amazon VPC CNI 附加组件的版本必须为 `1.10.1` 版或更高版本。默认情况下，此版本或更高版本随新的 `1.21` 版或更高版本的集群一起部署。部署附加组件后，您无法在不先删除集群所有节点组中的所有节点的情况下将 Amazon VPC CNI 附加组件版本降级到 `1.10.1` 版以下。
- 不支持 Windows pods 和 services。
- 如果您使用 Amazon EC2 节点，则必须使用 IP 前缀委派和 `IPv6` 配置 Amazon VPC CNI 附加组件。如果您在创建集群时选择 `IPv6` 系列，则 `1.10.1` 版的附加组件默认采用此配置。自行管理或 Amazon EKS 附加组件均为属于此种情况。有关 IP 前缀委派的更多信息，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量 (p. 270)。
- 创建集群时，您指定的 VPC 和子网必须具有分配给您指定的 VPC 和子网的 `IPv6` CIDR 块。您还必须为其分配一个 `IPv4` CIDR 块。这是因为，即使您只想使用 `IPv6`，VPC 仍然需要 `IPv4` CIDR 块才能正常工作。有关更多信息，请参阅《Amazon VPC 用户指南》中的将 `IPv6` CIDR 块与您的 VPC 关联。
- 创建集群和节点时，必须将配置指定为自动分配 `IPv6` 地址的子网。否则，您无法部署集群和节点。默认情况下，将禁用此配置。有关更多信息，请参阅《Amazon VPC 用户指南》中的修改子网的 `IPv6` 寻址属性。
- 分配给子网的路由表必须有 `IPv6` 地址的路由。有关更多信息，请参阅《Amazon VPC 用户指南》中的迁移到 `IPv6`。
- 您的安全组必须允许 `IPv6` 地址。有关更多信息，请参阅《Amazon VPC 用户指南》中的迁移到 `IPv6`。
- 您只能将 `IPv6` 与 Amazon 基于 Nitro 的 Amazon EC2 或 Fargate 节点一起使用。
- 您无法将 `IPv6` 与 pods 的安全组 (p. 273) 及 Amazon EC2 节点一起使用。但是，您可以将其与 Fargate 节点一起使用。如果您需要为各个 pods 使用单独的安全组，请继续将 `IPv4` 系列与 Amazon EC2 节点一起使用，或者改用 Fargate 节点。
- 如果您以前使用过自定义联网 (p. 258)帮助缓解 IP 地址耗尽，您可以改用 `IPv6`。您不能将自定义联网与 `IPv6` 一起使用。如果您使用自定义联网进行网络隔离，则可能需要为集群继续使用自定义联网和 `IPv4` 系列。
- 您不能将 `IPv6` 与部署在 Amazon Outposts 上的节点一起使用。

- Pods 和 services 只分配了 IPv6 地址。系统不会为其分配 IPv4 地址。由于 pods 能够通过实例本身上的 NAT 与 IPv4 端点通信，因此不需要 DNS64 和 NAT64。如果流量需要公有 IP 地址，则流量将源网络地址转换为公有 IP。
- 在 VPC 之外通信时，pod 的源 IPv6 地址并非转换为节点 IPv6 地址的源网络地址。它使用互联网网关或仅出口互联网网关进行路由。
- 所有节点均已分配 IPv4 和 IPv6 地址。
- 不支持 Amazon FSx for Lustre CSI 驱动程序 (p. 220)。
- 您可以使用 2.3.1 版或更高版本的 Amazon 负载均衡器控制器在 IP 模式（而不是实例模式）下对应用程序 (p. 332) 或到 IPv6 pods 的网络 (p. 327)流量进行负载均衡。有关更多信息，请参阅安装 Amazon Load Balancer Controller 附加组件 (p. 287)。
- 您必须将 IPv6 IAM policy 附加到节点 IAM 或 CNI IAM 角色。在两者之间，我们建议您将其附加到 CNI IAM 角色。有关更多信息，请参阅 为使用 IPv6 系列的集群创建 IAM policy (p. 246) 和 步骤 1：创建 Amazon VPC CNI plugin for Kubernetes IAM 角色 (p. 244)。
- 每个 pod 容器都会收到来自为其部署的子网指定的 CIDR 的 IPv6 地址。运行 Fargate pods 的底层硬件单元从分配给部署硬件单元的子网的 CIDR 获取唯一的 IPv4 和 IPv6 地址。
- 我们建议您在部署 IPv6 集群之前对集成的应用程序、Amazon EKS 附加组件和 Amazon 服务进行全面的评估。这是为了确保 IPv6 的所有功能按预期运行。
- 您不能将 IPv6 与 Amazon App Mesh 一起使用。
- Amazon EKS 不支持使用 Amazon EC2 实例元数据服务 IPv6 端点。
- 在集群中创建使用 IPv6 系列的自行管理节点组时，用户数据必须包含在节点启动时运行的 bootstrap.sh 文件的以下 BootstrapArguments。使用您集群 VPC 的 IPv6 CIDR 范围替换 *your-cidr*。

```
--ip-family ipv6 --service-ipv6-cidr your-cidr
```

如果您不知道集群的 IPv6 CIDR 范围，可以使用以下命令查看（需要 Amazon CLI 的 2.4.9 版本或更高版本）。

```
aws eks describe-cluster --name my-cluster --query
 cluster.kubernetesNetworkConfig.serviceIpv6Cidr --output text
```

部署 IPv6 集群和 Amazon Linux 托管节点

在本教程中，您将部署 IPv6 Amazon VPC、具有 IPv6 系列的 Amazon EKS 集群以及具有 Amazon EC2 Amazon Linux 节点的托管节点组。您无法在 IPv6 集群中部署 Amazon EC2 Windows 节点。您还可以将 Fargate 节点部署到集群，但为了简单起见，本主题中没有提供这些说明。

创建用于生产用途的集群前，我们建议您熟悉所有设置，并使用符合您要求的设置部署集群。有关更多信息，请参阅 创建 Amazon EKS 集群 (p. 21)、托管节点组 (p. 87) 和本主题中的注意事项 (p. 248)。一些设置仅在创建集群时才能启用。

先决条件

在开始使用本教程之前，您必须安装并配置创建和管理 Amazon EKS 集群所需的以下工具和资源。

- 您的设备或 Amazon CloudShell 上安装了 kubectl 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 1.22，则可以将 kubectl 版本 1.21、1.22 或 1.23 用于它。要安装或升级 kubectl，请参阅 安装或更新 kubectl (p. 4)。对于本教程，版本必须为 1.21 或更高版本。
- 您正在使用的 IAM 安全主体必须具有使用 Amazon EKS IAM 角色和服务相关角色的权限、Amazon CloudFormation 以及 VPC 和相关资源。有关更多信息，请参阅 IAM 用户指南中的用于 Amazon Elastic Kubernetes Service 的操作、资源和条件密钥和使用服务相关角色。

我们提供使用 `eksctl` 或 Amazon CLI 创建资源的程序。您还可以使用 Amazon Web Services Management Console 部署资源。但为了简单起见，本主题中没有提供这些说明。

eksctl

先决条件

您的计算机上安装了 `eksctl` 版本 `0.110.0` 或更高版本。要安装或更新到该版本，请参阅 the section called “安装 eksctl” (p. 10)。

使用 **`eksctl`** 部署 **IPv6** 集群

1. 将以下内容复制到您的设备。将 *example values* 替换为您自己的值。将 *region-code* 替换为 Amazon EKS 支持的任一 Amazon Web Services 区域。有关 Amazon Web Services 区域列表，请参阅 Amazon 一般参考中的 Amazon EKS 端点和配额。您可以将 *t3.medium* 替换为任何 Amazon Nitro 系统实例类型。`version` 的值必须为 *1.21* 或更高版本的支持的 Amazon EKS Kubernetes 版本 (p. 59)。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。替换值后，运行修改后的命令可创建 `ipv6-cluster.yaml` 文件。

```
cat >ipv6-cluster.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "1.21"

kubernetesNetworkConfig:
  ipFamily: IPv6

addons:
  - name: vpc-cni
    version: latest
  - name: coredns
    version: latest
  - name: kube-proxy
    version: latest

iam:
  withOIDC: true

managedNodeGroups:
  - name: my-nodegroup
    instanceType: t3.medium
EOF
```

2. 创建您的集群。

```
eksctl create cluster -f ipv6-cluster.yaml
```

创建集群需要几分钟时间。在看到输出的最后一行（与以下输出类似）之前，不要继续操作。

```
...
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

3. 确认默认 pods 已分配 `IPv6` 地址。

```
kubectl get pods -n kube-system -o wide
```

输出示例如下。

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                                NODE                                                NOMINATED
 NODE   READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

4. 确认默认服务已分配 IPv6 地址。

```
kubectl get services -n kube-system -o wide
```

输出示例如下。

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

5. （可选）部署示例应用程序 (p. 315)或部署 Amazon Load Balancer Controller (p. 287) 和示例应用程序以对应用程序 (p. 332)或至 IPv6 pods 的网络 (p. 327)流量进行负载均衡。
6. 在使用完成针对本教程而创建的集群和节点后，应使用以下命令清理创建的资源。

```
eksctl delete cluster my-cluster
```

Amazon CLI

先决条件

您的设备或 Amazon CloudShell 上安装并配置了 2.7.21 版或更高版本，或 1.25.46 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 aws configure 进行快速配置。如果您使用 Amazon CloudShell，则可能需要安装版本 2.7.21 或更高版本或者 1.25.46 版或更高版本的 Amazon CLI，因为 Amazon CloudShell 中安装的默认 Amazon CLI 版本可能是较早的版本。

Important

- 您必须以同一用户身份完成此程序中的所有步骤。
- 您必须在同一 shell 中完成此程序中的所有步骤。有几个步骤使用了之前步骤中设置的变量。如果在不同的 shell 中设置变量值，则使用这些变量的步骤将无法正常运行。如果使用 Amazon CloudShell 完成以下程序，请记住，如果您在大约 20–30 分钟内没有使用键盘或指针与其交互，则 shell 会话将结束。正在运行的进程不算作交互。

- 这些说明是针对 Bash shell 编写的，在其他 shell 中可能需要调整。

使用 Amazon CLI 创建集群

将此程序步骤中的所有*###*替换为您自己的值。

1. 运行以下命令以设置在后面的步骤中使用的一些变量。将 *region-code* 替换为您想要在其中部署资源的 Amazon Web Services 区域。该值可以是 Amazon EKS 支持的任何 Amazon Web Services 区域。有关 Amazon Web Services 区域列表，请参阅 Amazon 一般参考中的 Amazon EKS 端点和配额。将 *my-cluster* 替换为您的集群名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。将 *my-nodegroup* 替换为您的节点组的名称。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 100 个字符。请将 *111122223333* 替换为您的账户 ID。

   ```
   export region_code=region-code
   export cluster_name=my-cluster
   export nodegroup_name=my-nodegroup
   export account_id=111122223333
   ```

2. 创建具有公有和私有子网且符合 Amazon EKS 和 `IPv6` 要求的 Amazon VPC。

   a. 运行以下命令设置 Amazon CloudFormation 堆栈名称的变量。您可以将 *my-eks-ipv6-vpc* 替换为您选择的任何名称。

      ```
      export vpc_stack_name=my-eks-ipv6-vpc
      ```

   b. 使用 Amazon CloudFormation 模板创建 `IPv6` VPC。

      ```
      aws cloudformation create-stack --region $region_code --stack-name
       $vpc_stack_name \
        --template-url https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/
      cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
      ```

      堆栈需要几分钟的时间来创建。运行以下命令。在命令的输出变成 `CREATE_COMPLETE` 之前，请勿继续执行下一步。

      ```
      aws cloudformation describe-stacks --region $region_code --stack-name
       $vpc_stack_name --query Stacks[].StackStatus --output text
      ```

   c. 检索创建的公有子网的 ID。

      ```
      aws cloudformation describe-stacks --region $region_code --stack-name
       $vpc_stack_name \
          --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --
      output text
      ```

      输出示例如下。

      ```
      subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE
      ```

   d. 为创建的公有子网启用自动分配 `IPv6` 地址选项。

      ```
      aws ec2 modify-subnet-attribute --region $region_code --
      subnet-id subnet-0a1a56c486EXAMPLE --assign-ipv6-address-on-
      creation
      aws ec2 modify-subnet-attribute --region $region_code --subnet-id
       subnet-099e6ca77aEXAMPLE --assign-ipv6-address-on-creation
      ```

e. 从部署的 Amazon CloudFormation 堆栈中检索模板创建的子网和安全组的名称，将其存储在变量中，以供后续步骤使用。

```
security_groups=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SecurityGroups`].OutputValue' --
output text)

public_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --
output text)

private_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPrivate`].OutputValue' --
output text)

subnets=${public_subnets},${private_subnets}
```

3. 创建集群 IAM 角色并向其附加所需的 Amazon EKS IAM 托管策略。Amazon EKS 托管的 Kubernetes 集群会代表您调用其他 Amazon 服务，以管理您用于该服务的资源。

   a. 运行以下命令以创建 `eks-cluster-role-trust-policy.json` 文件。

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

   b. 运行以下命令设置角色名称的变量。您可以将 *myAmazonEKSClusterRole* 替换为您选择的任何名称。

```
export cluster_role_name=myAmazonEKSClusterRole
```

   c. 创建角色。

```
aws iam create-role --role-name $cluster_role_name --assume-role-policy-
document file://"eks-cluster-role-trust-policy.json"
```

   d. 检索 IAM 角色的 ARN 并将其存储在变量中以供后续步骤使用。

```
cluster_iam_role=$(aws iam get-role --role-name $cluster_role_name --
query="Role.Arn" --output text)
```

   e. 将所需的 Amazon EKS 托管 IAM policy 附加到角色。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

4. 创建您的集群。

a. 版本必须为 *1.21* 版或更高版本的支持的 Amazon EKS Kubernetes 版本 (p. 59)。

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.21 \
   --role-arn $cluster_iam_role --resources-vpc-config subnetIds=
$subnets,securityGroupIds=$security_groups \
   --kubernetes-network-config ipFamily=ipv6
```

Note

您可能会收到一个错误，指示请求中的可用区之一没有足够容量来创建 Amazon EKS 集群。如果发生这种情况，错误输出将包含可支持新集群的可用区。再次尝试使用至少两个位于您账户中支持的可用区的子网创建集群。有关更多信息，请参阅容量不足 (p. 478)。

b. 创建集群需要几分钟时间。运行以下命令。在命令的输出变成 `ACTIVE` 之前，请勿继续执行下一步。

```
aws eks describe-cluster --region $region_code --name $cluster_name --query
 cluster.status
```

5. 为集群创建或更新 `kubeconfig` 文件，以便您可以与集群通信。

```
aws eks update-kubeconfig --region $region_code --name $cluster_name
```

预设情况下，`config` 文件创建在 `~/.kube` 中或者新集群的配置已添加到 `~/.kube` 的现有 `config` 文件中。

6. 创建节点 IAM 角色。

a. 运行以下命令以创建 `vpc-cni-ipv6-policy.json` 文件。

```
cat >vpc-cni-ipv6-policy <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
EOF
```

b. 创建 IAM policy。

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

c. 运行以下命令以创建 node-role-trust-relationship.json 文件。

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

d. 运行以下命令设置角色名称的变量。您可以将 *AmazonEKSNodeRole* 替换为您选择的任何名称。

```
export node_role_name=AmazonEKSNodeRole
```

e. 创建 IAM 角色。

```
aws iam create-role --role-name $node_role_name --assume-role-policy-document
 file://"node-role-trust-relationship.json"
```

f. 将 IAM policy 附加到 IAM 角色。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy \
    --role-name $node_role_name
```

> **Important**
>
> 为简单起见，在本教程中，该策略附加到此 IAM 角色。但是，在生产集群中，我们建议将该策略附加到单独的 IAM 角色。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。

g. 将两个所需的 IAM 托管策略附加到 IAM 角色。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
```

h. 检索 IAM 角色的 ARN 并将其存储在变量中以供后续步骤使用。

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --query="Role.Arn"
 --output text)
```

7. 创建托管节点组。

a. 查看在上一步中创建的子网的 ID。

```
echo $subnets
```

输出示例如下。

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE,subnet-
0377963d69EXAMPLE,subnet-0c05f819d5EXAMPLE
```

b. 创建节点组。将 *0a1a56c486EXAMPLE*、*099e6ca77aEXAMPLE*、*0377963d69EXAMPLE* 和 *0c05f819d5EXAMPLE* 替换为在上一步输出中返回值。请务必在以下命令之前的输出中删除子网 ID 之间的逗号。您可以将 *t3.medium* 替换为任何 Amazon Nitro 系统实例类型。

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --subnets subnet-0a1a56c486EXAMPLE subnet-099e6ca77aEXAMPLE
 subnet-0377963d69EXAMPLE subnet-0c05f819d5EXAMPLE \
    --instance-types t3.medium --node-role $node_iam_role
```

创建节点组需要几分钟的时间。运行以下命令。在返回的输出为 ACTIVE 之前，请勿继续执行下一步。

```
aws eks describe-nodegroup --region $region_code --cluster-name $cluster_name
 --nodegroup-name $nodegroup_name \
    --query nodegroup.status --output text
```

8. 确认默认 pods 已分配 IP 列中的 IPv6 地址。

```
kubectl get pods -n kube-system -o wide
```

输出示例如下。

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                     NODE                                              NOMINATED
 NODE   READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

9. 确认默认服务已分配 IP 列中的 IPv6 地址。

```
kubectl get services -n kube-system -o wide
```

输出示例如下。

```
NAME         TYPE        CLUSTER-IP           EXTERNAL-IP   PORT(S)          AGE
 SELECTOR
kube-dns     ClusterIP   fd30:3087:b6c2::a    <none>        53/UDP,53/TCP    57m   k8s-
app=kube-dns
```

10. （可选）部署示例应用程序 (p. 315)或部署 Amazon Load Balancer Controller (p. 287) 和示例应用程序以对应用程序 (p. 332)或至 IPv6 pods 的网络 (p. 327)流量进行负载均衡。
11. 在使用完成针对本教程而创建的集群和节点后，应使用以下命令清理创建的资源。删除之前，确保您没有使用本教程之外的任何资源。

    a. 如果您在与之前步骤不同的 shell 中完成此步骤，请设置之前步骤中使用的所有变量的值，然后将 *example values* 替换为完成之前步骤时指定的值。如果在完成之前步骤的同一 shell 中完成此步骤，请跳至下一步。

    ```
    export region_code=region-code
    export vpc_stack_name=my-eks-ipv6-vpc
    export cluster_name=my-cluster
    export nodegroup_name=my-nodegroup
    export account_id=111122223333
    export node_role_name=AmazonEKSNodeRole
    export cluster_role_name=myAmazonEKSClusterRole
    ```

    b. 删除您的节点组。

    ```
    aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name --
    nodegroup-name $nodegroup_name
    ```

    删除需要花费几分钟的时间。运行以下命令。如果返回任何输出，请勿继续执行下一步。

    ```
    aws eks list-nodegroups --region $region_code --cluster-name $cluster_name --
    query nodegroups --output text
    ```

    c. 请删除集群。

    ```
    aws eks delete-cluster --region $region_code --name $cluster_name
    ```

    删除集群需要几分钟时间。在继续之前，请确保使用以下命令删除集群。

    ```
    aws eks describe-cluster --region $region_code --name $cluster_name
    ```

    在输出与以下输出类似之前，请勿继续执行下一步。

    ```
    An error occurred (ResourceNotFoundException) when calling the DescribeCluster
     operation: No cluster found for name: my-cluster.
    ```

    d. 删除您创建的 IAM 资源。如果您选择的名称与之前步骤中使用的名称不同，则将 *AmazonEKS_CNI_IPv6_Policy* 替换为您选择的名称。

    ```
    aws iam detach-role-policy --role-name $cluster_role_name --policy-arn
     arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
    aws iam detach-role-policy --role-name $node_role_name --policy-arn
     arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
    aws iam detach-role-policy --role-name $node_role_name --policy-arn
     arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
    aws iam detach-role-policy --role-name $node_role_name --policy-arn
     arn:aws:iam::$account_id:policy/AmazonEKS_CNI_IPv6_Policy
    ```

```
aws iam delete-policy --policy-arn arn:aws:iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-role --role-name $cluster_role_name
aws iam delete-role --role-name $node_role_name
```

e. 删除创建 VPC 的 Amazon CloudFormation 堆栈。

```
aws cloudformation delete-stack --region $region_code --stack-name
 $vpc_stack_name
```

适用于 pods 的 SNAT

如果您使用 IPv6 系列部署集群，那么本主题中的信息不适用于您的集群，因为 IPv6 地址未进行网络转换。有关集群使用 IPv6 的详细信息，请参阅 the section called “IPv6” (p. 248)。

默认情况下，您的集群中的每个 pod 都从无类别域间路由（CIDR）块中分配到一个私有 IPv4 地址，该块与 pod 部署在其中的 VPC 关联。相同 VPC 中的 Pods 使用这些私有 IP 地址作为端点来相互通信。当 pod 与不在与您的 VPC 关联的 CIDR 块内的任何 IPv4 地址通信时，适用于 Kubernetes 的 Amazon VPC CNI 插件会将 pod's IPv4 地址转换为 pod 运行所在的节点的主弹性网络接口的主私有 IPv4 地址，默认为 * (p. 258)。

由于此行为：

- 使用 VPC 对等、中转 VPC 或 Direct Connect 连接到您的集群 VPC 的网络或 VPC 中的资源无法启动与您的 pods 的通信。但是您的 pods 可以启动与这些资源的通信并接收他们的响应。
- 您的 pods 只有在运行它们的节点具有一个分配到它的公有或弹性 IP 地址且处于公有子网时，才会与您的互联网资源双向通信。公有子网的关联路由表具有指向互联网网关的路由。私有子网的关联路由表不具有指向互联网网关的路由。我们建议尽可能将节点部署到私有子网。

如果您有以下任一要求，则需要更改适用于 Kubernetes 的 Amazon VPC CNI 插件的默认配置：

- 您具有使用 VPC 对等、中转 VPC 或 Direct Connect 连接到您的集群 VPC 的网络或 VPC 中的资源，它们需要使用 IPv4 地址启动与您的 pods 的通信。
- 您的节点位于私有子网中。

使用以下命令更改默认配置。

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

如果您希望 pods 在更改设置后与 Internet 通信，则与私有子网关联的路由表必须包含指向公有 NAT 网关的路由。互联网上的资源无法启动与私有子网中的 pods 的通信，除非您已将负载均衡器或入口部署到 VPC 中的公有子网。负载均衡器或入口可以将流量路由到 pods 的私有 IP 地址。有关创建网络负载均衡器和入口（应用程序负载均衡器）的详细信息，请参阅 the section called “网络负载均衡” (p. 327) 和 the section called “应用程序负载均衡” (p. 332)。

*如果 pod's 规范包含 hostNetwork=true（默认值为 false），那么它的 IP 地址不会转换为其他地址。对于在您的集群上运行的 kube-proxy 和 Amazon VPC CNI plugin for Kubernetes pods，默认情况便是如此。对于这些 pods，IP 地址与节点的主 IP 地址相同，因此 pod's IP 地址未转换。有关的 pod's hostNetwork 设置的更多信息，请参阅 Kubernetes API 参考中的 PodSpec v1 核心。

教程：自定义联网

默认情况下，当 Amazon VPC CNI plugin for Kubernetes 为您的 Amazon EC2 节点创建辅助弹性网络接口时，它会在与该节点的主网络接口相同的子网中创建它们。它还将相同的安全组关联到与主网络接口关联

的辅助网络接口。出于以下一个或多个原因，您可能希望该插件在不同子网中创建辅助网络接口，或者希望将不同的安全组与辅助网络接口关联，或者希望同时实现此两者：

- 主网络接口所在子网中可用的 IPv4 地址数量有限。这可能会限制您可以在子网中创建的 pods 的数量。通过对辅助网络接口使用不同的子网，您可以增加可用于 pods 的 IPv4 地址的数量。
- 出于安全原因，您的 pods 可能需要使用与节点的主网络接口不同的子网或安全组。
- 节点在公有子网中配置，并且您希望将 pods 放在私有子网中。与公有子网关联的路由表包含指向互联网网关的路由。与私有子网关联的路由表不包含指向互联网网关的路由。

注意事项

- 启用自定义联网后，不会将分配给主网络接口的 IP 地址分配给 pods。仅会将辅助网络接口的 IP 地址分配给 pods。
- 如果您的集群使用 IPv6 系列，您无法使用自定义联网。
- 如果您计划仅为帮助缓解 IPv4 地址耗尽而使用自定义联网，您可以改为使用 IPv6 系列创建集群。有关更多信息，请参阅 教程：将 IPv6 地址分配给 pods 和 services (p. 248)。
- 尽管部署到为辅助网络接口指定的子网的 pods 可以使用与节点的主网络接口不同的子网和安全组，子网和安全组也必须与节点位于同一 VPC 中。

先决条件

- 熟悉 Amazon VPC CNI plugin for Kubernetes 如何创建辅助网络接口并将 IP 地址分配给 pods。有关更多信息，请参阅 GitHub 上的 ENI 分配。
- 您的设备或 Amazon CloudShell 上安装并配置了 2.7.21 版或更高版本，或 1.25.46 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 aws configure 进行快速配置。
- 您的设备或 Amazon CloudShell 上安装了 kubectl 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 1.22，则可以将 kubectl 版本 1.21、1.22 或 1.23 用于它。要安装或升级 kubectl，请参阅 安装或更新 kubectl (p. 4)。
- 我们建议您在 Bash shell 中完成本主题中的步骤。如果您没有使用 Bash shell，则某些脚本命令（例如行延续字符以及变量的设置和使用方式）需要调整 shell。

在本教程中，我们建议使用 *example values*，除非注意到要替换它们。您可以在完成生产集群的步骤时替换任何 *example value*。我们建议在同一终端完成所有步骤。这是因为变量是在整个步骤中设置和使用的，并且不会存在于不同的终端中。

步骤 1：创建测试 VPC 和集群

创建集群

以下步骤帮助您创建测试 VPC 和集群并为该集群配置自定义联网。我们不建议将测试集群用于生产工作负载，因为本主题没有涵盖在生产集群上可能使用的几种不相关的功能。有关更多信息，请参阅the section called “创建集群” (p. 21)。

当您想将自定义联网部署到生产集群时，请跳至 the section called “步骤 2：配置 VPC” (p. 261)。

1. 定义一些要在其余步骤中使用的变量。将 *region-code* 替换为您要在其中创建集群的 Amazon Web Services 区域。

```
export cluster_name=my-custom-networking-cluster
export region_code=region-code
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. 创建 VPC。

1. 使用 Amazon EKS Amazon CloudFormation 模板创建 VPC。

```
aws cloudformation create-stack --region $region_code --stack-name my-eks-custom-
networking-vpc \
  --template-url https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --parameters ParameterKey=VpcBlock,ParameterValue=192.168.0.0/24 \
  ParameterKey=PrivateSubnet01Block,ParameterValue=192.168.0.64/27 \
  ParameterKey=PrivateSubnet02Block,ParameterValue=192.168.0.96/27 \
  ParameterKey=PublicSubnet01Block,ParameterValue=192.168.0.0/27 \
  ParameterKey=PublicSubnet02Block,ParameterValue=192.168.0.32/27
```

   Amazon CloudFormation 堆栈需要几分钟的时间来创建。要检查堆栈的部署状态，请运行以下命令。

```
aws cloudformation describe-stacks --region $region_code --stack-name my-eks-custom-
networking-vpc \
    --query Stacks[].StackStatus --output text
```

   在命令的输出变成 `CREATE_COMPLETE` 之前，请勿继续执行下一步。

2. 使用模板创建的私有子网 ID 的值定义变量。

```
subnet_id_1=$(aws cloudformation describe-stack-resources --stack-name my-eks-custom-
networking-vpc --region $region_code \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-custom-
networking-vpc --region $region_code \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet02'].PhysicalResourceId" --output text)
```

3. 使用上一步中检索到的子网的可用区定义变量。

```
az_1=$(aws ec2 describe-subnets --subnet-ids $subnet_id_1 --region $region_code --
query 'Subnets[*].AvailabilityZone' --output text)
az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --region $region_code --
query 'Subnets[*].AvailabilityZone' --output text)
```

3. 创建集群 IAM 角色。

   a. 运行以下命令以创建 IAM 信任策略 JSON 文件。

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

   b. 创建 Amazon EKS 集群 IAM 角色。如有必要，使用您在上一步中将文件写入到的计算机上的路径为 `eks-cluster-role-trust-policy.json` 添加前缀。该命令将您在上一步中创建的信任策

略与角色关联。要创建 IAM 角色，必须为正在创建角色的 IAM 实体（用户或角色）分配以下 IAM 操作（权限）：iam:CreateRole。

```
aws iam create-role --role-name myCustomNetworkingAmazonEKSClusterRole --assume-
role-policy-document file://"eks-cluster-role-trust-policy.json"
```

c. 将名为 AmazonEKSClusterPolicy 的 Amazon EKS 托管 IAM policy 附加到角色。要将 IAM policy 附加到 IAM 实体（用户或角色），必须为附加该策略的 IAM 实体分配以下 IAM 操作（权限）之一：iam:AttachUserPolicy 或 iam:AttachRolePolicy。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name myCustomNetworkingAmazonEKSClusterRole
```

4. 创建 Amazon EKS 集群并配置您的设备以与其进行通信。

a. 创建集群。

```
aws eks create-cluster --region $region_code --name my-custom-networking-cluster \
   --role-arn arn:aws:iam::$account_id:role/myCustomNetworkingAmazonEKSClusterRole
 \
   --resources-vpc-config subnetIds=$subnet_id_1","$subnet_id_2
```

Note

您可能会收到一个错误，指示请求中的可用区之一没有足够容量来创建 Amazon EKS 集群。如果发生这种情况，错误输出将包含可支持新集群的可用区。再次尝试使用至少两个位于您账户中支持的可用区的子网创建集群。有关更多信息，请参阅容量不足 (p. 478)。

b. 创建集群需要几分钟时间。要检查集群的部署状态，请运行以下命令。

```
aws eks describe-cluster --region $region_code --name my-custom-networking-cluster
 --query cluster.status
```

在命令的输出变成 ACTIVE 之前，请勿继续执行下一步。

c. 配置 kubectl 以与集群通信。

```
aws eks update-kubeconfig --region $region_code --name my-custom-networking-cluster
```

步骤 2：配置 VPC

本教程需要在 the section called “步骤 1：创建测试 VPC 和集群” (p. 259) 中创建的 VPC。对于生产集群，通过将所有 *example values* 替换为您自己的值来相应调整 VPC 的步骤。

1. 通过运行以下命令确认您当前安装的 Amazon VPC CNI 插件版本为 1.6.3-eksbuild.2 或更高版本。

```
kubectl describe daemonset aws-node -n kube-system | grep amazon-k8s-cni: | cut -d "/"
 -f 2
```

输出示例如下。

```
amazon-k8s-cni:v1.11.3-eksbuild.1
```

如果您的版本早于 `1.6.3-eksbuild.2`，则必须予以更新。想要了解更多信息，请参阅 管理 Amazon VPC CNI plugin for Kubernetes (p. 233) 的更新部分。

2. 检索您的集群 VPC 的 ID，并将其存储在变量中，以便在后续步骤中使用。对于生产集群，请将 *my-custom-networking-cluster* 替换为集群的名称。

```
vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --region
 $region_code --query "cluster.resourcesVpcConfig.vpcId" --output text)
```

3. 将另一个无类别域间路由（CIDR）块与集群的 VPC 关联。CIDR 块不能与任何现有关联的 CIDR 块重叠。

   1. 查看与您的 VPC 关联的当前 CIDR 块。

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --region $region_code \
    --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

   输出示例如下。

```
---------------------------------
|          DescribeVpcs          |
+-----------------+--------------+
|    CIDRBlock    |    State     |
+-----------------+--------------+
|  192.168.0.0/24 |  associated  |
+-----------------+--------------+
```

   2. 将额外 CIDR 块与 VPC 关联。有关更多信息，请参阅《Amazon VPC 用户指南》中的将额外的 IPv4 CIDR 块与 VPC 关联。

```
aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --region $region_code --cidr-
block 192.168.1.0/24
```

   3. 确认新区块已关联。

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --region $region_code \
    --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

   输出示例如下。

```
---------------------------------
|          DescribeVpcs          |
+-----------------+--------------+
|    CIDRBlock    |    State     |
+-----------------+--------------+
|  192.168.0.0/24 |  associated  |
|  192.168.1.0/24 |  associated  |
+-----------------+--------------+
```

   在您的新 CIDR 区块的 `State` 为 `associated` 之前，请勿继续执行下一步。

4. 在现有子网所在的每个可用区中创建要使用的任意数量的子网。指定一个在之前的步骤中与 VPC 关联的 CIDR 块内的 CIDR 块。

   1. 创建新子网。子网必须在不同于现有子网所在的 VPC CIDR 块中创建子网，但与现有子网位于同一可用区中。在此示例中，在当前私有子网所在的每个可用区的新 CIDR 块中创建一个子网。创建的子网

的 ID 存储在变量中以供后续步骤使用。Name 值与分配给上一步中使用 Amazon EKS VPC 模板创建的子网的值相匹配。名称不是必填项。您可以使用不同的名称。

```
new_subnet_id_1=$(aws ec2 create-subnet --vpc-id $vpc_id --region $region_code --
availability-zone $az_1 --cidr-block 192.168.1.0/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet01},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
new_subnet_id_2=$(aws ec2 create-subnet --vpc-id $vpc_id --region $region_code --
availability-zone $az_2 --cidr-block 192.168.1.32/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet02},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
```

Important

默认情况下，您的新子网与您的 VPC 的主路由表隐式关联。此路由表允许在 VPC 中部署的所有资源之间进行通信。但是，它不允许与 IP 地址位于与您的 VPC 关联的 CIDR 块之外的资源进行通信。您可以将自己的路由表关联到子网以改变此行为。有关更多信息，请参阅《Amazon VPC 用户指南》中的子网路由表。

2. 查看 VPC 中的当前子网。

```
aws ec2 describe-subnets --region $region_code --filters "Name=vpc-id,Values=$vpc_id"
 \
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

输出示例如下。

```
-------------------------------------------------------------------
|                          DescribeSubnets                        |
+------------------+--------------------+--------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId         |
+------------------+--------------------+--------------------------+
|  cn-north-1d     |  192.168.0.0/27    |     subnet-example1      |
|  cn-north-1a     |  192.168.0.32/27   |     subnet-example2      |
|  cn-north-1a     |  192.168.0.64/27   |     subnet-example3      |
|  cn-north-1d     |  192.168.0.96/27   |     subnet-example4      |
|  cn-north-1a     |  192.168.1.0/27    |     subnet-example5      |
|  cn-north-1d     |  192.168.1.32/27   |     subnet-example6      |
+------------------+--------------------+--------------------------+
```

您可以看到您创建的 192.168.1.0 CIDR 块中的子网与 192.168.0.0 CIDR 块中的子网在同一个可用区中。

步骤 3：配置 Kubernetes 资源

要配置 Kubernetes 资源

1. 将 AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG 环境变量设置为 aws-node DaemonSet 中的 true。

```
kubectl set env daemonset aws-node -n kube-system
 AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

2. 检索集群安全组 (p. 231)的 ID，并将其存储在变量中，以便在后续步骤中使用。当您创建集群时，Amazon EKS 会自动创建此安全组。

```
cluster_security_group_id=$(aws eks describe-cluster --name $cluster_name --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
```

3. 为您希望在其中部署 pods 的每个子网创建 ENIConfig 自定义资源。

   a. 为每个网络接口配置创建一个唯一的文件。

   以下命令为您在上一步中创建的两个子网创建单独的 ENIConfig 文件。name 的值必须是唯一的。该名称与子网所在的可用区相同。将集群安全组分配给 ENIConfig。

```
cat >$az_1.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
EOF
```

```
cat >$az_2.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_2
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_2
EOF
```

   对于生产集群，您可以对以前的命令进行以下更改：

   - 将 *$cluster_security_group_id* 替换为您想要用于每个 ENIConfig 的现有安全组的 ID。
   - 我们建议将将您的 ENIConfigs 尽可能命名为与您将 ENIConfig 用于的可用区相同。您可能需要出于各种原因为您的 ENIConfigs 使用与可用区名称不同的名称。例如，如果您在同一可用区中有两个以上的子网，并且想将它们同时用于自定义联网，则需要将多个 ENIConfigs 用于同一个可用区。由于每个 ENIConfig 都需要一个唯一名称，您不能使用可用区名称命名多个 ENIConfigs。

     如果您的 ENIConfig 名称与可用区名称不一样，将 *$az_1* 和 *$az_2* 替换为上一个命令中您自己的名称，并且稍后在本教程中使用 ENIConfig 注释您的节点 (p. 267)。

     Note

     如果您未指定与生产集群一起使用的有效安全组并且您正在使用：

     - 版本 1.8.0 或更高版本的 Amazon VPC CNI plugin for Kubernetes，则使用与节点的主弹性网络接口关联的安全组。
     - 1.8.0 之前的 Amazon VPC CNI plugin for Kubernetes 版本，则 VPC 的默认安全组将分配给辅助网络接口。

Important

- `AWS_VPC_K8S_CNI_EXTERNALSNAT=false` 是适用于 Kubernetes 的 Amazon VPC CNI 插件配置中的默认设置。如果您使用的是默认设置，则发往不在与 VPC 关联的 CIDR 块之一内的 IP 地址的流量将使用节点主网络接口的安全组和子网。在用于创建辅助网络接口的 `ENIConfigs` 中定义的子网和安全组不用于此流量。有关该设置的更多信息，请参阅 适用于 pods 的 SNAT (p. 258)。
- 如果您还将安全组用于 pods，则使用 `SecurityGroupPolicy` 中指定的安全组而不是 `ENIConfigs` 中指定的安全组。有关更多信息，请参阅pods 的安全组 (p. 273)。

b. 使用以下命令将您创建的每个自定义资源文件应用于您的集群。

```
kubectl apply -f $az_1.yaml
kubectl apply -f $az_2.yaml
```

4. 确认您的 `ENIConfigs` 已创建。

```
kubectl get ENIConfigs
```

输出示例如下。

```
NAME         AGE
cn-north-1a   117s
cn-north-1d   105s
```

5. 如果您在生产集群上启用自定义联网并将您的 `ENIConfigs` 命名为您将它们用于的可用区之外的其他名称，然后跳到下一步 (p. 265)以部署 Amazon EC2 节点。

   启用 Kubernetes 以将可用区的 `ENIConfig` 自动应用于在您的集群中创建的任何新 Amazon EC2 节点。

   1. 对于本教程中的测试集群，请跳至下一步 (p. 265)。

      对于生产集群，请检查带有 `ENI_CONFIG_ANNOTATION_DEF` 环境变量的键 `k8s.amazonaws.com/eniConfig` 的 annotation 是否存在于 `aws-node DaemonSet` 的容器规范中。

      ```
      kubectl describe daemonset aws-node -n kube-system | grep ENI_CONFIG_ANNOTATION_DEF
      ```

      如果返回输出，则注释存在。如果没有返回输出，则未设置此变量。对于生产集群，您可以使用此设置或以下步骤中的设置。如果使用此设置，它将覆盖以下步骤中的设置。在本教程中，将使用下一步中的设置。

   2. 更新您的 `aws-node DaemonSet` 以将可用区的 `ENIConfig` 自动应用于在您的集群中创建的任何新 Amazon EC2 节点。

      ```
      kubectl set env daemonset aws-node -n kube-system
       ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
      ```

步骤 4：部署 Amazon EC2 节点

要部署 Amazon EC2 节点

1. 创建节点 IAM 角色。

a. 运行以下命令以创建 IAM 信任策略 JSON 文件。

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. 运行以下命令设置角色名称的变量。您可以将 *myCustomNetworkingAmazonEKSNodeRole* 替换为您选择的任何名称。

```
export node_role_name=myCustomNetworkingAmazonEKSNodeRole
```

c. 创建 IAM 角色并将返回的 Amazon 资源名称（ARN）存储在变量中以供后续步骤使用。

```
node_role_arn=$(aws iam create-role --role-name $node_role_name --assume-role-
policy-document file://"node-role-trust-relationship.json" \
    --query Role.Arn --output text)
```

d. 将三个所需的 IAM 托管策略附加到 IAM 角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name $node_role_name
```

> **Important**
>
> 为简单起见，在本教程中，AmazonEKS_CNI_Policy 策略附加到节点 IAM 角色。在生产集群中，我们建议将该策略附加到仅用于 Amazon VPC CNI plugin for Kubernetes 的单独 IAM 角色。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。

2. 创建以下一种类型的节点组。要确定您想要部署的实例类型，请参阅 选择 Amazon EC2 实例类型 (p. 145)。在本教程中，请完成 Managed（托管）、Without a launch template or with a launch template without an AMI ID specified（没有启动模板或带有未指定 AMI ID 的启动模板）选项。如果要将节点组用于生产工作负载，那么我们建议您在部署节点组之前自定熟悉所有的托管 (p. 90)和自行管理 (p. 106)节点组选项。

   - 托管：使用以下选项之一部署您的节点组：
     - 没有启动模板或者带有未指定 AMI ID 的启动模板 – 运行以下命令。在本教程中，请使用 *example values*。对于生产节点组，将所有的 *example values* 替换为您自己的值。

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name my-nodegroup \
```

```
    --subnets $subnet_id_1 $subnet_id_2 --instance-types t3.medium --node-role
 $node_role_arn
```

- 使用具有指定 AMI ID 的启动模板

  1. 确定 Amazon EKS 为您的节点推荐的最大 pods 数量。按照 Amazon EKS 建议每种 Amazon EC2 实例类型的最大 pods 数量 (p. 146) 中的说明进行操作，将 **--cni-custom-networking-enabled** 添加到该主题的步骤 3。请记下输出的内容，以便在下一个步骤中使用。
  2. 在启动模板中，指定 Amazon EKS 优化的 AMI ID，或者指定基于 Amazon EKS 优化 AMI 构建的自定义 AMI，然后使用启动模板部署节点组 (p. 100)并在启动模板中提供以下用户数据。此用户数据会将实际参数传递到 `bootstrap.sh` 文件中。有关引导文件的更多信息，请参阅 GitHub 上的 bootstrap.sh。您可以将 *20* 替换为上一步的值（建议）或您自己的值。

     ```
     /etc/eks/bootstrap.sh my-cluster --use-max-pods false --kubelet-extra-args '--
     max-pods=20'
     ```

     如果您创建的自定义 AMI 不是基于 Amazon EKS 优化 AMI 构建的，则需要自行自定义创建配置。

- 自行管理

  1. 确定 Amazon EKS 为您的节点推荐的最大 pods 数量。按照 Amazon EKS 建议每种 Amazon EC2 实例类型的最大 pods 数量 (p. 146) 中的说明进行操作，将 **--cni-custom-networking-enabled** 添加到该主题的步骤 3。请记下输出的内容，以便在下一个步骤中使用。
  2. 按照 启动自行管理的 Amazon Linux 节点 (p. 107) 中的说明部署节点组。为 BootstrapArguments 参数指定以下文本。您可以将 *20* 替换为上一步的值（建议）或您自己的值。

     ```
     --use-max-pods false --kubelet-extra-args '--max-pods=20'
     ```

节点组创建需要几分钟时间。您可以使用以下命令检查托管节点组创建的状态。

```
aws eks describe-nodegroup --cluster-name $cluster_name --nodegroup-name my-nodegroup
 --query nodegroup.status --output text
```

在返回的输出为 `ACTIVE` 之前，请勿继续执行下一步。

3. 在本教程中，您可以跳过此步骤。

   对于生产集群，如果您没有将您的 `ENIConfigs` 命名为您将其用于的可用区相同，则必须使用应与节点一起使用的 `ENIConfig` 名称对节点进行注释。如果您在每个可用区中只有一个子网并且将 `ENIConfigs` 命名为与可用区相同的名称，则此步骤不需要。这是因为当您在上一步 (p. 265)中启用了它执行此操作时，Amazon VPC CNI plugin for Kubernetes 会自动为您将正确的 `ENIConfig` 与节点关联。

   a. 获取集群中的节点列表。

      ```
      kubectl get nodes
      ```

      输出示例如下。

      ```
      NAME                                             STATUS   ROLES    AGE     VERSION
      ip-192-168-0-126.cn-north-1.compute.internal   Ready    <none>   8m49s   v1.22.9-
      eks-810597c
      ip-192-168-0-92.cn-north-1.compute.internal    Ready    <none>   8m34s   v1.22.9-
      eks-810597c
      ```

   b. 确定每个节点所在的可用区。运行上一步中返回的每个节点的以下命令。

```
aws ec2 describe-instances --filters Name=network-interface.private-dns-
name,Values=ip-192-168-0-126.cn-north-1.compute.internal \
--query 'Reservations[].Instances[].{AvailabilityZone: Placement.AvailabilityZone,
 SubnetId: SubnetId}'
```

输出示例如下。

```
[
    {
        "AvailabilityZone": "cn-north-1d",
        "SubnetId": "subnet-Example5"
    }
]
```

c. 使用您为子网 ID 和可用区创建的 ENIConfig 注释每一个节点。您只能使用一个 ENIConfig 注释节点，尽管使用同一个 ENIConfig 可以注释多个节点。将 *example values* 替换为您自己的值。

```
kubectl annotate node ip-192-168-0-126.cn-north-1.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.cn-north-1.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName2
```

4. 如果您在切换到使用自定义联网功能之前在具有运行的 pods 的生产集群中有节点，请完成以下任务：

a. 确保您有可用的节点正在使用自定义联网功能。

b. 封锁并耗尽节点以正常关闭 pods。有关更多信息，请参阅 Kubernetes 文档中的安全耗尽节点。

c. 终止节点。如果节点位于现有的托管节点组中，则可以删除该节点组。将 *example values* 替换为您自己的值。

```
aws eks delete-nodegroup --cluster-name my-cluster --nodegroup-name my-nodegroup
```

只有注册有 k8s.amazonaws.com/eniConfig 标注的新节点将使用新的自定义联网功能。

5. 确认从 CIDR 块中为 pods 分配一个 IP 地址，该地址与您在上一步中创建的其中一个子网相关联。

```
kubectl get pods -A -o wide
```

输出示例如下。

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
       NODE                                            NOMINATED NODE   READINESS GATES
kube-system   aws-node-2rkn4                1/1     Running   0          7m19s
 192.168.0.93    ip-192-168-0-92.us-west-2.compute.internal    <none>           <none>
kube-system   aws-node-k96wp                1/1     Running   0          7m15s
 192.168.0.108   ip-192-168-0-126.us-west-2.compute.internal   <none>           <none>
kube-system   coredns-657694c6f4-smcgr      1/1     Running   0          56m
 192.168.1.23    ip-192-168-0-92.us-west-2.compute.internal    <none>           <none>
kube-system   coredns-657694c6f4-stwv9      1/1     Running   0          56m
 192.168.1.28    ip-192-168-0-92.us-west-2.compute.internal    <none>           <none>
kube-system   kube-proxy-jgshq              1/1     Running   0          7m19s
 192.168.0.93    ip-192-168-0-92.us-west-2.compute.internal    <none>           <none>
kube-system   kube-proxy-wx9vk              1/1     Running   0          7m15s
 192.168.0.108   ip-192-168-0-126.us-west-2.compute.internal   <none>           <none>
```

您可以看到 coredns pods 从您添加到 VPC 中的 192.168.1.0 CIDR 块中分配到了 IP 地址。如果没有自定义联网，他们就会从 192.168.0.0 CIDR 块分配地址，因为它是最初与 VPC 关联的唯一 CIDR 块。

如果 pod's spec 包含 hostNetwork=true，则会被分配节点的主 IP 地址。不会从您添加的子网中向其分配地址。默认情况下，该值设置为 false。对于在集群上运行的 kube-proxy 和 Amazon VPC CNI plugin for Kubernetes (aws-node) pods，此值设置为 true。这就是 kube-proxy 和插件的 aws-node pods 在上一个输出中没有被分配 192.168.1.*x* 地址的原因。有关的 pod's hostNetwork 设置的更多信息，请参阅 Kubernetes API 参考中的 PodSpec v1 核心。

步骤 5：删除教程资源

完成本教程后，我们建议您删除创建的资源。然后，您可以调整步骤以为生产集群启用自定义联网。

要删除教程资源

1. 如果您创建的节点组只是为了测试，那么请将其删除。

```
aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name my-nodegroup
```

即使在 Amazon CLI 输出显示集群已删除，删除过程实际上可能尚未完成。删除过程需要几分钟时间。通过运行以下命令确认已完成。

```
aws eks describe-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name my-nodegroup --query nodegroup.status --output text
```

在返回的输出类似于以下输出之前，请勿继续执行。

```
An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup
 operation: No node group found for name: my-nodegroup.
```

2. 如果您创建的节点组只是为了测试，则请删除节点 IAM 角色。

   a. 将策略与角色分离。

```
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

   b. 删除角色。

```
aws iam delete-role --role-name myCustomNetworkingAmazonEKSNodeRole
```

3. 请删除集群。

```
aws eks delete-cluster --region $region_code --name $cluster_name
```

通过以下命令确认集群已删除。

```
aws eks describe-cluster --region $region_code --name $cluster_name --query
 cluster.status --output text
```

返回类似以下内容的输出时，将成功删除集群。

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-cluster.
```

4. 删除集群 IAM 角色。

   a. 将策略与角色分离。

   ```
   aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole --
   policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
   ```

   b. 删除角色。

   ```
   aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
   ```

5. 删除您在上一步中创建的子网。

```
aws ec2 delete-subnet  --region $region_code --subnet-id $new_subnet_id_1
aws ec2 delete-subnet --region $region_code --subnet-id $new_subnet_id_2
```

6. 删除您创建的 VPC。

```
aws cloudformation delete-stack --region $region_code --stack-name my-eks-custom-
networking-vpc
```

提高 Amazon EC2 节点的可用 IP 地址数量

Important

此功能在 Amazon 中国区域不可用。

预设情况下，可分配给 pods 的 IP 地址数取决于分配给弹性网络接口的 IP 地址数量以及附加到 Amazon EC2 节点的网络接口数。您可以配置 `1.9.0` 版或更高版本的 Amazon VPC CNI 附加组件以分配 /28 `IPv4` 地址前缀。使用 `1.10.1` 或更高版本的附加组件时，您仍然可以分配 `/28` `IPv4` 地址前缀，但如果您的集群是 `1.21` 版或更高版本，并且您已将其配置为 `IPv6` (p. 248)，则可以改为分配 `/80` `IPv6` 地址前缀。

配置为前缀分配时，CNI 附加组件可以为网络接口分配的 IP 地址远远超过分配单个 IP 地址时的 IP 地址。然后，节点可以向 pods 分配的可用 IP 地址数量将大幅提升。有关使附加组件能够执行此操作的 Amazon EC2 功能的更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的为 Amazon EC2 网络接口分配前缀。

如果不启用此功能，附加组件必须进行更多 Amazon EC2 应用程序编程接口（API）调用，以配置 pod 连接所需的网络接口和 IP 地址。这些 API 调用的频率与每个 VPC 中的大量网络接口可能会共同导致 pod 和实例启动时间延长，因为集群要扩展到更大的规模。这会导致扩展延迟以满足大型和尖峰工作负载的需求，并增加成本和管理开销，因为您需要配置额外的集群和 VPC 来满足扩展需求。请参阅 Kubernetes 可扩展性阈值了解更多信息。

注意事项

- Amazon基于 Nitro 的节点使用此功能。不基于 Nitro 的实例会继续分配单个辅助 IP 地址，但分配给 pods 的 IP 地址数量比 Nitro-based 实例少得多。
- 将附加组件配置为向网络接口分配前缀后，您将无法在不删除集群所有节点组中的所有节点的情况下将 Amazon VPC CNI 附加组件版本降级到 `1.9.0` 版（或 `1.10.1` 版）以下。
- 您的 VPC 必须具有足够的可用连续 `/28` `IPv4` 地址数据块来支持此功能。

- 每种实例类型支持最大数量的 pods。如果您的托管节点组由多种实例类型组成，则集群中某个实例的最大 pods 数量的最小值将应用于集群中的所有节点。
- 如果您有现有的托管节点组，则节点组的下一次 AMI 或启动模板更新会导致新的 Worker 节点出现时带有启用了新的 IP 地址前缀分配 `max-pod` 值。

先决条件

- 现有集群。要部署一个角色，请参阅 创建 Amazon EKS 集群 (p. 21)。
- `1.9.0` 版或更高版本（适用于 `1.20` 版或更早版本的集群，或 `1.21` 版或更高版本的集群，配置为使用 `IPv4`）或者 `1.10.1` 版或更高版本（适用于 `1.21` 版或更高版本的集群，配置为使用 `IPv6`）的 Amazon VPC CNI plugin for Kubernetes 附加组件已部署到您的集群。

提高 Amazon EC2 节点的可用 IP 地址数量的步骤

1. 确认您当前安装的 Amazon VPC CNI plugin for Kubernetes 版本为 `1.9.0` 或 `1.10.1` 或更高版本。

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/" -f 2
   ```

   输出示例如下。

   ```
   amazon-k8s-cni:v1.10.1-eksbuild.1
   ```

   如果您的版本早于 `1.9.0`，则必须予以更新。想要了解更多信息，请参阅 管理 Amazon VPC CNI plugin for Kubernetes (p. 233) 的更新部分。

2. 启用参数，以便为 Amazon VPC CNI `DaemonSet` 的网络接口分配前缀。当您部署 `1.21` 或更高版本的集群时，`1.10.1` 或更高版本的 Amazon VPC CNI plugin for Kubernetes 附加组件随它一起部署。如果您使用 `IPv6` 系列创建集群，这个设置将被默认设置为 `true`。如果您使用 `IPv4` 系列创建集群，这个设置将被默认设置为 `false`。

   ```
   kubectl set env daemonset aws-node -n kube-system ENABLE_PREFIX_DELEGATION=true
   ```

   Important

   即使您的子网有免费的 IP 地址，如果子网没有任何连续 `/28` 数据块可用，您也将在 Amazon VPC CNI plugin for Kubernetes 日志中看到以下错误：

   ```
   "failed to allocate a private IP/Prefix address: InsufficientCidrBlocks: The
    specified subnet does not have enough free cidr blocks to satisfy the request"
   ```

   发生这种情况的原因可能是分散在子网中的现有辅助 IP 地址的碎片。要解决此错误，请创建一个新子网并在那里启动 pods，或者使用 Amazon EC2 子网 CIDR 预留在子网中预留空间以便与前缀分配一起使用。有关更多信息，请参阅《Amazon VPC 用户指南》中的子网 CIDR 预留。

3. 如果您计划在没有启动模板的情况下部署托管节点组，或者采用尚未在其中指定 AMI ID 的启动模板，并且您使用的是在先决条件中列出的 Amazon VPC CNI plugin for Kubernetes 版本或更高版本，则跳至下一步。托管节点组会自动为您计算最大 pods 数量。

   如果您正在部署自我管理的节点组或带有启动模板的托管节点组，且其启动模板已指定了 AMI ID，则必须确定 Amazon EKS 为节点推荐的最大 pods 数量。按照 Amazon EKS 建议每种 Amazon EC2 实例类型的最大 pods 数量 (p. 146) 中的说明进行操作，将 **`--cni-prefix-delegation-enabled`** 添加到步骤 3。请记下输出的内容，以便在下一个步骤中使用。

Important

托管节点组强制执行 maxPods 的值的最大数量。对于 vCPUs 少于 30 个的实例，最大数量为 110，对于所有其他实例，最大数量为 250。无论是否启用前缀委派，均应用此最大数量。

4. 如果您使用配置为使用 IPv6 的 1.21 或更高版本的集群，请跳至下一步。

   在以下选项中指定参数。要确定哪个选项适合您以及为其提供哪些值，请参阅 GitHub 上的 WARM_PREFIX_TARGET、WARM_IP_TARGET 和 MINIMUM_IP_TARGET。

   您可以将 *example values* 替换为大于零的值。

   - WARM_PREFIX_TARGET

     ```
     kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
     ```

   - WARM_IP_TARGET 或 MINIMUM_IP_TARGET：如果设置了两种值之一，则其会覆盖所设置的 WARM_PREFIX_TARGET 的值。

     ```
     kubectl set env ds aws-node -n kube-system WARM_IP_TARGET=5
     ```

     ```
     kubectl set env ds aws-node -n kube-system MINIMUM_IP_TARGET=2
     ```

5. 使用至少一种 Amazon EC2 Nitro Amazon Linux 2 实例类型创建以下类型之一的节点组。有关 Nitro 实例类型的列表，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的基于 Nitro 系统构建的实例。Windows 中不支持此功能。对于包含 *110* 的选项，将其替换为第 3 步中的值（建议）或您自己的值。

   - 自行管理的节点组 – 按照 启动自行管理的 Amazon Linux 节点 (p. 107) 中的说明部署节点组。为 BootstrapArguments 参数指定以下文本。

     ```
     --use-max-pods false --kubelet-extra-args '--max-pods=110'
     ```

     如果使用 eksctl 创建节点组，可以使用下面的命令。

     ```
     eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-node 110
     ```

   - 托管：使用以下选项之一部署您的节点组：
     - 没有启动模板或者没有指定 AMI ID 的启动模板：完成 创建托管节点组 (p. 90) 中的过程。托管节点组会自动为您计算 Amazon EKS 建议的 max-pods 值。
     - 使用具有指定 AMI ID 的启动模板：在启动模板中，指定 Amazon EKS 优化的 AMI ID，或者指定基于 Amazon EKS 优化 AMI 构建的自定义 AMI，然后使用启动模板部署节点组 (p. 100)并在启动模板中提供以下用户数据。此用户数据会将实际参数传递到 bootstrap.sh 文件中。有关引导文件的更多信息，请参阅 GitHub 上的 bootstrap.sh。

       ```
       /etc/eks/bootstrap.sh my-cluster \
         --use-max-pods false \
         --kubelet-extra-args '--max-pods=110'
       ```

       如果使用 eksctl 创建节点组，可以使用下面的命令。

       ```
       eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
       ```

       如果您创建的自定义 AMI 不是基于 Amazon EKS 优化 AMI 构建的，则需要自行自定义创建配置。

Note

如果您还想将 IP 地址分配给不同于实例子网的 pods，则需要在此步骤中启用该功能。有关更多信息，请参阅教程：自定义联网 (p. 258)。

6. 节点部署完成后，请查看集群中的节点。

```
kubectl get nodes
```

输出示例如下。

```
NAME                                             STATUS     ROLES    AGE    VERSION
ip-192-168-22-103.region-code.compute.internal   Ready      <none>   19m    v1.20.4-
eks-6b7464
ip-192-168-97-94.region-code.compute.internal    Ready      <none>   19m    v1.20.4-
eks-6b7464
```

7. 描述其中一个节点以确定节点的 max-pods。

```
kubectl describe node node-name ip-192-168-22-103.region-code.compute.internal
```

输出示例如下。

```
...
Allocatable:
  attachable-volumes-aws-ebs:  25
  cpu:                         1930m
  ephemeral-storage:           76224326324
  hugepages-1Gi:               0
  hugepages-2Mi:               0
  memory:                      7244720Ki
  pods:                        110
...
```

在先前的输出中，110 是 Kubernetes 将部署到节点的最大 pods 数量。

pods 的安全组

适用于 pods 的安全组将 Amazon EC2 安全组与 Kubernetes pods 集成在一起。您可以使用 Amazon EC2 安全组定义允许流向和来自于您部署到运行在许多 Amazon EC2 实例类型和 Fargate 上节点的 pods 的入站和出站网络流量的规则。有关此功能的详细说明，请参阅 pods 的安全组介绍博客文章。

注意事项

在部署适用于 pods 的安全组之前，请考虑以下限制和条件：

- 在 Amazon EC2 节点上运行的 Pods 可以是 Amazon EKS 支持的任何 Kubernetes 版本 (p. 59)，但在 Fargate (p. 127) 上运行的 pods 必须在 1.18 或更高版本的集群中。
- 适用于 pods 的安全组不能与 Windows 节点一起使用。
- pods 的安全组不能与为包含 Amazon EC2 节点的 IPv6 系列配置的集群一起使用。但是，您可以将 pods 的安全组与为仅包含 Fargate 节点的 IPv6 系列配置的集群一起使用。有关更多信息，请参阅教程：将 IPv6 地址分配给 pods 和 services (p. 248)。
- 大多数基于 Nitro 的 Amazon EC2 实例系列都支持 pods 的安全组，包括 m5、c5、r5、p3、m6g、c6g 和 r6g 实例系列。t3 实例系列不受支持。有关受支持的实例的完整集，请参阅 GitHub 上的 limits.go 文件。您的节点必须是在该文件中拥有 IsTrunkingCompatible: true 的所列出的实例类型之一。

- 如果您还使用 pod 安全策略来限制对 pod 变异的访问，则 `eks-vpc-resource-controller` 和 `vpc-resource-controller` Kubernetes 服务账户必须在 Kubernetes `ClusterRoleBinding` 中指定用于 `psp` 所分配到的 `role`。如果您使用的是默认 Amazon EKS `psp`、`role` 和 `ClusterRoleBinding` (p. 453)，则此为 `eks:podsecuritypolicy:authenticated` `ClusterRoleBinding`。例如，您将服务账户添加到 `subjects:` 部分，如以下示例所示：

```
...
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: system:authenticated
  - kind: ServiceAccount
    name: vpc-resource-controller
  - kind: ServiceAccount
    name: eks-vpc-resource-controller
```

- 如果您将自定义联网 (p. 258)和适用于 pods 的安全组一起使用，则请使用适用于 pods 的安全组所指定的安全组，不要使用 `ENIconfig` 中指定的安全组。
- 如果您使用的是版本 `1.10.2` 或更早版本的 Amazon VPC CNI 插件，并且将 `terminationGracePeriodSeconds` 设置包括在您的 pod 规范中，则该设置的值不能为零。
- 如果您使用的是版本 `1.10` 或更早版本的 Amazon VPC CNI 插件或 `POD_SECURITY_GROUP_ENFORCING_MODE=strict` 的版本 `1.11`（默认设置），则使用 `externalTrafficPolicy` 设置为 `Local` 的实例目标的 `NodePort` 和 `LoadBalancer` 型 Kubernetes 服务在您将安全组分配到的 pods 中不受支持。有关将负载均衡器与实例目标一起使用的更多信息，请参阅 Amazon EKS 上的网络负载均衡 (p. 327)。如果您使用的是 `POD_SECURITY_GROUP_ENFORCING_MODE=standard` 的版本 `1.11` 或者更高版本的插件，则支持 `externalTrafficPolicy` 被设置为 `Local` 的实例类型。
- 如果您使用的是版本 `1.10` 或更早版本的 Amazon VPC CNI 插件或 `POD_SECURITY_GROUP_ENFORCING_MODE=strict` 的版本 `1.11`（默认设置），对于来自分配有安全组的 pods 的出站流量，禁用源 NAT，以便应用出站安全组规则。要访问 Internet，必须在配置了 NAT 网关或实例的私有子网中部署的节点上启动具有已分配安全组的 pods。将分配的安全组部署到公有子网的 Pods 无法访问 Internet。

  如果您使用的是 `POD_SECURITY_GROUP_ENFORCING_MODE=standard` 的版本 `1.11` 或者更高版本的插件，则发往 VPC 之外的 pod 流量将转换为实例主网络接口的 IP 地址。对于此流量，将使用主网络接口的安全组中的规则，而不是 pod's 安全组中的规则。
- 要将 Calico 网络策略用于具有关联安全组的 pods，您必须使用版本 `1.11.0` 或更高版本的 Amazon VPC CNI 插件并设置 `POD_SECURITY_GROUP_ENFORCING_MODE=standard`。否则，流向和来自具有关联安全组的 pods 的流量不受 Calico 网络策略 (p. 306)执行限制，并且仅受限于 Amazon EC2 安全组执行。要更新 Amazon VPC CNI 版本，请参阅 管理 Amazon VPC CNI plugin for Kubernetes (p. 233)。
- 在使用集群中的安全组的 Amazon EC2 节点上运行且使用 Nodelocal DNSCache 的 Pods 仅支持版本 `1.11.0` 或更高版本 Amazon VPC CNI 插件和 `POD_SECURITY_GROUP_ENFORCING_MODE=standard`。要更新 Amazon VPC CNI 插件版本，请参阅 管理 Amazon VPC CNI plugin for Kubernetes (p. 233)。

将 Amazon VPC CNI 附加组件配置为 pods 的安全组

要部署适用于 pods 的安全组

如果您使用的是仅适用于 Fargate pods 的安全组，而且集群上没有任何 Amazon EC2 节点，请跳至 部署示例应用程序 (p. 278)。

1. 使用以下命令查看您当前的 CNI 插件版本：

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/" -f 2
```

输出示例如下。

```
amazon-k8s-cni-init:1.7.5-eksbuild.1
amazon-k8s-cni:1.7.5-eksbuild.1
```

如果您的 CNI 插件版本低于 1.7.7，请将其更新到 `1.7.7` 或更高版本。有关更多信息，请参阅更新 Amazon VPC CNI plugin for Kubernetes 附加组件 (p. 236)。

2. 将 `AmazonEKSVPCResourceController` 托管策略添加到与您的 Amazon EKS 集群关联的集群角色 (p. 426)。策略允许角色管理网络接口、网络接口的私有 IP 地址以及与网络实例之间的连接和分离。

   a. 确定您的集群 IAM 角色的 ARN。将 *my-cluster* 替换为您的集群名称。

   ```
   aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text
   ```

   输出示例如下。

   ```
   arn:aws:iam::111122223333:role/eksClusterRole
   ```

   b. 以下命令可将策略附加到名为 *eksClusterRole* 的集群角色。将 *eksClusterRole* 替换为您的角色的名称。

   ```
   aws iam attach-role-policy \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController \
       --role-name eksClusterRole
   ```

3. 通过在 `aws-node DaemonSet` 中将 `ENABLE_POD_ENI` 变量设置为 `true`，启用 Amazon VPC CNI 附加组件来管理 pods 的网络接口。此设置一旦设置为 `true` 后，附加组件会为集群中的每个节点添加一个带有值 `vpc.amazonaws.com/has-trunk-attached=true` 的标注。VPC 资源控制器会创建并附加一个带有 `aws-k8s-trunk-eni` 描述且名为中继网络接口的特殊网络接口。

   ```
   kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
   ```

   Note

   中继网络接口包含在实例类型支持的最大网络接口数中。有关每种实例类型支持的最大网络接口数的列表，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的每种实例类型的每个网络接口的 IP 地址数。如果您的节点已经附加了最大数量的标准网络接口，则 VPC 资源控制器将预订一个空间。您将必须缩减您正在运行的 pods，以便控制器足以分离和删除标准网络接口、创建中继网络接口并将其附加到实例。

   您可以使用以下命令查看哪些节点的 `aws-k8s-trunk-eni` 设置为 `true`。如果返回 `No resources found`，则等几秒钟后重试。上一步需要重新启动 CNI pods，这需要几秒钟的时间。

   ```
   kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
   ```

   创建中继网络接口后，可以从中继或标准网络接口为 pods 分配辅助 IP 地址。如果节点被删除，中继接口将自动删除。

   当您在后面的步骤中部署适用于 pod 的安全组时，VPC 资源控制器会创建一个具有 `aws-k8s-branch-eni` 描述的名为分支网络接口的特殊网络接口，并将安全组与其关联。除了附加到节点的标准网络接口和中继网络接口之外，还会创建分支网络接口。如果您使用的是存活探测或就绪探测器，则您还需要禁用 TCP 早期解复用器，以便 `kubelet` 可以使用 TCP 连接到分支网络接口上的 pods。要禁用 TCP 早期解复用器，请运行以下命令：

```
kubectl patch daemonset aws-node \
  -n kube-system \
  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":
[{"name":"DISABLE_TCP_EARLY_DEMUX","value":"true"}],"name":"aws-vpc-cni-init"}]}}}}'
```

Note

如果您使用的是 `1.11.0` 或更高版本的 Amazon VPC CNI plugin for Kubernetes 附加组件并设置 `POD_SECURITY_GROUP_ENFORCING_MODE=standard`，如下一步所述，则您就不需要运行上一个命令。

4. 如果您的集群使用 NodeLocal DNSCache，或者您想将 Calico 网络策略用于拥有自己的安全组的 pods，或者您有 `NodePort` 和 `LoadBalancer` 类型的 Kubernetes 服务，将 `externalTrafficPolicy` 设置为 `Local` 的实例目标用于您要将安全组分配到的 pods，则您必须使用版本 `1.11.0` 或更高版本的 Amazon VPC CNI plugin for Kubernetes 附加组件，并启用以下设置：

```
kubectl set env daemonset aws-node -n kube-system
 POD_SECURITY_GROUP_ENFORCING_MODE=standard
```

Important

- Pod 安全组规则不适用于位于相同节点上的 pods 之间或介于 pods 和services之间之外的流量，例如 `kubelet` 或 `nodeLocalDNS`。
- 从 pods 到 VPC 以外的地址的出站流量是转换为实例的主网络接口的 IP 地址的网络地址（除非您还设置了 `AWS_VPC_K8S_CNI_EXTERNALSNAT=true`）。对于此流量，将使用主网络接口的安全组中的规则，而不是 pod's 安全组中的规则。
- 要将此设置应用于现有 pods，必须重新启动 pods 在其上运行的 pods 或节点。

5. 创建要将资源部署到该处的命名空间。

```
kubectl create namespace my-namespace
```

6. 将 Amazon EKS `SecurityGroupPolicy` 部署到集群。

   a. 将以下示例安全策略保存到名为 *my-security-group-policy.yaml* 的文件中。如果您希望根据服务账户标签选择 pods，您可以将 `podSelector` 替换为 `serviceAccountSelector`。您必须指定一个或其他选择器。空的 `podSelector`（示例：`podSelector: {}`）会选择命名空间中的所有 pods。空的 `serviceAccountSelector` 会选择命名空间中的所有服务账户。您必须为 `groupIds` 指定 1-5 个安全组 ID。如果指定了多个 ID，则所有安全组中的所有规则的组合都会对选定的 pods 生效。

```
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - sg-abc123
```

Important

- 您在策略中指定的安全组必须是存在的。如果它们不存在，那么当您部署与选择器匹配的 pod 时，该 pod 会在创建过程中处于卡住状态。如果您

描述 pod，则会看到类似于以下内容的错误消息：`An error occurred (InvalidSecurityGroupID.NotFound) when calling the CreateNetworkInterface operation: The securityGroup ID 'sg-abc123' does not exist`。

- 安全组必须允许通过您为其配置了探测器的任何端口来自集群安全组的入站通信（对于 `kubelet`）。
- 安全组必须允许与安全组进行出站通信。此安全组应允许来自其的入站 TCP 和 UDP 端口 53（针对 CoreDNS pods）。
- 如果您将安全组策略与 Fargate 一起使用，请确保您的安全组具有允许 pods 与 Kubernetes 控制面板通信的规则。执行此操作的最简单方法是将集群安全组指定为安全组之一。
- 安全组策略仅适用于新调度的 pods。不影响正在运行的 pods。

b. 部署策略。

```
kubectl apply -f my-security-group-policy.yaml
```

7. 部署其标注与上一步中指定的 *podselector* 的 *my-role* 值相匹配的示例应用程序。

a. 将以下内容保存到一个文件中。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
        role: my-role
    spec:
      containers:
      - name: my-container
        image: my-image
        ports:
        - containerPort: 80
```

b. 使用以下命令部署应用程序。当您部署应用程序时，Amazon VPC CNI plugin for Kubernetes 插件将匹配 `role` 标注，并且您在上一步中指定的安全组将应用到 pod。

```
kubectl apply -f my-security-group-policy.yaml
```

Note

- 如果您的 pod 卡在 `Waiting` 状态，而且您在描述 pod 时看到了 `Insufficient permissions: Unable to create Elastic Network Interface.`，请确认您已在上一步中将 IAM policy 添加到 IAM 集群角色。
- 如果您的 pod 卡在 `Pending` 状态，请确认您的节点实例类型在 Github 上的 `limits.go` 中 `IsTrunkingCompatible: true`，并且尚未达到实例类型支持的最大分支网络接口数乘以节点组中节点数得到的数值。例如，`m5.large` 实例支持 9 个分支网络接口。如果节点组有 5 个节点，则最多可以为节点组创建 45 个分支网络接口。在

删除另一个具有关联安全组的 pod 前，您尝试部署的第 46 个 pod 将会处于 `Pending` 状态。

如果您运行 `kubectl describe pod my-deployment-xxxxxxxxxx-xxxxx -n my-namespace` 并看到类似于以下消息内容的消息，则可以安全地忽略该消息。当 CNI 插件尝试在创建网络接口时设置主机联网并失败时，可能会出现此消息。Amazon VPC CNI plugin for Kubernetes 会记录此事件，直到创建了网络接口为止。

```
Failed to create pod sandbox: rpc error: code =
 Unknown desc = failed to set up sandbox container
 "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f" network for pod
 "my-deployment-59f5f68b58-c89wx": networkPlugin
cni failed to set up pod "my-deployment-59f5f68b58-c89wx_my-namespace" network: add
 cmd: failed to assign an IP address to container
```

您不能超过可在实例类型上运行的 pods 的最大数量。有关可在每种实例类型上运行的 pods 的最大数量的列表，请参阅 GitHub 上的 eni-max-pods.txt。当您删除具有关联安全组的 pod 或删除运行该 pod 的节点时，VPC 资源控制器会删除分支网络接口。如果您使用适用于安全组的 pods 删除带有 pods 的集群，则该控制器不会删除分支网络接口，因此您需要自行删除它们。

部署示例应用程序

要将安全组用于 pods，您必须拥有现有安全组并且部署 Amazon EKS `SecurityGroupPolicy` (p. 279) 到集群中，如以下程序所述。以下步骤介绍如何将安全组策略用于 pod。除非另有说明，请从同一个终端完成所有步骤，因为以下步骤中使用的变量不会在终端之间持续存在。

要使用安全组部署示例 pod

1. 创建要与 pod 一起使用的安全组。后续步骤可帮助您创建一个简单的安全组，仅用于说明目的。在生产集群中，您的规则可能会有所不同。

   a. 检索集群的 VPC 和集群安全组的 ID。将 *my-cluster* 替换为您的集群名称。

   ```
   my_cluster_name=my-cluster
   my_cluster_vpc_id=$(aws eks describe-cluster \
       --name $my_cluster_name \
       --query cluster.resourcesVpcConfig.vpcId \
       --output text)
   my_cluster_security_group_id=$(aws eks describe-cluster \
       --name $my_cluster_name \
       --query cluster.resourcesVpcConfig.clusterSecurityGroupId \
       --output text)
   ```

   b. 为 pod 创建安全组。将 *my-pod-security-group* 替换为您自己的值。请记下运行命令后在输出中返回的安全组的 ID。您将在后面的步骤中用到它。

   ```
   my_pod_security_group_name=my-pod-security-group
   aws ec2 create-security-group \
       --vpc-id $my_cluster_vpc_id \
       --group-name $my_pod_security_group_name \
       --description "My pod security group"
   my_pod_security_group_id=$(aws ec2 describe-security-groups \
       --filters Name=group-name,Values=$my_pod_security_group_name \
       --query 'SecurityGroups[].GroupId' \
       --output text)
   echo $my_pod_security_group_id
   ```

c. 允许 TCP 和 UDP 端口 53 流量从您在上一步中创建的 pod 安全组发送到您的集群安全组。如果您希望 DNS 流量从您的 pod 流向与集群安全组不同的安全组，则将 *$my_cluster_security_group_id* 替换为您自己的安全组 ID。记下每个命令返回的输出中的 SecurityGroupRuleId 返回的值。您将在后面的步骤中用到它们。

```
aws ec2 authorize-security-group-ingress \
    --group-id $my_cluster_security_group_id \
    --protocol tcp --port 53 --source-group $my_pod_security_group_id
aws ec2 authorize-security-group-ingress \
    --group-id $my_cluster_security_group_id \
    --protocol udp --port 53 --source-group $my_pod_security_group_id
```

d. 允许通过任何协议和端口从与安全组关联的任何 pod 到您的 pod 安全组的入站流量。安全组具有默认出站规则，允许通过任何协议和端口从与安全组关联的 pods 到您的任何目的地的出站流量。

```
aws ec2 authorize-security-group-ingress \
    --group-id $my_pod_security_group_id \
    --protocol -1 --port -1 --source-group $my_pod_security_group_id
```

2. 创建要将资源部署到该处的 Kubernetes 命名空间。

```
kubectl create namespace my-namespace
```

3. 将 Amazon EKS SecurityGroupPolicy 部署到集群。

a. 将以下示例安全策略保存到名为 *my-security-group-policy.yaml* 的文件中。如果您希望根据服务账户标签选择 pods，您可以将 podSelector 替换为 serviceAccountSelector。您必须指定一个或其他选择器。空的 podSelector（示例：podSelector: {}）会选择命名空间中的所有 pods。空的 serviceAccountSelector 会选择命名空间中的所有服务账户。您必须为 groupIds 指定 1-5 个安全组 ID。如果指定了多个 ID，则所有安全组中的所有规则的组合都会对选定的 pods 生效。将 *sg-05b1d815d1EXAMPLE* 替换为您在为 pod 创建安全组时在上一步中注明的安全组的 ID。

```
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - sg-05b1d815d1EXAMPLE
```

Important

您为 pod 指定的一个或多个安全组必须符合以下标准：

- 它们必须存在。如果它们不存在，那么当您部署与选择器匹配的 pod 时，该 pod 会在创建过程中处于卡住状态。如果您描述 pod，则会看到类似于以下内容的错误消息：An error occurred (InvalidSecurityGroupID.NotFound) when calling the CreateNetworkInterface operation: The securityGroup ID '*sg-05b1d815d1EXAMPLE*' does not exist。
- 它们必须允许通过您为其配置了探测器的任何端口从集群安全组 (p. 231)进行入站通信（对于 kubelet）。

- 它们必须允许通过 TCP 和 UDP 端口 53 对分配到运行 CoreDNS 的 pods 的安全组的出站通信。您的 CoreDNS pods 的安全组必须允许来自您的安全组的入站 TCP 和 UDP 端口 53 流量。
- 它们必须具备必要的入站和出站规则才能与它们需要与其通信的其他 pods 进行通信。
- 如果您将安全组与 Fargate 一起使用，它们必须具有允许 pods 与 Kubernetes 控制面板通信的规则。执行此操作的最简单方法是将集群安全组指定为安全组之一。

安全组策略仅适用于新调度的 pods。不影响正在运行的 pods。

b. 部署策略。

```
kubectl apply -f my-security-group-policy.yaml
```

4. 部署其标注与上一步中指定的 *podSelector* 的 *my-role* 值相匹配的示例应用程序。

a. 将以下内容保存到名为 *sample-application.yaml* 的文件中。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 4
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
        role: my-role
    spec:
      terminationGracePeriodSeconds: 120
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.21
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

b. 使用以下命令部署应用程序。当您部署应用程序时，Amazon VPC CNI plugin for Kubernetes 插件将匹配 `role` 标注，并且您在上一步中指定的安全组将应用到 pod。

```
kubectl apply -f sample-application.yaml
```

Note

- 如果您的 pod 卡在 `Waiting` 状态，而且您在描述 pod 时看到了 `Insufficient permissions: Unable to create Elastic Network Interface.`，请确认您已在上一步中将 IAM policy 添加到 IAM 集群角色。
- 如果您的 pod 卡在 `Pending` 状态，请确认您的节点实例类型已列在 `limits.go` 中，并且尚未达到实例类型支持的最大分支网络接口数乘以节点组中节点数得到的数值。例如，`m5.large` 实例支持 9 个分支网络接口。如果节点组有 5 个节点，则最多可以为节点组创建 45 个分支网络接口。在删除另一个具有关联安全组的 pod 前，您尝试部署的第 46 个 pod 将会处于 `Pending` 状态。

如果您运行 `kubectl describe pod` ***my-deployment-xxxxxxxxxx-xxxxx*** **-n** ***my-namespace*** 并看到类似于以下消息内容的消息，则可以安全地忽略该消息。当 CNI 插件尝试在创建网络接口时设置主机联网并失败时，可能会出现此消息。CNI 插件会记录此事件，直到创建了网络接口为止。

```
Failed to create pod sandbox: rpc error: code =
 Unknown desc = failed to set up sandbox container
 "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f" network for pod
 "my-deployment-5df6f7687b-4fbjm": networkPlugin
cni failed to set up pod "my-deployment-5df6f7687b-4fbjm-c89wx_my-namespace"
 network: add cmd: failed to assign an IP address to container
```

您不能超过可在实例类型上运行的 pods 的最大数量。有关可在每种实例类型上运行的 pods 的最大数量的列表，请参阅 GitHub 上的 eni-max-pods.txt。当您删除具有关联安全组的 pod 或删除运行该 pod 的节点时，VPC 资源控制器会删除分支网络接口。如果您使用适用于安全组的 pods 删除带有 pods 的集群，则该控制器不会删除分支网络接口，因此您需要自行删除它们。

5. 查看使用示例应用程序部署的 pods。对于本主题的其余部分，此终端称为 `TerminalA`。

```
kubectl get pods -n my-namespace -o wide
```

输出示例如下。

```
NAME                             READY   STATUS    RESTARTS   AGE     IP
 NODE                                                NOMINATED NODE   READINESS GATES
my-deployment-5df6f7687b-4fbjm   1/1     Running   0          7m51s   192.168.53.48
 ip-192-168-33-28.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-j9fl4   1/1     Running   0          7m51s   192.168.70.145
 ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-rjxcz   1/1     Running   0          7m51s   192.168.73.207
 ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-zmb42   1/1     Running   0          7m51s   192.168.63.27
 ip-192-168-33-28.region-code.compute.internal   <none>           <none>
```

6. 在单独的终端中，shell 进入其中一个 pods。对于本主题的其余部分，此终端称为 `TerminalB`。将 *5df6f7687b-4fbjm* 替换为您在上一步输出中返回的其中一个 pods 的 ID。

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

7. 从 `TerminalB` 的 shell 中，确认示例应用程序是否有效。

```
curl my-app
```

输出示例如下。

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
```

您收到了输出，因为运行该应用程序的所有 pods 与您创建的安全组关联。该组包含一条规则，允许安全组所关联的所有 pods 之间的所有流量。允许 DNS 流量从该安全组出站到与您的节点关联的集群安全组。节点正在运行 CoreDNS pods，您的 pods 对该 DNS 进行了名称查找。

8. 从 `TerminalA` 中，删除允许从安全组与集群安全组进行 DNS 通信的安全组规则。如果您在上一步中没有将 DNS 规则添加到集群安全组中，请将 *`$my_cluster_security_group_id`* 替换为您在其中创建规则的安全组的 ID。将 *`sgr-0407e21efdEXAMPLE`* 和 *`sgr-04dedf9701EXAMPLE`* 替换为您在上一步中注明的安全组规则的 ID。

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids sgr-0407e21efdEXAMPLE
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids sgr-04dedf9701EXAMPLE
```

9. 从 `TerminalB` 中，尝试再次访问该应用程序。

```
curl my-app
```

由于 pod 不再能够访问具有与其关联的集群安全组的 CoreDNS pods，尝试失败。集群安全组不再具有安全组规则，该规则允许从与 pod 关联的安全组进行 DNS 通信。

如果您尝试使用上一步中的其中一个 pods 返回的 IP 地址访问应用程序，您仍然会收到响应，因为在与安全组相关联的所有 pods 之间允许所有端口，并且不需要名称查找。

10. 完成实验后，您可以删除使用以下命令创建的示例安全组策略、应用程序和安全组。

```
kubectl delete namespace my-namespace
aws ec2 delete-security-group --group-id $my_pod_security_group_id
```

pods 的多个网络接口

Multus CNI 是一个适用于 Amazon EKS 的容器网络接口（CNI）插件，可将多个网络接口附加到 pod。有关更多信息，请参阅 GitHub 上的 Multus-CNI 文档。

在 Amazon EKS 中，每个 pod 都有一个由 Amazon VPC CNI 插件分配的网络接口。使用 Multus，您可以创建一个具有多个接口的多宿主 pod。这是由作为“元插件”的 Multus 完成的；一个 CNI 插件，可以调用多个其他 CNI 插件。Amazon 对 Multus 的支持将 Amazon VPC CNI 插件配置为默认委托插件。

注意事项

- Amazon EKS 将不会构建和发布单个根输入/输出虚拟化 (SR-IOV) 和数据层面开发套件 (DPDK) CNI 插件。但是，您可以通过 Multus 托管主机设备和 `ipvlan` 插件直接连接到 Amazon EC2 弹性网络适配器 (ENA) 来实现数据包加速。
- Amazon EKS 支持 Multus，它提供了一个通用的流程，可以简单地连接其他 CNI 插件。支持 Multus 和连接过程，但 Amazon 不会支持所有可以连接的兼容 CNI 插件，或者那些与连接配置无关的 CNI 插件中可能出现的问题。
- Amazon EKS 正在为 Multus 插件提供支持和生命周期管理，但不对与其他网络接口相关的任何 IP 地址或其他管理负责。使用 Amazon VPC CNI 插件的默认网络接口的 IP 地址和管理保持不变。
- 官方仅支持 Amazon VPC CNI 插件作为默认委派插件。如果您选择不对主联网使用 Amazon VPC CNI 插件，则需要修改已发布的 Multus 安装清单，以将默认委派插件重新配置为备用 CNI。

- 仅当使用 Amazon VPC CNI 作为主 CNI 时，才支持 Multus。当用于更高阶接口时，我们不支持 Amazon VPC CNI，无论是次要接口还是其他。
- 为防止 Amazon VPC CNI 插件尝试管理分配给 pods 的其他网络接口，请将以下标签添加到网络接口：

  键：`node.k8s.amazonaws.com/no_manage`

  值：`true`
- Multus 与网络策略兼容，但必须对策略进行充实，方可包括某些端口和 IP 地址，而其可能是附加到 pods 的其他网络接口的一部分。

有关实施演练，请参阅 GitHub 上的 Multus 设置指南。

指标帮助程序

CNI 指标帮助程序是一种工具，您可以用来抓取网络接口和 IP 地址信息、在集群级别聚合指标以及将指标推送到 Amazon CloudWatch。要详细了解指标帮助程序，请参阅 GitHub 上的 cni-metrics-helper。

在管理 Amazon EKS 集群时，您可能希望了解已经分配了多少个 IP 地址以及还有多少个地址可用。CNI 指标帮助程序可帮助您：

- 跟踪这些指标随时间的变化
- 诊断和排除与 IP 分配和回收相关的问题
- 提供容量规划见解

预置了节点时，CNI 插件会自动从节点的子网向主网络接口 (`eth0`) 分配辅助 IP 地址池。此 IP 地址池称为暖池，其大小由节点的实例类型确定。例如，一个 `c4.large` 实例的每个接口可以支持三个网络接口和九个 IP 地址。可供指定 pod 使用的 IP 地址数量比最大值（十个）少一个，因为其中一个 IP 地址预留给弹性网络接口自身。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的每个实例类型的每个网络接口的 IP 地址。

在 IP 地址池耗尽时，该插件会自动附加另一个弹性网络接口到实例，并将另外一组辅助 IP 地址分配到接口。此过程将继续，直到节点不再支持额外的弹性网络接口。

系统为您的集群收集以下指标并导出到 CloudWatch：

- 集群可以支持的网络接口的最大数量
- 已分配到 pods 的网络接口数
- 当前分配给 pods 的 IP 地址
- IP 地址总数以及最大可用 IP 地址数
- ipamD 错误数

先决条件

- 集群的现有 Amazon Identity and Access Management (IAM) OpenID Connect (OIDC) 提供程序。要确定您是否已经拥有一个或是否要创建一个，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。
- 您的设备或 Amazon CloudShell 上安装并配置了 `2.7.21` 版或更高版本，或 `1.25.46` 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 `aws configure` 进行快速配置。
- 您的设备或 Amazon CloudShell 上安装了 `kubectl` 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 `1.22`，则可以将 `kubectl` 版本 `1.21`、`1.22` 或 `1.23` 用于它。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl` (p. 4)。
- 如果您的集群为 `1.21` 版本或更高版本，确保您的 Amazon VPC CNI plugin for Kubernetes、`kube-proxy` 和 CoreDNS 附加组件为服务账户令牌 (p. 395)中列出的最低版本。

部署 CNI 指标帮助程序

创建 IAM policy 和角色并部署指标帮助程序。

部署 CNI 指标帮助程序

1. 创建一项 IAM policy，该策略需向 CNI 指标帮助程序授予 `cloudwatch:PutMetricData` 权限，以便向 CloudWatch 发送指标数据。

   a. 要创建名为 `cni-metrics-helper-policy.json` 的文件，请运行以下命令。

   ```
   cat >cni-metrics-helper-policy.json <<EOF
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "cloudwatch:PutMetricData"
               ],
               "Resource": "*"
           }
       ]
   }
   EOF
   ```

   b. 创建一项名为 *AmazonEKSVPCCNIMetricsHelperPolicy* 的 IAM policy。

   ```
   aws iam create-policy --policy-name AmazonEKSVPCCNIMetricsHelperPolicy \
       --description "Grants permission to write metrics to CloudWatch" \
       --policy-document file://cni-metrics-helper-policy.json
   ```

2. 创建 IAM 角色并向其附加此 IAM policy。创建 Kubernetes 服务账户。用 IAM 角色 ARN 对 Kubernetes 服务账户进行注释，用 Kubernetes 服务账户名称对 IAM 角色进行注释。您可以使用 `eksctl` 或 Amazon CLI 创建角色。

   eksctl

   运行以下命令以创建 IAM 角色。请将 *my-cluster* 替换为集群名称，将 *111122223333* 替换为账户 ID，并将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

   ```
   eksctl create iamserviceaccount \
       --name cni-metrics-helper \
       --namespace kube-system \
       --cluster my-cluster \
       --role-name "AmazonEKSVPCCNIMetricsHelperRole" \
       --attach-policy-arn
    arn:aws:iam::111122223333:policy/AmazonEKSVPCCNIMetricsHelperPolicy \
       --approve
   ```

   Amazon CLI

   1. 确定集群的 OIDC 提供商 URL。将 *my-cluster* 替换为您的集群名称。如果命令的输出为 `None`，请查看先决条件。

      ```
      aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer"
       --output text
      ```

      输出示例如下。

```
oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. 创建 IAM 角色，从而向 Kubernetes 服务账户授予 AssumeRoleWithWebIdentity 操作权限。

   a. 将以下内容复制到您的设备。请将 *111122223333* 替换为您的账户 ID。将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 和 *region-code* 替换为在上一步中返回的值。替换值后，运行命令以创建一个名为 trust-policy.json 的文件。

```
cat >trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com.cn/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
          "oidc.eks.region-code.amazonaws.com.cn/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:cni-metrics-helper"
        }
      }
    }
  ]
}
EOF
```

   b. 创建角色。

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIMetricsHelperRole \
  --assume-role-policy-document file://"trust-policy.json"
```

3. 向角色附加 IAM policy。请将 *111122223333* 替换为您的账户 ID。

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/AmazonEKSVPCCNIMetricsHelperPolicy \
  --role-name AmazonEKSVPCCNIMetricsHelperRole
```

3. 使用以下命令将 CNI 指标帮助程序的推荐版本添加到您的集群中。

   Important

   您每次只更新一个次要版本。例如，如果您的当前版本为 1.9，并且您想要更新到 1.11，则应首先更新到 1.10，再更新到 1.11，方法是在以下命令更改版本号。
   推荐版本和最新版本适用于 Amazon EKS 支持的所有 Kubernetes 版本。

   中国（北京）(cn-north-1) 或中国（宁夏）(cn-northwest-1)

```
kubectl apply -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.11.3/
config/master/cni-metrics-helper-cn.yaml
```

4. 用之前创建的 IAM 角色的 ARN 注释上一步创建的 `cni-metrics-helper` Kubernetes 服务账户。将 *111122223333* 替换为账户 ID，将 *my-cluster* 替换为集群名称，将 *AmazonEKSVPCCNIMetricsHelperRole* 替换为上一步创建的 IAM 角色名称。

```
kubectl annotate serviceaccount cni-metrics-helper \
    -n kube-system \
    eks.amazonaws.com/role-
arn=arn:aws:iam::111122223333:role/AmazonEKSVPCCNIMetricsHelperRole
```

5. （可选）配置您的 Kubernetes 服务账户使用的 Amazon Security Token Service 端点类型。有关更多信息，请参阅配置服务账户的 Amazon Security Token Service 端点 (p. 405)。
6. 重新启动 `cni-metrics-helper` 部署。

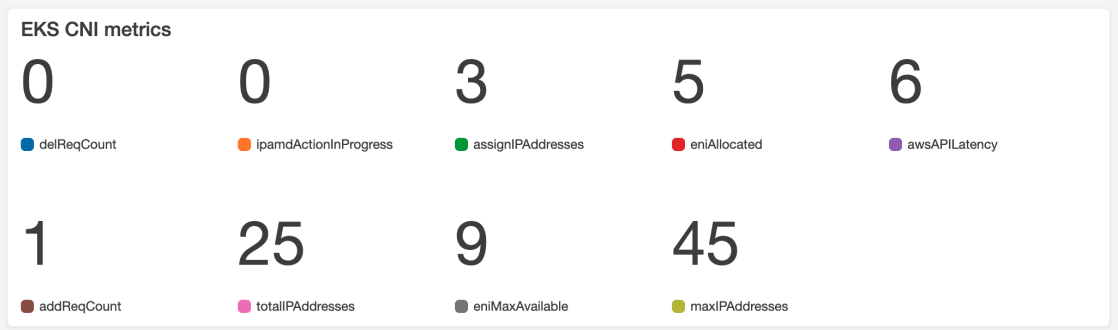
```
kubectl rollout restart \
    deployment cni-metrics-helper \
    -n kube-system
```

创建指标控制面板

部署 CNI 指标帮助程序之后，您可在 CloudWatch 控制台中查看 CNI 指标。本主题可帮助您创建控制面板来查看集群的 CNI 指标。

创建 CNI 指标面板

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 请在左侧导航窗格中，选择 Metrics（指标），然后选择 All metrics（全部指标）。
3. 在 Custom Namespaces（自定义命名空间）中选择 Kubernetes。
4. 选择 CLUSTER_ID。
5. 在 Metrics（指标）选项卡上，选择要添加到控制面板的指标。
6. 在控制台右上方，选择 Actions（操作），然后选择 Add to dashboard（添加到控制面板）。
7. 在 Select a dashboard（选择控制面板）部分选择 Create new（新建），为控制面板输入一个名称（例如 **EKS-CNI-metrics**），然后选择 Create（创建）。
8. 在 Widget type（小组件类型）部分选择 Number（数字）。
9. 在 Customize widget title（自定义小组件标题）部分中，输入控制面板标题的逻辑名称，例如 **EKS CNI metrics**。
10. 选择 Add to dashboard（添加到控制面板）完成操作。现在，您的 CNI 指标已添加到您可以监控的控制面板。



备选的兼容 CNI 插件

Amazon EKS 仅官方支持 Amazon VPC CNI plugin for Kubernetes (p. 233)。Amazon EKS 运行上游 Kubernetes 且经认证符合 Kubernetes 要求，但是，您可以将备用 CNI 插件安装到集群中的 Amazon EC2 节点。如果您的集群中有 Fargate 节点，则您的集群无法将备用 CNI 插件安装到 Fargate 节点。适用于 Amazon EKS 的 Amazon VPC CNI 插件已在您的 Fargate 节点上，但它是唯一可以与 Fargate 节点一起使用的插件。如果您计划在生产中使用备选 CNI 插件，那么我们强烈建议要么您获得商业支持，要么拥有内部专业知识来排查开源 CNI 插件项目的问题并提供修复。

Amazon EKS 与为备选的兼容 CNI 插件提供支持的合作伙伴网络保持着合作关系。请参阅以下合作伙伴的文档，了解受支持的 Kubernetes 版本以及获得的资格和所执行的测试的详细信息。

合作伙伴	产品	文档
Tigera	Calico	安装说明
Isovalent	Cilium	安装说明
Weaveworks	Weave Net	安装说明
VMware	Antrea	安装说明

Amazon EKS 旨在为您提供广泛的选项来涵盖所有使用案例。如果您开发的商业上支持的 Kubernetes CNI 插件未在此处列出，请通过 aws-container-partners@amazon.com 与我们的合作伙伴团队联系以获取更多信息。

安装 Amazon Load Balancer Controller 附加组件

Amazon Load Balancer Controller 管理适用于 Kubernetes 集群的 Amazon 弹性负载均衡器。此控制器预置以下资源：

- 当您创建 Kubernetes `Ingress` 时的 Amazon 应用程序负载均衡器 (ALB, Application Load Balancer)。
- 当您创建 `LoadBalancer` 类型的 Kubernetes 服务时的 Amazon 网络负载均衡器（NLB）。在过去，实例目标使用 Kubernetes 网络负载均衡器，而 IP 目标使用 Amazon 负载均衡器控制器。使用 Amazon Load Balancer Controller 版本 `2.3.0` 或更高版本，您可以使用任一目标类型创建 NLB。有关 NLB 目标类型的更多信息，请参阅 Network Load Balancer 用户指南中的目标类型。

Amazon Load Balancer Controller 控制器以前名为 Amazon ALB 入口控制器。它是托管在 GitHub 上的一个开源项目。本主题描述如何使用原定设置选项安装该控制器。您可以在 GitHub 上查看关于该控制器的完整文档。在部署控制器之前，我们建议您查看 Amazon EKS 上的应用程序负载均衡 (p. 332) 和 Amazon EKS 上的网络负载均衡 (p. 327) 的先决条件和注意事项。这些主题还包括如何部署需要使用 Amazon Load Balancer Controller 来预置 Amazon 应用程序负载均衡器和网络负载均衡器的示例应用程序。

先决条件

- 现有 版 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS (p. 4)。要使用控制器的 `2.4.3` 版本（本主题中使用的版本），您的集群必须为 `1.19` 版或更高版本。如果您的集群早于 `1.19` 版本，则我们建议使用 `2.3.1` 版本。
- 集群的现有 Amazon Identity and Access Management (IAM) OpenID Connect (OIDC) 提供程序。要确定您是否已经拥有一个或是否要创建一个，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。
- 如果您的集群为 `1.21` 版本或更高版本，确保您的 Amazon VPC CNI plugin for Kubernetes、`kube-proxy` 和 CoreDNS 附加组件为服务账户令牌 (p. 395)中列出的最低版本。

要将 Amazon Load Balancer Controller 部署到 Amazon EKS 集群

在以下步骤中，将 *example values* 替换为您自己的值：。如果您的集群早于 1.19 版本，请将 2.4.3 的所有实例更改为 2.3.1，并将 v2_4_3_full.yaml 的所有实例更改为 v2_3_1_full.yaml。

1. 创建一个 IAM policy。

   a. 下载 Amazon Load Balancer Controller 的 IAM policy，该策略允许负载均衡器代表您调用 Amazon API。

   ```
   curl -o iam_policy_cn.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.3/docs/install/iam_policy_cn.json
   ```

   b. 使用上一步中下载的策略创建一个 IAM policy。

   ```
   aws iam create-policy \
       --policy-name AWSLoadBalancerControllerIAMPolicy \
       --policy-document file://iam_policy_cn.json
   ```

   Note

   如果您在 Amazon Web Services Management Console 中查看策略，则可能会看到 ELB 的警告。您可以安全地忽略这些操作，因为有些操作仅适用于 ELB v2。您不会看到 ELB v2 的警告。

2. 创建一个 IAM 角色。在 Amazon Load Balancer Controller 的 kube-system 命名空间中创建名为 aws-load-balancer-controller 的 Kubernetes 服务账户，并使用 IAM 角色的名称注释 Kubernetes 服务账户。

   您可以使用 eksctl 或 Amazon CLI 和 kubectl 创建 IAM 角色和 Kubernetes 服务账户。

   eksctl

   请将 *my-cluster* 替换为您的集群的名称，将 *111122223333* 替换为您的账户 ID，然后运行命令。

   ```
   eksctl create iamserviceaccount \
     --cluster=my-cluster \
     --namespace=kube-system \
     --name=aws-load-balancer-controller \
     --role-name "AmazonEKSLoadBalancerControllerRole" \
     --attach-policy-
   arn=arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
     --approve
   ```

   Amazon CLI and kubectl

   使用 Amazon CLI 和 **kubectl**

   a. 查看集群的 OIDC 提供商 URL。

   ```
   aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer"
    --output text
   ```

   输出示例如下。

   ```
   oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE
   ```

如果没有返回输出，则您必须为集群创建 IAM OIDC 提供程序 (p. 396)。

b. 将以下内容复制到您的设备。请将 *111122223333* 替换为您的账户 ID。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 替换为上一步中返回的输出。替换文本后，运行修改后的命令可创建 `load-balancer-role-trust-policy.json` 文件。

```
cat >load-balancer-role-trust-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com.cn/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com.cn/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
load-balancer-controller"
                }
            }
        }
    ]
}
EOF
```

c. 创建 IAM 角色。

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

d. 将所需的 Amazon EKS 托管 IAM policy 附加到 IAM 角色。请将 *111122223333* 替换为您的账户 ID。

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
  --role-name AmazonEKSLoadBalancerControllerRole
```

e. 将以下内容复制到您的设备。请将 *111122223333* 替换为您的账户 ID。替换文本后，运行修改后的命令可创建 `aws-load-balancer-controller-service-account.yaml` 文件。

```
cat >aws-load-balancer-controller-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn:
 arn:aws:iam::111122223333:role/AmazonEKSLoadBalancerControllerRole
```

```
EOF
```

f. 在集群上创建 Kubernetes 服务账户。使用您创建的名为 *AmazonEKSLoadBalancerControllerRole* 的 IAM 角色注释名为 `aws-load-balancer-controller` 的 Kubernetes 服务账户。

```
kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

3. （可选）配置您的 Kubernetes 服务账户使用的 Amazon Security Token Service 端点类型。有关更多信息，请参阅配置服务账户的 Amazon Security Token Service 端点 (p. 405)。
4. 如果您当前没有安装适用于 Kubernetes 的 Amazon ALB 入口控制器，或者当前安装了 Helm 的 Amazon Load Balancer Controller 的 `0.1.x` 版本，则跳至下一步。

   卸载 Amazon ALB 入口控制器或 Amazon Load Balancer Controller 的 `0.1.x` 版本（仅在安装了 Helm 时）。使用原来安装它的工具完成该过程。Amazon Load Balancer Controller 取代适用于 Kubernetes 的 Amazon ALB 入口控制器的功能。

   Helm

   a. 如果您安装了 `incubator/aws-alb-ingress-controller` Helm 图表，请将其卸载。

   ```
   helm delete aws-alb-ingress-controller -n kube-system
   ```

   b. 如果安装了 `eks-charts/aws-load-balancer-controller` 图表的 `0.1.x` 版本，请将其卸载。由于与 Webhook API 版本不兼容，从 `0.1.x` 升级到版本 `1.0.0` 不起作用。

   ```
   helm delete aws-load-balancer-controller -n kube-system
   ```

   Kubernetes manifest

   a. 检查当前是否安装了该控制器。

   ```
   kubectl get deployment -n kube-system alb-ingress-controller
   ```

   这是未安装控制器情况下的输出。跳至安装控制器 (p. 291)步骤。

   ```
   Error from server (NotFound): deployments.apps "alb-ingress-controller" not found
   ```

   这是安装了控制器情况下的输出。

   ```
   NAME                       READY UP-TO-DATE AVAILABLE AGE
   alb-ingress-controller 1/1   1          1         122d
   ```

   b. 输入以下命令以删除控制器。

   ```
   kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-
   ingress-controller/v1.1.8/docs/examples/alb-ingress-controller.yaml
   kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-
   ingress-controller/v1.1.8/docs/examples/rbac-role.yaml
   ```

   c. 将以下 IAM policy 添加到在前一步骤 (p. 288)中创建的 IAM 角色。该策略允许 Amazon Load Balancer Controller 访问由适用于 Kubernetes 的 ALB 入口控制器创建的资源。

      1. 下载该 IAM policy。您还可以查看策略。

```
curl -o iam_policy_v1_to_v2_additional.json https://raw.githubusercontent.com/
kubernetes-sigs/aws-load-balancer-controller/v2.4.3/docs/install/
iam_policy_v1_to_v2_additional.json
```

2. 将文件中的 `arn:aws:` 替换为 `arn:aws:`。

```
sed -i.bak -e 's|arn:aws:|arn:aws:|' iam_policy_v1_to_v2_additional.json
```

3. 创建 IAM policy 并记下返回的 ARN。

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

4. 将您创建的 IAM policy 附加到您在前一步骤 (p. 288)中创建的 IAM 角色。将 *your-role-name* 替换为角色的名称。如果您使用 `eksctl` 创建该角色，请找到已创建的角色名称，打开 Amazon CloudFormation 控制台并选择 eksctl-*my-cluster*-addon-iamserviceaccount-kube-system-aws-load-balancer-controller 堆栈。选择 Resources 选项卡。角色名称位于 Physical ID（物理 ID）列。如果您使用 Amazon Web Services Management Console 来创建角色，那么角色名称就是您命名的名称，例如 *AmazonEKSLoadBalancerControllerRole*。

```
aws iam attach-role-policy \
  --role-name your-role-name \
  --policy-arn
 arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerAdditionalIAMPolicy
```

5. 使用 Helm V3 (p. 384) 或更高版本或通过应用 Kubernetes 清单来安装 Amazon Load Balancer Controller。如果要在 Fargate 上部署控制器，请使用 Helm 程序，因为它不依赖于 `cert-manager`。

   Helm

   a. 添加 `eks-charts` 存储库。

   ```
   helm repo add eks https://aws.github.io/eks-charts
   ```

   b. 更新您的本地存储库，以确保您拥有最新的图表。

   ```
   helm repo update
   ```

   c. 如果您的节点无权访问 Amazon EKS Amazon ECR 镜像存储库，则需要提取以下容器镜像并将其推送到节点有权访问的存储库。有关如何提取、标记和推送镜像到您自己的存储库的更多信息，请参阅 将容器镜像从一个存储库复制到另一个存储库 (p. 338)。请将 *918309763551* 和 *region-code* 替换为 Amazon 容器镜像注册表 (p. 340) 中列出的 Amazon Web Services 区域 的值。

   ```
   918309763551.dkr.ecr.region-code.amazonaws.com.cn/amazon/aws-load-balancer-
   controller:v2.4.3
   ```

   d. 安装 Amazon Load Balancer Controller。如果要将控制器部署到被限制访问 Amazon EC2 实例元数据服务 (IMDS) 的 Amazon EC2 节点，或者部署到 Fargate 节点，则需要在以下 `helm` 命令中添加以下标志：

   - **`--set region=`*`region-code`***
   - **`--set vpcId=`*`vpc-xxxxxxxx`***

   如果要部署除 `us-west-2` 外的 Amazon Web Services 区域，则将以下标志添加到 `helm` 命令，以将 *918309763551* 和 *region-code* 替换为 Amazon 容器镜像注册表 (p. 340) 中列出

的您的 Amazon Web Services 区域 的值。如果您已提取镜像并将其推送到自己的存储库中，请将完整的注册表和存储库替换为您自己的注册表和存储库。

```
--set image.repository=918309763551.dkr.ecr.region-code.amazonaws.com.cn/amazon/
aws-load-balancer-controller
```

将 *my-cluster* 替换为您自己的值。在以下命令中，aws-load-balancer-controller 是您在上一步中创建的 Kubernetes 服务账户。

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=my-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --set enableShield=false \
  --set enableWaf=false \
  --set enableWafv2=false
```

Important

已部署的图表不会自动接收安全更新。当新图表可用时，您需要手动升级到新图表。升级时，请将 *install* 更改为上一个命令中的 **upgrade**，但在运行上一个命令前，运行以下命令以安装 TargetGroupBinding 自定义资源定义。

```
kubectl apply -k "github.com/aws/eks-charts/stable/aws-load-balancer-
controller/crds?ref=master"
```

Kubernetes manifest

a. 使用以下方法之一安装 cert-manager 以将证书配置注入 Webhook。

- 如果您的节点有权访问 quay.io 容器注册表，请安装 cert-manager 以将证书配置注入 Webhook。

```
kubectl apply \
    --validate=false \
    -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert-
manager.yaml
```

- 如果您的节点无权访问 quay.io，请完成以下任务：

i. 下载清单。

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/
download/v1.5.4/cert-manager.yaml
```

ii. 请提取以下镜像并将其推送到节点有权访问的存储库。有关如何提取、标记和推送镜像到您自己的存储库的更多信息，请参阅 将容器镜像从一个存储库复制到另一个存储库 (p. 338)。

```
quay.io/jetstack/cert-manager-cainjector:v1.5.4
quay.io/jetstack/cert-manager-controller:v1.5.4
quay.io/jetstack/cert-manager-webhook:v1.5.4
```

iii. 请将三个镜像的清单中的 quay.io 替换为您自己的注册表名称。以下命令假定您的私有存储库名称与源存储库的名称相同。将 *111122223333.dkr.ecr.region-code.amazonaws.com.cn* 替换为您的私有注册表。

```
sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-
code.amazonaws.com.cn|' ./cert-manager.yaml
```

iv. 运用该清单。

```
kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

b. 安装控制器。

i. 下载控制器规范。有关控制器的更多信息，请参阅 GitHub 上的文档。

```
curl -Lo v2_4_3_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-
controller/releases/download/v2.4.3/v2_4_3_full.yaml
```

ii. 对文件进行以下编辑。

- 如果您已下载 `v2_4_3_full.yaml` 文件，请运行以下命令以删除清单中的 `ServiceAccount` 部分。如果您不删除此部分，系统将覆盖您在上一步中对服务账户所做的必需注释。如果删除此部分，若您删除了控制器，系统还会保留您在上一步中创建的服务账户。

```
sed -i.bak -e '480,488d' ./v2_4_3_full.yaml
```

如果您已下载其他文件版本，请在编辑器中打开此文件，然后删除以下行。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
---
```

- 将文件 `Deployment spec` 部分的 *`your-cluster-name`* 替换为您的集群的名称，然后在 `--ingress-class=alb` 下添加 **`following parameters`**。

```
...
spec:
      containers:
        - args:
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --enable-shield=false
            - --enable-waf=false
            - --enable-wafv2=false


...
```

- 如果要将控制器部署到被限制访问 Amazon EC2 实例元数据服务 (IMDS) 的 Amazon EC2 节点，或者部署到 Fargate，则需要在 **`following parameters`** 下添加 `- args:`：

```
...
spec:
      containers:
        - args:
```

```
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            -  --enable-shield=false
            - --enable-waf=false
            - --enable-wafv2=false
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code


...
```

iii. 应用文件。

```
kubectl apply -f v2_4_3_full.yaml
```

iv. 将 `IngressClass` 和 `IngressClassParams` 清单应用于集群。

```
curl -Lo v2_4_3_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.4.3/v2_4_3_ingclass.yaml
```

v. 将清单应用于集群。

```
kubectl apply -f v2_4_3_ingclass.yaml
```

6. 验证控制器是否已安装。

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

输出示例如下。

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

如果使用 Helm 进行部署，则会收到之前的输出结果。如果您使用 Kubernetes 清单进行部署，则只有一个副本。

7. 在使用控制器预置Amazon资源前，您的集群必须满足特定要求。有关更多信息，请参阅 Amazon EKS 上的应用程序负载均衡 (p. 332) 和 Amazon EKS 上的网络负载均衡 (p. 327)。

管理 CoreDNS 附加组件

CoreDNS 是一个灵活、可扩展的 DNS 服务器，可用作 Kubernetes 集群 DNS。当您启动具有至少一个节点的 Amazon EKS 集群时，无论集群中部署的节点数量如何，预设情况下都会部署 CoreDNS 镜像的两个副本。这些 CoreDNS pods 为集群中的所有 pods 提供名称解析。如果您的集群包含命名空间与 CoreDNS `deployment` 的命名空间相匹配的 Amazon Fargate 配置文件 (p. 131) 时，则可以将 CoreDNS pods 容器部署到 Fargate 节点。有关 CoreDNS 的更多信息，请参阅 Kubernetes 文档中的使用 CoreDNS 进行服务发现。

下表列出了随每个 Amazon EKS 集群版本一起部署的 CoreDNS 附加组件的版本。

与每个 Amazon EKS 支持的集群版本一起部署的 CoreDNS 版本

Kubernetes 版本	1.23	1.22	1.21	1.20	1.19	1.18
CoreDNS	1.8.7-eksbuild.2	1.8.7	1.8.4	1.8.3	1.8.0	1.7.0

如果您使用 Amazon Web Services Management Console 创建了 `1.18` 版或更高版本的集群，则 Amazon EKS 将附加组件安装为 Amazon EKS 附加组件。如果您使用任何工具创建了 `1.17` 版本或更低版本的集群，或者您使用 Amazon Web Services Management Console 以外的其他工具创建了 `1.18` 版本或更高版本的集群，则 Amazon EKS 将插件安装为自行管理的附加组件。

您可以按照 添加 CoreDNS Amazon EKS 附加组件 (p. 295) 中的步骤将自行管理的附加组件迁移到 Amazon EKS 附加组件。如果您已将 CoreDNS 附加组件添加到集群，则您可以按照 更新 CoreDNS Amazon EKS 附加组件 (p. 296) 和 删除 CoreDNS Amazon EKS 附加组件 (p. 298) 部分中的程序来管理该附加组件。有关 Amazon EKS 附加组件的更多信息，请参阅 Amazon EKS 附加组件 (p. 341)。

如果您尚未添加 CoreDNS Amazon EKS 附加组件，则 CoreDNS 自行管理的附加组件仍在您的集群上运行。您可以使用 更新 CoreDNS 自行管理的附加组件 (p. 299) 中的程序更新 CoreDNS 自行管理附加组件。

先决条件

- 现有 版 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS (p. 4)。
- 如果您的集群为 `1.21` 版本或更高版本，确保您的 Amazon VPC CNI plugin for Kubernetes 和 `kube-proxy` 附加组件为服务账户令牌 (p. 395)中列出的最低版本。

添加 CoreDNS Amazon EKS 附加组件

选择带有您想要用于将 CoreDNS Amazon EKS 附加组件添加到集群的工具名称的选项卡。

> **Important**
>
> 在添加 CoreDNS Amazon EKS 附加组件之前，请确认您不会自行管理 Amazon EKS 将开始管理的任何设置。要确定 Amazon EKS 管理的设置，请参阅 Amazon EKS 附加组件配置 (p. 342)。

eksctl

使用 **`eksctl`** 添加 CoreDNSAmazon EKS 附加组件

将 *`my-cluster`* 替换为集群名称，然后运行以下命令。

```
eksctl create addon --name coredns --cluster my-cluster --force
```

如果您删除 `--force` 选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么添加 Amazon EKS 附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在指定此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置，因为这些设置会被此选项覆盖。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 添加 CoreDNS Amazon EKS 附加组件

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择 Clusters（集群），然后选择要为其配置 CoreDNS Amazon EKS 附加组件的集群的名称。
3. 选择 Add-ons（附加组件）选项卡。
4. 选择 Add new（新增）。
   a. 对于 Name（名称），选择 CoreDNS。
   b. 选择您想使用的 Version（版本）。

c. 如果您选择 Override existing configuration for this add-on on the cluster（覆盖集群上此附加组件的现有配置），那么 Amazon EKS 附加组件的设置可以覆盖现有附加组件的任何设置。如果您未启用此选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么添加 Amazon EKS 附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在选择此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

d. 选择 Add（添加）。

Amazon CLI

使用 Amazon CLI 添加 CoreDNS Amazon EKS 附加组件

将 *my-cluster* 替换为集群名称，然后运行以下命令。

```
aws eks create-addon \
    --cluster-name my-cluster \
    --addon-name coredns \
    --resolve-conflicts OVERWRITE
```

如果您删除 `--resolve-conflicts OVERWRITE` 选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么创建附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在指定此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置，因为这些设置会被此选项覆盖。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

更新 CoreDNS Amazon EKS 附加组件

本程序用于更新 CoreDNS Amazon EKS 附加组件。如果您尚未添加 CoreDNS Amazon EKS 附加组件，请改为按照 更新 CoreDNS 自行管理的附加组件 (p. 299) 中的程序操作，或添加 CoreDNS Amazon EKS 附加组件 (p. 295)。当新版本发布之后或您将集群更新 (p. 28)到一个新的 Kubernetes 次要版本后，Amazon EKS 不会自动更新集群上的 CoreDNS。要更新现有集群上的 CoreDNS，您必须启动更新，然后 Amazon EKS 会为您更新 Amazon EKS 附加组件。

eksctl

使用 **`eksctl`** 更新 CoreDNS Amazon EKS 附加组件

1. 检查 `coredns` Amazon EKS 附加组件的当前版本。将 *my-cluster* 替换为您的集群名称。

```
eksctl get addon --name coredns --cluster my-cluster
```

输出示例如下。

```
NAME            VERSION                 STATUS  ISSUES  IAMROLE UPDATE AVAILABLE
coredns          v1.8.0-eksbuild.1      ACTIVE  0               v1.8.3-eksbuild.1
```

2. 将附加组件更新为上一个步骤的输出中的 `UPDATE AVAILABLE` 下返回的版本。

```
eksctl update addon \
    --name coredns \
    --version v1.8.3-eksbuild.1 \
    --cluster my-cluster \
    --force
```

如果您删除 `--force` 选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么更新 Amazon EKS 附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在指定此选项之

前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置，因为这些设置会被此选项覆盖。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 更新 CoreDNS Amazon EKS 附加组件

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择 Clusters（集群），然后选择要为其更新 CoreDNS Amazon EKS 附加组件的集群的名称。
3. 选择 Add-ons（附加组件）选项卡。
4. 选择 coredns 框右上角中的单选按钮，然后选择 Edit（编辑）。
   a. 选择您要使用的 Amazon EKS 附加组件的 Version（版本）。
   b. 如果您选择 Override existing configuration for this add-on on the cluster（覆盖集群上此附加组件的现有配置），那么 Amazon EKS 附加组件的设置可以覆盖现有附加组件的任何设置。如果您未启用此选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么更新 Amazon EKS 附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在选择此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。
   c. 选择 Update（更新）。

Amazon CLI

使用 Amazon CLI 更新 CoreDNS Amazon EKS 附加组件

1. 检查 CoreDNS 附加组件的当前版本。将 *my-cluster* 替换为您的集群名称。

```
aws eks describe-addon \
    --cluster-name my-cluster \
    --addon-name coredns \
    --query "addon.addonVersion" \
    --output text
```

输出示例如下。

```
1.7.0
```

2. 确定哪些版本的 CoreDNS 附加组件可用于您的集群版本。

```
aws eks describe-addon-versions \
    --addon-name coredns \
    --kubernetes-version 1.19 \
    --query "addons[].addonVersions[].[addonVersion,
 compatibilities[].defaultVersion]" \
    --output text
```

输出示例如下。

```
1.8.0
True
1.7.0
False
```

下面带有 `True` 的版本是使用新集群部署的默认版本。

3. 将附加组件更新为上一个步骤的输出中返回的带有 `True` 的版本。您也可以更新到更高版本（如果输出中返回了）。

```
aws eks update-addon \
    --cluster-name my-cluster \
    --addon-name coredns \
    --addon-version 1.8.0 \
    --resolve-conflicts OVERWRITE
```

如果您删除 `--resolve-conflicts OVERWRITE` 选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么更新附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在指定此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置，因为这些设置会被此选项覆盖。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

删除 CoreDNS Amazon EKS 附加组件

移除 Amazon EKS 附加组件时，您有两种选择：

- 在集群上保留附加组件的软件：此选项删除了 Amazon EKS 对任何设置和功能的管理，使 Amazon EKS 能够在您启动更新后向您发出更新通知并自动更新 Amazon EKS 附加组件，但会将附加组件的软件保留在您的集群上。此选项使附加组件成为自我管理的附加组件，而不是 Amazon EKS 附加组件。附加组件没有停机时间。
- 从集群中完全删除附加组件：只有在集群中没有资源依赖于附加组件提供的功能时，才应从集群中移除 Amazon EKS 附加组件。删除 Amazon EKS 附加组件后，如有需要，您可以再次添加它。

如果附加组件有与其关联的 IAM 账户，则不会移除该 IAM 账户。

选择带有要用于将 CoreDNS Amazon EKS 附加组件从 `1.18` 或更高版本的集群中移除的工具名称的选项卡。

eksctl

使用 **`eksctl`** 删除 CoreDNS Amazon EKS 附加组件

将 *`my-cluster`* 替换为集群名称，然后运行以下命令。移除 `--preserve` 会从您的集群中移除附加组件软件。

```
eksctl delete addon --cluster my-cluster --name coredns --preserve
```

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 删除 CoreDNS Amazon EKS 附加组件

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择 Clusters（集群），然后选择要删除其 CoreDNS Amazon EKS 附加组件的集群的名称。
3. 选择 Add-ons（附加组件）选项卡。
4. 选择 coredns 框右上角中的单选按钮，然后选择 Remove（删除）。键入 **`coredns`**，然后选择 Remove（删除）。

Amazon CLI

使用 Amazon CLI 删除 CoreDNS Amazon EKS 附加组件

将 *my-cluster* 替换为集群名称，然后运行以下命令。移除 --preserve 会从您的集群中移除附加组件软件。

```
aws eks delete-addon --cluster-name my-cluster --addon-name coredns --preserve
```

更新 CoreDNS 自行管理的附加组件

如果您有一个集群，但尚未向其添加 CoreDNS Amazon EKS 附加组件，请完成以下步骤来更新附加组件。如果您已添加了 CoreDNS Amazon EKS 附加组件，请改为按照 更新 CoreDNS Amazon EKS 附加组件 (p. 296) 中的程序操作。

要使用 **kubectl** 更新 CoreDNS 自行管理的附加组件

1. 检查 CoreDNS 附加组件的当前版本。

```
kubectl describe deployment coredns \
    --namespace kube-system \
    | grep Image \
    | cut -d "/" -f 3
```

输出示例如下。

```
coredns:v1.8.0
```

2. 如果您的当前 CoreDNS 版本是 1.5.0 或更高版本，但早于 CoreDNS 版本 (p. 294)表中列出的版本，请跳过此步骤。如果您的当前版本低于 1.5.0，则需要修改 ConfigMap，以使 CoreDNS 使用转发附加组件，而不是代理附加组件。

   a. 使用以下命令打开 configmap。

```
kubectl edit configmap coredns -n kube-system
```

   b. 将以下行中的 proxy 替换为 forward。保存文件，然后退出编辑器。

```
proxy . /etc/resolv.conf
```

3. 如果您最初在 Kubernetes 1.17 或更早版本上部署了集群，那么您可能需要从 CoreDNS 清单中删除已停用的术语。

   Important

   在升级到 CoreDNS 版本 1.7.0 之前，您必须完成此步骤，但是即使您要升级到早期版本，我们也建议您完成此步骤。

   a. 检查您的 CoreDNS 清单是否包含该行。

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' | grep
 upstream
```

   如果未返回任何输出，则说明您的清单没有此类行，您可以跳到下一步更新 CoreDNS。如果返回了输出，则需要删除该行。

   b. 使用以下命令编辑 ConfigMap，删除文件中包含词语 upstream 的该行。不要更改文件中的任何其他内容。删除该行后，保存更改。

```
kubectl edit configmap coredns -n kube-system -o yaml
```

4. 检索您当前的 CoreDNS 映像：

```
kubectl get deployment coredns \
    --namespace kube-system \
    -o=jsonpath='{$.spec.template.spec.containers[:1].image}'
```

5. 如果您要更新到 CoreDNS 1.8.3 或更高版本，则需要将 endpointslices 权限添加到 system:coredns Kubernetes clusterrole。

```
kubectl edit clusterrole system:coredns -n kube-system
```

在文件中的 rules 部分的现有权限行下添加以下行。

```
...
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
...
```

6. 通过更新 CoreDNS 附加组件将 *918309763551* 和 *region-code* 替换为在上一步返回的输出中的值。将 *1.8.7-eksbuild.2* 替换为您的集群的推荐 CoreDNS 版本 (p. 294)或更高版本：

```
kubectl set image --namespace kube-system deployment.apps/coredns \
    coredns=918309763551.dkr.ecr.region-code.amazonaws.com.cn/eks/coredns:v1.8.7-
eksbuild.2
```

管理 kube-proxy 附加组件

Kube-proxy 在每个 Amazon EC2 节点上维护网络规则。它能够实现与您的 pods 进行网络通信。Kube-proxy 未部署到 Fargate 节点。有关更多信息，请参阅 Kubernetes文档中的 kube-proxy。每个 Kubernetes 版本都有两种类型的 kube-proxy 映像可用：

- 默认：默认使用新集群部署此类型。这是可以与 Amazon EKS 附加组件配合使用的唯一类型，可供 1.22 版本和更低版本的集群使用。
- 最低：与默认类型不同，此映像类型基于 Amazon EKS Distro 维护的最低要求基本映像，其中包含最低要求的程序包并且没有 shell。有关更多信息，请参阅 Amazon EKS Distro。此类型映像可作为自行管理的附加组件提供。它也可作为 Amazon EKS 附加组件供 1.23 版本和更高版本的集群使用，但不能作为 Amazon EKS 附加组件供 1.22 版本和更低版本的集群使用。如果您选择将此类型的附加组件安装为自行管理的附加组件，以供 1.22 版本和更低版本的集群使用，请完成 the section called “更新 自行管理的附加组件” (p. 305) 中的步骤，并为下表中的最低要求类型指定最新版本。默认情况下，此类型会添加到任何新版本 1.23 以及更高版本的集群。如果将现有集群更新到版本 1.23，您可以将 Amazon EKS 附加组件添加 (p. 301)到集群，或将现有 Amazon EKS 附加组件更新 (p. 302)为此类型。

每个 Amazon EKS 集群版本的最新可用的 **kube-proxy** 版本

Image type	1.23	1.22	1.21	1.20	1.19	1.18
kube-proxy（默认类型）	不可用	1.22.11-eksbuild.2	1.21.14-eksbuild.2	1.20.15-eksbuild.2	1.19.16-eksbuild.2	1.18.8-eksbuild.1
kube-proxy（最低要求类型）	1.23.7-minimal-eksbuild.1	1.22.11-minimal-eksbuild.1	1.21.14-minimal-eksbuild.2	1.20.15-minimal-eksbuild.2	1.19.16-minimal-eksbuild.3	1.18.20-minimal-eksbuild.3

如果您使用 Amazon Web Services Management Console 创建了 1.18 版或更高版本的集群，则 Amazon EKS 已为您安装插件作为 Amazon EKS 附加组件。如果您使用任何工具创建了 1.17 版本或更低版本的集群，或者您使用 Amazon Web Services Management Console 以外的其他工具创建了 1.18 版本或更高版本的集群，则 Amazon EKS 将插件安装为自行管理的附加组件。您可以按照 添加 kube-proxy Amazon EKS 附加组件 (p. 301) 中的步骤将自行管理的附加组件迁移到 Amazon EKS 附加组件。如果您已将 kube-proxy 附加组件添加到集群，则您可以按照 更新 kube-proxy Amazon EKS 附加组件 (p. 302) 和 删除 kube-proxy Amazon EKS 附加组件 (p. 304) 部分中的程序来管理该附加组件。有关 Amazon EKS 附加组件的更多信息，请参阅 Amazon EKS 附加组件 (p. 341)。

如果您尚未添加 kube-proxy Amazon EKS 附加组件，则 kube-proxy 自行管理的附加组件仍在您的集群上运行。您可以按照 更新 kube-proxy 自行管理的附加组件 (p. 305) 中的程序手动更新 kube-proxy 自行管理附加组件。

与 Kubernetes 的 kube-proxy 一样，Amazon EKS 也有相同的兼容性和偏斜策略。Kube-proxy 必须满足以下要求：

- 它必须与节点上的 kubelet 具有相同的次要版本。
- 它不得比集群控制面板的版本新。
- 它的版本不得比控制面板的版本低两个次要版本。例如，如果您的控制面板正在运行 Kubernetes 1.23，kube-proxy 次要版本不能低于 1.21。

先决条件

- 现有 版 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS (p. 4)。
- 如果您的集群为 1.21 版本或更高版本，确保您的 Amazon VPC 和 CoreDNS 附加组件为服务账户令牌 (p. 395)中列出的最低版本。

添加 kube-proxy Amazon EKS 附加组件

您可以使用 eksctl、Amazon Web Services Management Console 或 Amazon CLI 将 kube-proxy Amazon EKS 附加组件添加到集群。

> **Important**
>
> 在添加 kube-proxy Amazon EKS 附加组件之前，请确认您不会自行管理 Amazon EKS 将开始管理的任何设置。要确定 Amazon EKS 管理的设置，请参阅 Amazon EKS 附加组件配置 (p. 342)。

eksctl

使用 **eksctl** 添加 **kube-proxy**Amazon EKS 附加组件

将 *my-cluster* 替换为您集群的名称，然后运行以下命令。Eksctl 添加可用于集群版本的最新版本附加组件。

```
eksctl create addon --name kube-proxy --cluster my-cluster --force
```

如果您删除 `--force` 选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么添加 Amazon EKS 附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在指定此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置，因为这些设置会被此选项覆盖。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 添加 **kube-proxy** Amazon EKS 附加组件

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择 Clusters（集群），然后选择要为其配置 `kube-proxy` Amazon EKS 附加组件的集群的名称。
3. 选择 Add-ons（附加组件）选项卡。
4. 选择 Add new（新增）。
   a. 对于 Name（名称），请选择 **kube-proxy**。
   b. 选择您想使用的 Version（版本）。选择标记为 Latest（最新）的版本。
   c. 如果您选择 Override existing configuration for this add-on on the cluster（覆盖集群上此附加组件的现有配置），那么 Amazon EKS 附加组件的设置可以覆盖现有附加组件的任何设置。如果您未启用此选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么添加 Amazon EKS 附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在选择此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。
   d. 选择 Add（添加）。

Amazon CLI

使用 Amazon CLI 添加 **kube-proxy** Amazon EKS 附加组件

创建附加组件。将 *my-cluster* 替换为您集群的名称，将 *1.23.7-minimal-eksbuild.1* 替换为最新版本表 (p. 300)中适用于您集群的最新版本。

```
aws eks create-addon --cluster-name my-cluster --addon-name kube-proxy --addon-version
 v1.23.7-minimal-eksbuild.1 --resolve-conflicts OVERWRITE
```

如果您删除 `--resolve-conflicts OVERWRITE` 选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么创建附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在指定此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置，因为这些设置会被此选项覆盖。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

更新 kube-proxy Amazon EKS 附加组件

本程序用于更新 `kube-proxy` Amazon EKS 附加组件。如果您尚未添加 `kube-proxy` Amazon EKS 附加组件，请改为按照 更新 kube-proxy 自行管理的附加组件 (p. 305) 中的程序操作。当新版本发布之后或您将集群更新 (p. 28)到一个新的 Kubernetes 次要版本后，Amazon EKS 不会自动更新集群上的 `kube-proxy`。要更新现有集群上的 `kube-proxy`，您必须启动更新，然后 Amazon EKS 会为您更新该附加组件。

Important

在将 `kube-proxy` 更新为与已更新的集群次要版本相同的次要版本前，请将集群和节点更新到新的 Kubernetes 次要版本。

eksctl

使用 **eksctl** 更新 **kube-proxy** Amazon EKS 附加组件

1. 检查 kube-proxy Amazon EKS 附加组件的当前版本。将 *my-cluster* 替换为您的集群名称。

```
eksctl get addon --name kube-proxy --cluster my-cluster
```

输出示例如下。

```
NAME            VERSION                 STATUS  ISSUES  IAMROLE UPDATE AVAILABLE
kube-proxy      v1.22.11-eksbuild.2      ACTIVE  0               v1.23.7-minimal-
eksbuild.1
```

2. 将附加组件更新为上一个步骤的输出中的 UPDATE AVAILABLE 下返回的版本。

```
eksctl update addon \
    --name kube-proxy \
    --version v1.23.7-minimal-eksbuild.1 \
    --cluster my-cluster \
    --force
```

如果您删除 --force 选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么更新附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在指定此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置，因为这些设置会被此选项覆盖。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 更新 **kube-proxy** Amazon EKS 附加组件

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择 Clusters（集群），然后选择要为其更新 kube-proxy Amazon EKS 附加组件的集群的名称。
3. 选择 Add-ons（附加组件）选项卡。
4. 选择 **kube-proxy** 框右上角中的框，然后选择 Edit（编辑）。
   a. 选择标记为 Latest（最新）的 Version（版本）。
   b. 如果您选择 Override existing configuration for this add-on on the cluster（覆盖集群上此附加组件的现有配置），那么 Amazon EKS 附加组件的设置可以覆盖现有附加组件的任何设置。如果您未启用此选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么更新 Amazon EKS 附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在选择此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。
   c. 选择 Update（更新）。

Amazon CLI

使用 Amazon CLI 更新 **kube-proxy** Amazon EKS 附加组件

1. 检查 kube-proxy Amazon EKS 附加组件的当前版本。将 *my-cluster* 替换为您的集群名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
 "addon.addonVersion" --output text
```

输出示例如下。

```
v1.22.11-eksbuild.2
```

2. 确定哪些版本的 kube-proxy 附加组件可用于您的集群版本。将 *1.23* 替换为集群版本。

```
aws eks describe-addon-versions --addon-name kube-proxy --kubernetes-version 1.23 \
    --query "addons[].addonVersions[].[addonVersion,
 compatibilities[].defaultVersion]" --output text
```

输出示例如下。

```
v1.23.7-minimal-eksbuild.1
False
v1.22.11-eksbuild.2
True
...
```

下面带有 True 的版本是使用您指定的 Kubernetes 版本的新集群部署的默认版本。

3. 将附加组件更新为上一步输出中返回的最新版本。

```
aws eks update-addon --cluster-name my-cluster --addon-name kube-proxy --addon-
version v1.23.7-minimal-eksbuild.1 --resolve-conflicts OVERWRITE
```

如果您删除 --resolve-conflicts OVERWRITE 选项，并且任何 Amazon EKS 附加组件设置与您的现有设置冲突，那么更新附加组件将会失败，而且您会收到一条帮助您解决冲突的错误消息。在指定此选项之前，请确保 Amazon EKS 附加组件不会管理您需要管理的设置，因为这些设置会被此选项覆盖。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Amazon EKS 附加组件配置 (p. 342)。

删除 kube-proxy Amazon EKS 附加组件

移除 Amazon EKS 附加组件时，您有两种选择：

- 在集群上保留附加组件的软件：此选项删除了 Amazon EKS 对任何设置和功能的管理，使 Amazon EKS 能够在您启动更新后向您发出更新通知并自动更新 Amazon EKS 附加组件，但会将附加组件的软件保留在您的集群上。此选项使附加组件成为自我管理的附加组件，而不是 Amazon EKS 附加组件。附加组件没有停机时间。
- 从集群中完全删除附加组件：只有在集群中没有资源依赖于附加组件提供的功能时，才应从集群中移除 Amazon EKS 附加组件。删除 Amazon EKS 附加组件后，如有需要，您可以再次添加它。

如果附加组件有与其关联的 IAM 账户，则不会移除该 IAM 账户。

您可以使用 eksctl、Amazon Web Services Management Console 或 Amazon CLI 将 kube-proxy Amazon EKS 附加组件从集群中删除。

eksctl

使用 **eksctl** 删除 **kube-proxy** Amazon EKS 附加组件

将 *my-cluster* 替换为集群名称，然后运行以下命令。移除 --preserve 会从您的集群中移除附加组件软件。

```
eksctl delete addon --cluster my-cluster --name kube-proxy --preserve
```

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 删除 **kube-proxy** Amazon EKS 附加组件

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择 Clusters（集群），然后选择要删除其 kube-proxy Amazon EKS 附加组件的集群的名称。
3. 选择 Add-ons（附加组件）选项卡。
4. 选择 **kube-proxy** 框的右上角的复选框，然后选择 Remove（删除）。键入 **kube-proxy**，然后选择 Remove（删除）。

Amazon CLI

使用 Amazon CLI 删除 **kube-proxy** Amazon EKS 附加组件

将 *my-cluster* 替换为集群名称，然后运行以下命令。移除 --preserve 会从您的集群中移除附加组件软件。

```
aws eks delete-addon --cluster-name my-cluster --addon-name kube-proxy --preserve
```

更新 kube-proxy 自行管理的附加组件

如果您有一个集群，但尚未向其添加 kube-proxy Amazon EKS 附加组件，请完成以下步骤来更新自行管理式附加组件。如果您已添加了 kube-proxy Amazon EKS 附加组件，请改为按照 更新 kube-proxy Amazon EKS 附加组件 (p. 302) 中的程序操作。

Important

在将 kube-proxy 更新为与已更新的集群次要版本相同的次要版本前，请将集群和节点更新到新的 Kubernetes 次要版本。

要使用 **kubectl** 更新 **kube-proxy** 自行管理的附加组件

1. 检查 kube-proxy 部署的当前版本。

```
kubectl get daemonset kube-proxy --namespace kube-system -
o=jsonpath='{$.spec.template.spec.containers[:1].image}'
```

输出示例如下。

```
918309763551.dkr.ecr.region-code.amazonaws.com.cn/eks/kube-proxy:v1.22.11-eksbuild.2
```

2. 通过将 *918309763551* 和 *region-code* 替换为输出的值，更新 kube-proxy 附加组件。将 *1.22.11-eksbuild.2* 替换为每个 Amazon EKS 集群版本的最新可用 kube-proxy 版本 (p. 300)表中所列的适用于您集群版本的 kube-proxy 版本。您可以指定默认或最低要求映像类型的版本号。

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-
proxy=918309763551.dkr.ecr.region-code.amazonaws.com.cn/eks/kube-proxy:v1.23.7-minimal-
eksbuild.1
```

3. 如果您在同一集群中使用 x86 和 Arm 节点，而且您的集群是在 2020 年 8 月 17 日之前部署的。那么请使用以下命令编辑您的 kube-proxy 清单以将多个硬件架构的节点选择器纳入其中。您只需执行此操作一次。将选择器添加到清单中之后，您无需在每次更新时都进行添加。如果您的集群是在 2020 年 8 月 17 日或之后部署的，则 kube-proxy 已经具有多架构功能。

```
kubectl edit -n kube-system daemonset/kube-proxy
```

将以下节点选择器添加到编辑器中的文件中，然后保存文件。有关在编辑器的何处纳入此文本的示例，请参阅 GitHub 上的 CNI 清单文件。这使 Kubernetes 能够根据节点的硬件架构提取正确的硬件镜像。

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

4. 如果您的集群最初是使用 Kubernetes 版本 1.14 或更高版本创建的，则可以跳过此步骤，因为 kube-proxy 已经包含此 Affinity Rule。如果您的 Amazon EKS 集群最初是使用 Kubernetes 版本 1.13 或更早版本创建的，并打算使用 Fargate 节点，请编辑 kube-proxy 清单以将 NodeAffinity 规则纳入其中，以防从 Fargate 节点上调度 kube-proxy pods。您只需执行此编辑一次。将 Affinity Rule 添加到清单之后，您无需在每次升级集群时都进行添加。编辑您的 kube-proxy DaemonSet。

```
kubectl edit -n kube-system daemonset/kube-proxy
```

将以下 Affinity Rule 添加到编辑器中文件中的 DaemonSet spec 部分，然后保存文件。有关在编辑器的何处纳入此文本的示例，请参阅 GitHub 上的 CNI 清单文件。

```
- key: eks.amazonaws.com/compute-type
  operator: NotIn
  values:
  - fargate
```

安装 Calico 附加组件

Project Calico 是一种适用于 Kubernetes 的网络策略引擎。通过强制实施 Calico 网络策略，您可以实现网络隔离和租户隔离。在必须隔离各个租户的多租户环境中，或者当您要针对开发、暂存和生产创建单独的环境时，这非常有用。网络策略类似于 Amazon 安全组的一点是，您可以创建网络传入和传出规则。您使用 pod 选择器和标签向 pods 分配网络策略，而不是向安全组分配实例。

注意事项

- 当将 Fargate 与 Amazon EKS 一起使用时不支持 Calico。
- Calico 将规则添加到节点上的 iptables，这些规则的优先级可能高于您已经在 Calico 之外实施的现有规则。考虑将现有 iptables 规则添加到您的 Calico 策略，以避免 Calico 策略之外的规则被 Calico 覆盖。
- 如果您使用 Amazon VPC CNI 附加组件版本 1.10 或更早的版本，则分支网络接口上流入 pods 的 pods 安全组 (p. 273)流量将不受 Calico 网络策略执行的约束，并且仅受到 Amazon EC2 安全组执行的限制。如果您使用的是 1.11.0 或更高版本的 Amazon VPC CNII 附加组件，如您为 Amazon VPC CNI 附加组件设置 POD_SECURITY_GROUP_ENFORCING_MODE=standard，则分支网络接口上流入 pods 的流量将受 Calico 网络策略执行的约束。

先决条件

- 现有 版 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS (p. 4)。

- 您的设备或 Amazon CloudShell 上安装了 `kubectl` 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 `1.22`，则可以将 `kubectl` 版本 `1.21`、`1.22` 或 `1.23` 用于它。要安装或升级 `kubectl`，请参阅 安装或更新 kubectl (p. 4)。

以下过程显示如何在 Amazon EKS 集群中的 Linux 节点上安装 Calico。要在 Windows 节点上安装 Calico，请参阅在 Amazon EKS Windows 容器上使用 Calico。

在 Amazon EKS Linux 节点上安装 Calico

Important

Amazon EKS 不维护以下过程中使用的清单或图表。在 Amazon EKS 上安装 Calico 的推荐方法是使用 Calico Operator 而不是这些图表或清单。有关更多信息，请参阅 GitHub 上的重要公告：Amazon EKS 将不再维护和更新此存储库中的 Calico 图表。如果在安装和使用 Calico 过程中遇到问题，请直接将问题提交给 Calico Operator 和 Calico 项目。在将任何新的 Calico Operator 和 Calico 版本安装到集群之前，您应始终与 Tigera 联系以了解兼容性。

您可以使用 Helm 程序或清单来安装 Calico。

Helm

先决条件

在您的计算机上安装 Helm 版本 `3.0` 或更高版本。要安装或升级 Helm，请参阅 将 Helm 与 Amazon EKS 结合使用 (p. 384)。

要使用 Helm 安装 Calico

1. 将 Project Calico 添加到 Helm 存储库中。

```
helm repo add projectcalico https://docs.projectcalico.org/charts
```

2. 如果您已经添加了 Calico，则可能需要更新它以获得最新发布的版本。

```
helm repo update
```

3. 请安装 `3.21.4` 或更高版本的 Tigera Calico Operator 和自定义资源定义。

```
helm install calico projectcalico/tigera-operator --version v3.21.4
```

4. 在 `tigera-operator` 命名空间中查看资源。

```
kubectl get all -n tigera-operator
```

输出示例如下。

`replicaset` 的 `DESIRED` 和 `READY` 列中的值应该相一致。返回的值与以下输出中的 *`values`* 不同。

```
NAME                                       READY   STATUS    RESTARTS   AGE
pod/tigera-operator-c4b9549c7-h4zp5        1/1     Running   0          110m
...
NAME                                             DESIRED   CURRENT   READY   AGE
replicaset.apps/tigera-operator-c4b9549c7        1         1         1       2m35s
```

5. 在 `calico-system` 命名空间中查看资源。

```
kubectl get all -n calico-system
```

输出示例如下。

calico-node DaemonSet 的 DESIRED和 READY 列中的值应该相一致。两个 replicasets 的 DESIRED 和 READY 列中的值也应该一致。返回的值与以下输出中的 *values* 不同。

```
NAME                                           READY   STATUS    RESTARTS   AGE
pod/calico-kube-controllers-579b45dcf-z5tsf    1/1     Running   0          100m
pod/calico-node-v9dhf                          1/1     Running   0          100m
pod/calico-typha-6f9c6786d-f2mc7               1/1     Running   0          100m
...
NAME                                     DESIRED   CURRENT   READY   UP-TO-DATE
 AVAILABLE   NODE SELECTOR                AGE
daemonset.apps/calico-node               1         1         1       1            1
          kubernetes.io/os=linux     100m
...
NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/calico-kube-controllers-579b45dcf   1         1         1
  100m
replicaset.apps/calico-typha-6f9c6786d              1         1         1
  100m
```

6. 请确认其中一个 calico-node、calico-typha 和 tigera-operator pods 的日志不包含 ERROR。请将以下命令中的值替换为上一步的输出中返回的值。

```
kubectl logs tigera-operator-c4b9549c7-h4zp5 -n tigera-operator | grep ERROR
kubectl logs calico-node-v9dhf -n calico-system | grep ERROR
kubectl logs calico-typha-6f9c6786d-f2mc7 -n calico-system | grep ERROR
```

如果之前的命令没有返回任何输出，那么日志中不存在 ERROR，并且一切都应该正常运行。

Manifests

要使用清单安装 Calico

1. 将 Calico 清单应用于集群。这些清单将在 calico-system 命名空间中创建 DaemonSet。

```
kubectl apply -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/master/
config/master/calico-operator.yaml
kubectl apply -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/master/
config/master/calico-crs.yaml
```

2. 在 calico-system 命名空间中查看资源。

```
kubectl get daemonset calico-node --namespace calico-system
```

输出示例如下。

DESIRED 和 READY 列中的值应该相一致。返回的值与以下输出中的 *values* 不同。

```
NAME          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
         AGE
calico-node   1         1         1       1            1           kubernetes.io/
os=linux   26m
```

Stars 策略演示

本部分介绍了 Project Calico 文档提供的 Stars 策略演示，它对于集群上的 Calico 功能并非必需。该演示在您的 Amazon EKS 集群上创建前端、后端和客户端服务。该演示还创建管理图形用户界面，用于显示各服务之间可用的传入和传出路径。我们建议您在不运行生产工作负载的集群上完成演示。

在您创建任何网络策略之前，所有服务可以双向通信。在应用网络策略后，您可以看到客户端只能与前端服务进行通信，而后端只能接受来自前端的流量。

运行 Stars 策略演示

1. 应用前端、后端、客户端和管理用户界面服务：

```
kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/manifests/00-namespace.yaml
kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/manifests/01-management-ui.yaml
kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/manifests/02-backend.yaml
kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/manifests/03-frontend.yaml
kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/manifests/04-client.yaml
```

2. 请查看集群上的所有 pods。

```
kubectl get pods -A
```

输出示例如下。

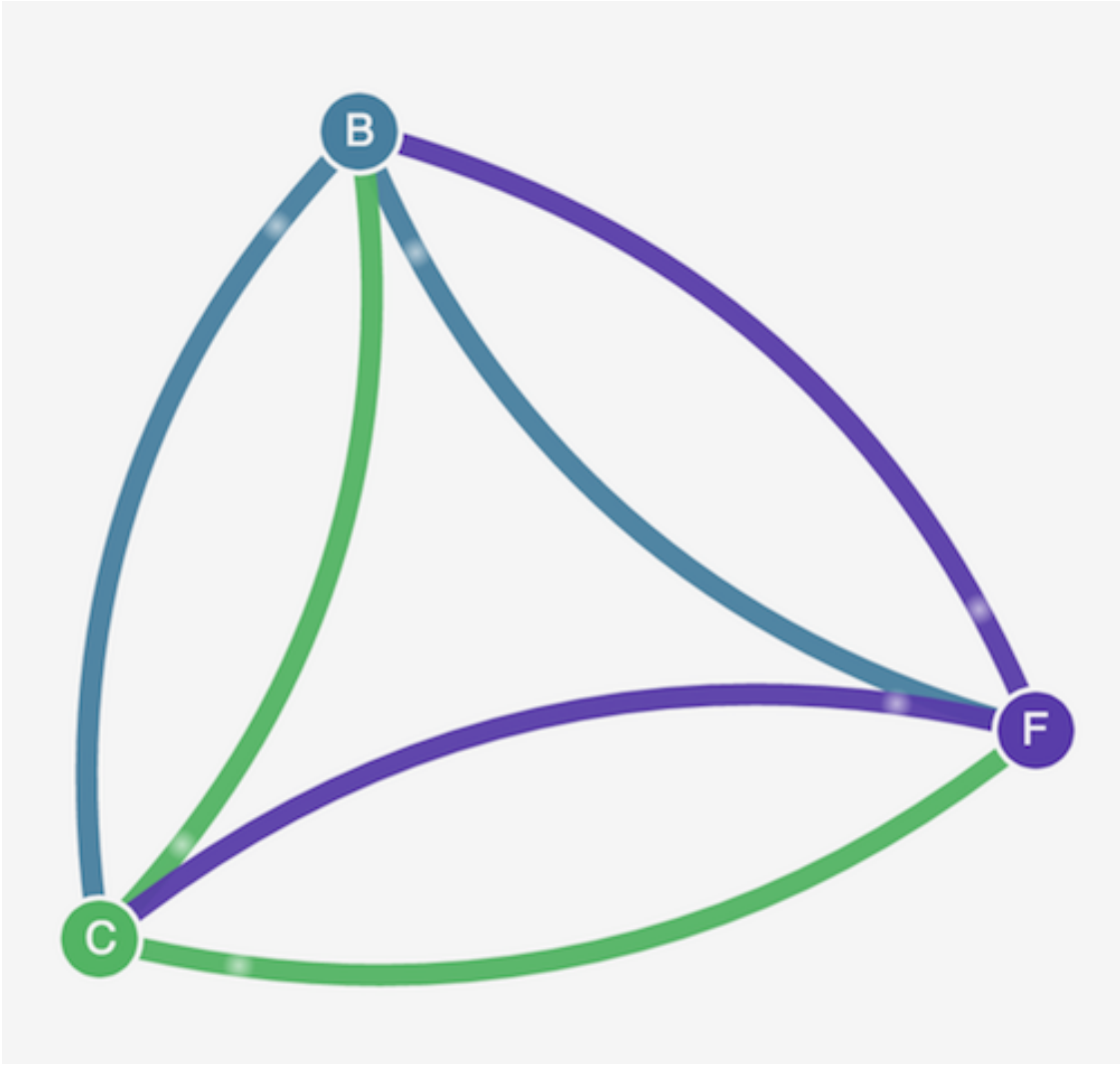
在输出中，您应该会在以下输出中显示的命名空间中看到 pods。*READY* 列中的 pod *NAMES* 和 pods 数量与以下输出中的不同。在看到具有相似名称的 pods 并且它们在 STATUS 列中都具有 Running 之前，请不要继续。

```
NAMESPACE         NAME                                READY   STATUS    RESTARTS
   AGE
...
client            client-xlffc                        1/1     Running   0
   5m19s
...
management-ui     management-ui-qrb2g                 1/1     Running   0
   5m24s
stars             backend-sz87q                       1/1     Running   0
   5m23s
stars             frontend-cscnf                      1/1     Running   0
   5m21s
...
```

3. 要连接到管理用户界面，请将您的本地端口 9001 转发至集群上运行的 management-ui 服务：

```
kubectl port-forward service/management-ui -n management-ui 9001
```

4. 在本地系统上打开浏览器，然后将其指向 http://localhost:9001/。您应该会看到管理用户界面。C 节点是客户端服务，F 节点是前端服务，B 节点是后端服务。每个节点都有到所有其他节点的完整通信访问权限（如粗体、彩色行所示）。



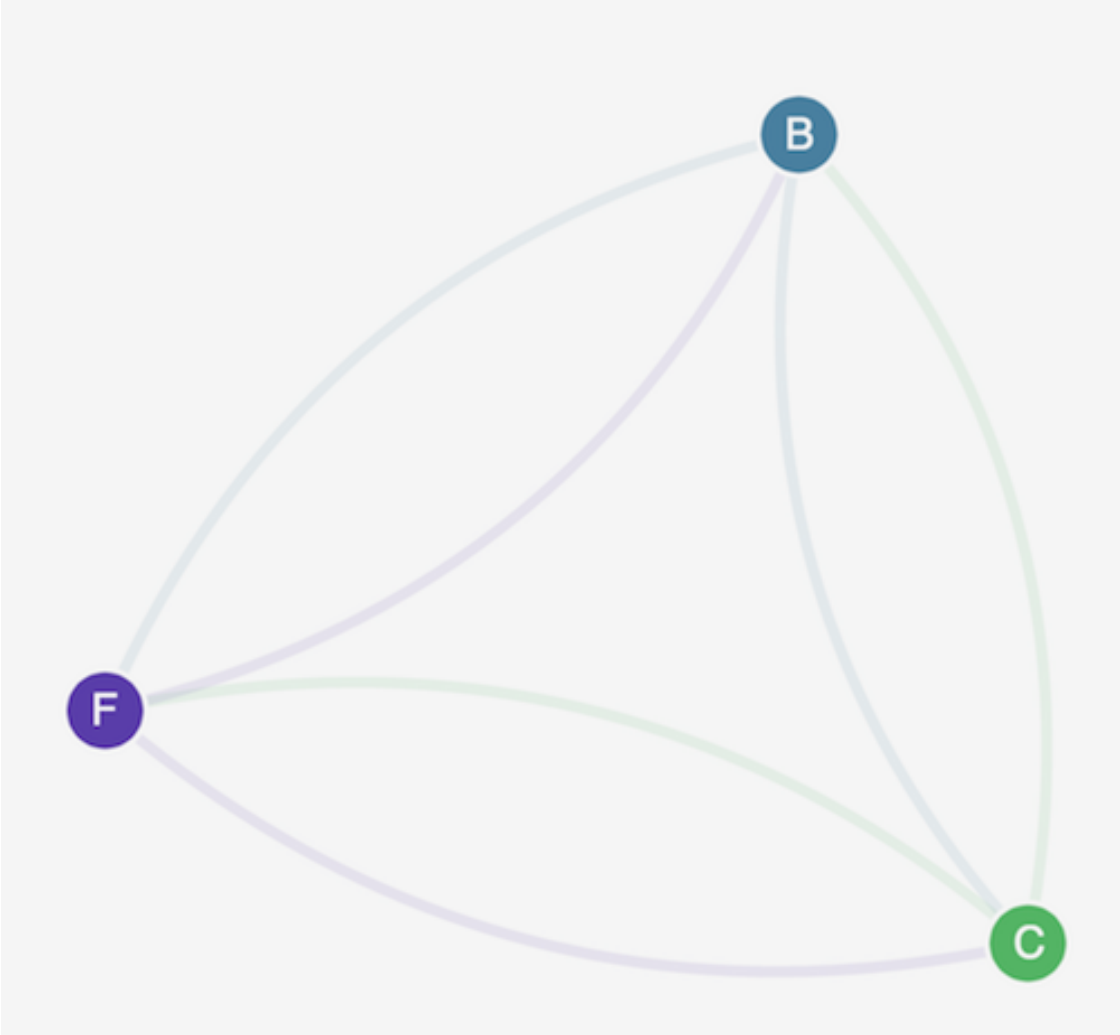
5. 应用以下网络策略以隔离各个服务：

```
kubectl apply -n stars -f https://docs.projectcalico.org/v3.5/getting-started/
kubernetes/tutorials/stars-policy/policies/default-deny.yaml
kubectl apply -n client -f https://docs.projectcalico.org/v3.5/getting-started/
kubernetes/tutorials/stars-policy/policies/default-deny.yaml
```

6. 刷新您的浏览器。您可以看到管理用户界面不再能访问任何节点，因此它们不会显示在用户界面中。
7. 应用以下网络策略以允许管理用户界面访问这些服务：

```
kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/policies/allow-ui.yaml
kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/policies/allow-ui-client.yaml
```

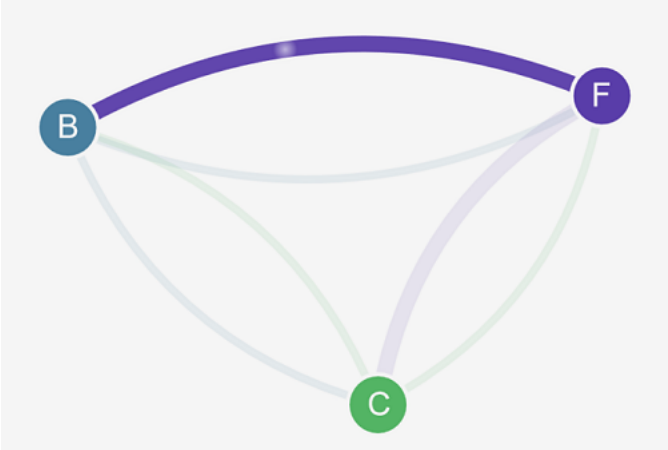
8. 刷新您的浏览器。您可以看到管理用户界面可以再次访问节点，但各节点无法相互通信。



9. 应用以下网络策略以允许流量从前端服务传到后端服务：

```
kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/policies/backend-policy.yaml
```

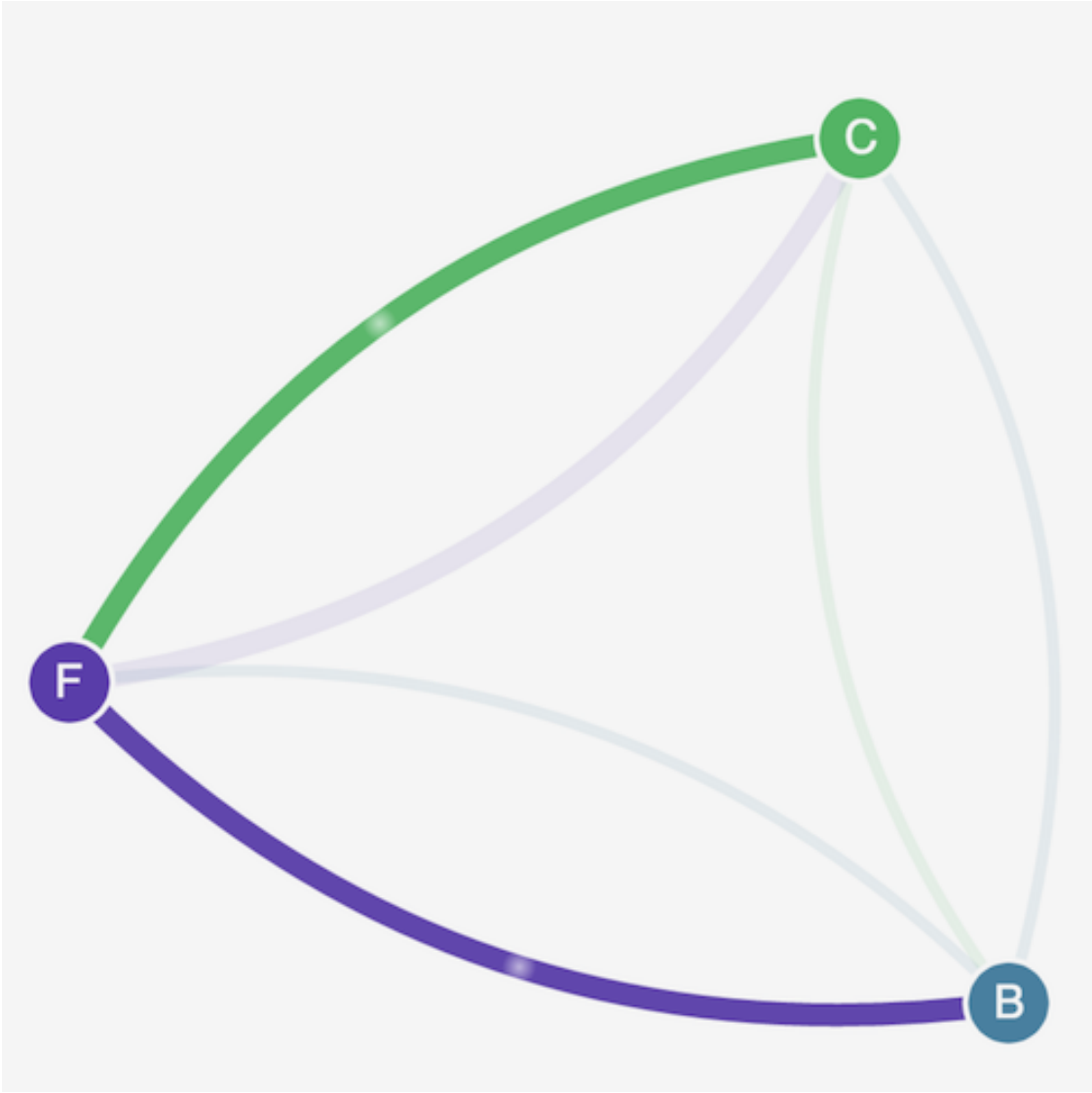
10. 刷新您的浏览器。您将看到前端可以与后端进行通信。



11. 应用以下网络策略以允许流量从客户端传到前端服务。

```
kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/policies/frontend-policy.yaml
```

12. 刷新您的浏览器。您将看到客户端可以与前端服务进行通信。前端服务仍可以与后端服务进行通信。



13. （可选）完成该演示后，您可以删除其资源。

```
kubectl delete -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/manifests/04-client.yaml
kubectl delete -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/manifests/03-frontend.yaml
kubectl delete -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/manifests/02-backend.yaml
kubectl delete -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/manifests/01-management-ui.yaml
kubectl delete -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/
tutorials/stars-policy/manifests/00-namespace.yaml
```

甚至在删除资源后，节点上仍然存在一些 `iptables` 规则，这些规则可能以意外方式干扰集群中的联网。删除 Calico 的唯一可靠方法是终止所有节点并将它们回收。要终止所有节点，请将 Auto Scaling 组所需计数设置为 0，然后创建所需数量的备份，或者仅终止节点。如果您无法回收节点，请参阅 Calico GitHub 存储库中的禁用和删除 Calico 策略以了解最后的解决方法。

删除 Calico

使用安装 Calico 的方法删除 Calico。

Helm

从集群中删除 Calico。

```
helm uninstall calico
```

Manifests

从集群中删除 Calico。

```
kubectl delete -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/master/
config/master/calico-crs.yaml
kubectl delete -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/master/
config/master/calico-operator.yaml
```

工作负载

您的工作负载部署在容器中，而容器部署在 Kubernetes 中的 pods 中。一个 pod 包含一个或多个容器。通常会有一个或多个提供相同服务的 pods 部署在 Kubernetes 服务中。在部署多个提供相同服务的 pods 以后，您可以：

- 使用 Amazon Web Services Management Console 查看在您的每个集群上运行的工作负载的相关信息 (p. 458)。
- 使用 Kubernetes Vertical Pod Autoscaler (p. 321) 纵向扩缩 pods。
- 使用 Kubernetes Horizontal Pod Autoscaler (p. 325) 横向扩缩满足需求所需的 pods 的数量。
- 创建外部（对于可通过网络访问的 pods）或内部（对于私有 pods）网络负载均衡器 (p. 327)以平衡 pods 之间的网络流量。负载均衡器在 OSI 模型的第 4 层路由流量。
- 创建 Amazon EKS 上的应用程序负载均衡 (p. 332) 以在 pods 之间平衡应用程序流量。Application Load Balancer 在 OSI 模型的第 7 层路由流量。
- 如果您是刚开始使用 Kubernetes，此主题可以帮助您 部署示例应用程序 (p. 315)。
- 您可以使用 `externalIPs` 限制可分配给服务的 IP 地址 (p. 337)。

部署示例应用程序

在本主题中，您在集群中部署一个示例应用程序。

先决条件

- 现有的 Kubernetes 集群至少有一个节点。如果没有现有 Amazon EKS 集群，可以根据 开始使用 Amazon EKS (p. 4) 指南部署一个集群。如果您正在部署 Windows 应用程序，必须为您的集群和至少一个 Amazon EC2 Windows 节点启用 Windows 支持 (p. 48)。
- 安装在计算机上的 `Kubectl`。有关更多信息，请参阅 安装或更新 `kubectl` (p. 4)。
- `Kubectl` 配置为与集群通信。有关更多信息，请参阅为 Amazon EKS 创建 `kubeconfig` (p. 365)。
- 如果计划将示例工作负载部署到 Fargate，现有 Fargate 配置文件 (p. 131)必须包含本教程创建的相同命名空间，即 `eks-sample-app`，除非更改名称。如果使用入门指南 (p. 4)创建集群，则必须创建新配置文件或将命名空间添加到现有配置文件，因为在入门指南中创建的配置文件没有指定本教程使用的命名空间。您的 VPC 还必须具有至少一个私有子网。

部署示例应用程序

尽管下列步骤中有许多变量都可以更改，但我们建议仅在指定的情况下更改变量值。更深入了解 Kubernetes pods、部署和服务后，可以尝试更改其他值。

1. 创建命名空间。命名空间允许您在 Kubernetes 中对资源进行分组。有关更多信息，请参阅 Kubernetes 文档中的命名空间。如果您计划将示例应用程序部署到 Amazon Fargate (p. 127)，请确保 Amazon Fargate 配置文件 (p. 131) 中的 `namespace` 值为 `eks-sample-app`。

   ```
   kubectl create namespace eks-sample-app
   ```

2. 创建 Kubernetes 部署。此示例部署从公共存储库中提取容器镜像，并将其三个副本（各个 pods）部署到您的集群中。有关更多信息，请参阅 Kubernetes 文档中的部署。您可以将应用程序部署到 Linux 或 Windows 节点。如果部署到 Fargate，则只能部署 Linux 应用程序。

   a. 将以下内容保存到名为 `eks-sample-deployment.yaml` 的文件中。示例应用程序中的容器不使用网络存储，但可能有需要的应用程序。有关更多信息，请参阅存储 (p. 194)。

Linux

`kubernetes.io/arch` 键下的 `amd64` 或 `arm64` `values` 意味着应用程序可以部署到任一硬件架构（如果集群中两者都有）。可能出现这种情况，因为此映像是一个多架构映像，但并非全部都是。可以通过查看从中提取的存储库的映像详细信息，确定支持映像的硬件结构。部署不支持硬件架构类型的映像或不希望将映像部署到的映像时，请从清单中删除该类型。有关更多信息，请参阅 Kubernetes 文档中的众所周知的标签、注释和污点。

`kubernetes.io/os: linux` `nodeSelector` 意味着如果集群中有 Linux 和 Windows 节点（例如），则映像将只部署到 Linux 节点。有关更多信息，请参阅 Kubernetes 文档中的众所周知的标签、注释和污点。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-linux-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-linux-app
  template:
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.21
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
      nodeSelector:
        kubernetes.io/os: linux
```

Windows

`kubernetes.io/os: windows` `nodeSelector` 意味着如果集群中有 Windows 和 Linux 节点（例如），则映像将只部署到 Windows 节点。有关更多信息，请参阅 Kubernetes 文档中的众所周知的标签、注释和污点。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-windows-deployment
  namespace: eks-sample-app
  labels:
```

```
    app: eks-sample-windows-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-windows-app
  template:
    metadata:
      labels:
        app: eks-sample-windows-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: beta.kubernetes.io/arch
                operator: In
                values:
                - amd64
      containers:
      - name: windows-server-iis
        image: mcr.microsoft.com/windows/servercore:ltsc2019
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
        command:
        - powershell.exe
        - -command
        - "Add-WindowsFeature Web-Server; Invoke-WebRequest -UseBasicParsing
 -Uri 'https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.6/
ServiceMonitor.exe' -OutFile 'C:\\ServiceMonitor.exe'; echo '<html><body><br/
><br/><marquee><H1>Hello EKS!!!<H1><marquee></body><html>' > C:\\inetpub\
\wwwroot\\default.html; C:\\ServiceMonitor.exe 'w3svc'; "
      nodeSelector:
        kubernetes.io/os: windows
```

b. 将部署清单应用于集群。

```
kubectl apply -f eks-sample-deployment.yaml
```

3. 创建服务。服务允许您通过单个 IP 地址或名称访问所有副本。有关更多信息，请参阅 Kubernetes 文档中的服务。虽然没有在示例应用程序中实施，但如果应用程序需要与其他 Amazon 服务交互，我们建议为 pods 创建 Kubernetes 服务账户，然后将其关联到 Amazon IAM 账户。指定服务账户可使您的 pods 仅拥有为其指定的与其他服务交互的最低权限。有关更多信息，请参阅服务账户的 IAM 角色 (p. 395)。

   a. 将以下内容保存到一个名为 `eks-sample-service.yaml` 的文件中。Kubernetes 为服务分配其自己的 IP 地址，该 IP 地址只能从集群内部访问。要从集群外部访问服务，请部署 Amazon Load Balancer Controller (p. 287) 以负载均衡服务的应用程序 (p. 332)或网络 (p. 327)流量。

      Linux

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-linux-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  selector:
```

```
    app: eks-sample-linux-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Windows

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-windows-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  selector:
    app: eks-sample-windows-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

b. 将服务清单应用于集群。

```
kubectl apply -f eks-sample-service.yaml
```

4. 查看 eks-sample-app 命名空间中存在的所有资源。

```
kubectl get all -n eks-sample-app
```

输出示例如下。

如果您部署了 Windows 资源，则以下输出中的所有 *linux* 实例都是 windows。其他 *example values* 可能与您的输出不同。

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/eks-sample-linux-deployment-65b7669776-m6qxz     1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-mmxvd     1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-qzn22     1/1     Running   0          27m

NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP   PORT(S)
 AGE
service/eks-sample-linux-service    ClusterIP     10.100.74.8     <none>        80/TCP
 32m

NAME                                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-linux-deployment    3/3     3            3           27m

NAME                                                      DESIRED   CURRENT   READY
 AGE
replicaset.apps/eks-sample-linux-deployment-776d8f8fd8    3         3         3
 27m
```

在输出中，可以看到之前步骤部署的示例清单中指定的服务和部署。您还可以看到三个 pods。这是因为在示例清单中指定了 3 replicas。有关 pods 的更多信息，请参阅 Kubernetes 文档中的容器组（pod）。即使没有在示例清单中指定，Kubernetes 也会自动创建 replicaset 资源。有关 ReplicaSets 的更多信息，请参阅 Kubernetes 文档中的 ReplicaSet。

Note

Kubernetes 保持清单中指定的副本数目。如果这是生产部署，并且您希望 Kubernetes 为 pods 横向扩展副本数量或纵向扩展计算资源，则使用 Horizontal Pod Autoscaler (p. 325) 和 Vertical Pod Autoscaler (p. 321) 进行此操作。

5. 查看已部署服务的详细信息。如果您部署了 Windows 服务，请将 *linux* 替换为 **windows**。

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

输出示例如下。

如果您部署了 Windows 资源，则以下输出中的所有 *linux* 实例都是 windows。其他 *example values* 可能与您的输出不同。

```
Name:              eks-sample-linux-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-linux-app
Annotations:       <none>
Selector:          app=eks-sample-linux-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

在之前的输出中，IP: 值是一个唯一的 IP 地址，可以从集群内的任何节点或 pod 访问，但无法从集群外部访问。Endpoints 的值是从 VPC 内分配给 pods 的 IP 地址，其中，pods 属于服务的一部分。

6. 在上一步查看命名空间 (p. 318)时，查看输出中列出的一个 pods 的详细信息。如果部署了 Windows 应用程序，请将 *linux* 替换为 **windows**，将 *776d8f8fd8-78w66* 替换为其中一个 pods 返回的值。

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

缩减的输出

如果您部署了 Windows 资源，则以下输出中的所有 *linux* 实例都是 windows。其他 *example values* 可能与您的输出不同。

```
Name:         eks-sample-linux-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.cn-north-1.compute.internal/192.168.45.132
...
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-linux-deployment-65b7669776
...
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
...
```

```
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-linux-deployment-65b7669776-m6qxz to
 ip-192-168-45-132.cn-north-1.compute.internal
...
```

在之前的输出中，`IP:` 值是一个唯一的 IP 地址，该 IP 地址从分配给节点所在子网的 CIDR 块中分配给 pod。如果您希望从其他 CIDR 块中为 pods 分配 IP 地址，则可以更改默认行为。有关更多信息，请参阅教程：自定义联网 (p. 258)。您还可以看到 Kubernetes 调度器在 IP 地址为 *192.168.45.132* 的 `Node` 上计划 pod。

> Tip
>
> 无需使用命令行，即可在 Amazon Web Services Management Console 中查看 pods、服务、部署和其他 Kubernetes 资源的许多详细信息。有关更多信息，请参阅查看 Kubernetes 资源 (p. 458)。

7. 在上一步中描述的 pod 上运行 shell，将 *65b7669776-m6qxz* 替换为其中一个 pods 的 ID。

   Linux

   ```
   kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app
    -- /bin/bash
   ```

   Windows

   ```
   kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-app
    -- powershell.exe
   ```

8. 在 pod shell 中，查看上一步中随部署一起安装的 Web 服务器的输出。您只需指定服务名称。默认情况下，CoreDNS 将其解析为服务的 IP 地址，该地址与 Amazon EKS 集群一起部署。

   Linux

   ```
   curl eks-sample-linux-service
   ```

   输出示例如下。

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   ...
   ```

   Windows

   ```
   Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
   ```

   输出示例如下。

   ```
   StatusCode        : 200
   StatusDescription : OK
   Content           : < h t m l > < b o d y > < b r / > < b r / > < m a r q u e e > <
    H 1 > H e l l o
   ```

```
                    E K S ! ! ! < H 1 > < m a r q u e e > < / b o d y > < h t m l
 >
```

9. 在 pod shell 中，查看 pod 的 DNS 服务器。

   Linux

   ```
   cat /etc/resolv.conf
   ```

   输出示例如下。

   ```
   nameserver 10.100.0.10
   search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local cn-
   north-1.compute.internal
   options ndots:5
   ```

   在之前的输出中，`10.100.0.10` 自动分配为部署到集群的所有 pods 的 `nameserver`。

   Windows

   ```
   Get-NetIPConfiguration
   ```

   缩减的输出

   ```
   InterfaceAlias       : vEthernet
   ...
   IPv4Address          : 192.168.63.14
   ...
   DNSServer            : 10.100.0.10
   ```

   在之前的输出中，`10.100.0.10` 自动分配为部署到集群的所有 pods 的 DNS 服务器。

10. 键入 `exit`，与 pod 断开连接。
11. 使用完示例应用程序后，您可以使用以下命令删除示例命名空间、服务和部署。

    ```
    kubectl delete namespace eks-sample-app
    ```

Vertical Pod Autoscaler

Kubernetes Vertical Pod Autoscaler 为 pods 自动调整 CPU 和内存预留，使应用程序调整至合适大小。此调整可以提高集群资源利用率并释放 CPU 和内存供其他 pods 使用。本主题可帮助您将 Vertical Pod Autoscaler 部署到集群并验证它在正常工作。

先决条件

- 您拥有现有 Amazon EKS 集群。如果没有，请参阅 开始使用 Amazon EKS (p. 4)。
- 您已安装 Kubernetes Metrics Server。有关更多信息，请参阅安装 Kubernetes Metrics Server (p. 380)。
- 您使用的是配置为与 Amazon EKS 集群通信 (p. 17)的 `kubectl` 客户端。
- 您的设备已安装 OpenSSL `1.1.1` 或更高版本。

部署 Vertical Pod Autoscaler

在此部分中，您将部署 Vertical Pod Autoscaler 到集群。

部署 Vertical Pod Autoscaler

1. 打开终端窗口，导航到您要下载 Vertical Pod Autoscaler 源代码的目录。
2. 克隆 kubernetes/autoscaler GitHub 存储库。

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. 切换到 `vertical-pod-autoscaler` 目录。

```
cd autoscaler/vertical-pod-autoscaler/
```

4. （可选）如果您已经部署另一个版本的 Vertical Pod Autoscaler，请使用以下命令将其删除。

```
./hack/vpa-down.sh
```

5. 如果您的节点无权访问 `k8s.gcr.io` 容器注册表，则您需要提取下面的镜像并将它们推送到自己的私有存储库。有关如何提取镜像并将它们推送到您自己的私有存储库的更多信息，请参阅 将容器镜像从一个存储库复制到另一个存储库 (p. 338)。

```
k8s.gcr.io/autoscaling/vpa-admission-controller:0.10.0
k8s.gcr.io/autoscaling/vpa-recommender:0.10.0
k8s.gcr.io/autoscaling/vpa-updater:0.10.0
```

   如果要将镜像推送到私有 Amazon ECR 存储库，请将清单中的 `k8s.gcr.io` 替换为您的注册表。请将 *111122223333* 替换为您的账户 ID。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。以下命令假定您将存储库命名为与清单中的存储库名称相同。如果已将存储库命名为其他名称，则也需要进行更改。

```
sed -i.bak -e 's/k8s.gcr.io/111122223333.dkr.ecr.region-codeamazonaws.com.cn/' ./
deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/k8s.gcr.io/111122223333.dkr.ecr..dkr.ecr.region-
codeamazonaws.com.cn/' ./deploy/recommender-deployment.yaml
sed -i.bak -e 's/k8s.gcr.io/111122223333.dkr.ecr..dkr.ecr.region-
codeamazonaws.com.cn/' ./deploy/updater-deployment.yaml
```

6. 使用以下命令将 Vertical Pod Autoscaler 部署到您的集群。

```
./hack/vpa-up.sh
```

7. 验证已成功创建 Vertical Pod Autoscaler pods。

```
kubectl get pods -n kube-system
```

   输出示例如下。

```
NAME                                        READY   STATUS    RESTARTS   AGE
...
metrics-server-8459fc497-kfj8w              1/1     Running   0          83m
vpa-admission-controller-68c748777d-ppspd   1/1     Running   0          7s
vpa-recommender-6fc8c67d85-gljpl            1/1     Running   0          8s
vpa-updater-786b96955c-bgp9d                1/1     Running   0          8s
```

测试 Vertical Pod Autoscaler 安装

在此部分中，您部署示例应用程序以验证 Vertical Pod Autoscaler 在正常运行。

测试 Vertical Pod Autoscaler 安装

1. 使用以下命令部署 `hamster.yaml` Vertical Pod Autoscaler 示例。

```
kubectl apply -f examples/hamster.yaml
```

2. 从 `hamster` 示例应用程序获取 pods。

```
kubectl get pods -l app=hamster
```

输出示例如下。

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. 描述其中一个 pods 以查看其 `cpu` 和 `memory` 预留。请将 *c7d89d6db-rglf5* 替换为上一步输出中返回的 ID 之一。

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

输出示例如下。

```
...
Containers:
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:         k8s.gcr.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://k8s.gcr.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:35:16 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     50Mi
...
```

您可以看到原始 pod 预留了 100 millicpu 和 50 MiB 内存。对于本示例应用程序，100 millicpu 小于 pod 运行所需的数量，因此 CPU 受限。它预留的内存也远小于所需的数量。Vertical Pod Autoscaler `vpa-recommender` 部署分析 `hamster` pods，以查看 CPU 和内存需求是否合适。如果需要调整，`vpa-updater` 使用更新后的值重新启动 pods。

4. 等待 `vpa-updater` 启动新 `hamster` pod。这大概需要一两分钟。您可以使用以下命令监控 pods。

Note

如果您不确定已经启动了新 pod，请将 pod 名称与您之前的列表比较。新 pod 启动时，您会看到新 pod 名称。

```
kubectl get --watch pods -l app=hamster
```

5. 当新 hamster pod 启动时，描述它并查看更新后的 CPU 和内存预留。

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

输出示例如下。

```
...
Containers:
  hamster:
    Container ID:
 docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:         k8s.gcr.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://k8s.gcr.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:37:08 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        587m
      memory:     262144k
...
```

在之前的输出中，您可以看到 cpu 预留提升到了 587 个 millicpu，这是原始值的五倍多。memory 提高到了 262144 KB，即大约 250 MB，也就是原始值的五倍。此 pod 资源不足，Vertical Pod Autoscaler 使用更为合适的值纠正了估计值。

6. 描述 hamster-vpa 资源以查看新的建议。

```
kubectl describe vpa/hamster-vpa
```

输出示例如下。

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/
v1beta2","kind":"VerticalPodAutoscaler","metadata":{"annotations":{},"name":"hamster-
vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:          23
  Resource Version:    14411
```

```
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/
verticalpodautoscalers/hamster-vpa
  UID:                 d0d85fb9-e153-11e9-ae53-0205785d75b0
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
  Conditions:
    Last Transition Time:  2019-09-27T18:23:28Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
        Cpu:     550m
        Memory:  262144k
      Target:
        Cpu:     587m
        Memory:  262144k
      Uncapped Target:
        Cpu:     587m
        Memory:  262144k
      Upper Bound:
        Cpu:     21147m
        Memory:  387863636
Events:          <none>
```

7. 在完成对示例应用程序的试验后，使用以下命令可将其删除。

```
kubectl delete -f examples/hamster.yaml
```

Horizontal Pod Autoscaler

Kubernetes Horizontal Pod Autoscaler 根据资源的 CPU 利用率，自动缩放部署、复制控制器或副本集中的 pods 数量。这有助于您的应用程序进行扩展以满足增长的需求，或在不需要资源时进行缩减，从而释放出节点用于其他应用程序。当您设置目标 CPU 利用率百分比时，Horizontal Pod Autoscaler 扩展或缩减应用程序来尝试满足该目标。

Horizontal Pod Autoscaler 是 Kubernetes 中的标准 API 资源，只需在 Amazon EKS 集群上安装一个指标源（如 Kubernetes Metrics Server）即可正常运行。您不需要在集群上部署或安装 Horizontal Pod Autoscaler 以开始扩展您的应用程序。有关更多信息，请参阅 Kubernetes文档中的 Horizontal Pod Autoscaler。

使用本主题为您的 Amazon EKS 集群准备 Horizontal Pod Autoscaler 并验证它可用于示例应用程序。

> **Note**
>
> 本主题基于 Kubernetes 文档中的 Horizontal pod autoscaler 演练。

先决条件

- 您拥有现有 Amazon EKS 集群。如果没有，请参阅 开始使用 Amazon EKS (p. 4)。
- 您已安装 Kubernetes Metrics Server。有关更多信息，请参阅安装 Kubernetes Metrics Server (p. 380)。
- 您使用的是配置为与 Amazon EKS 集群通信 (p. 17)的 `kubectl` 客户端。

运行 Horizontal Pod Autoscaler 测试应用程序

在此部分中，您部署示例应用程序以验证 Horizontal Pod Autoscaler 在正常运行。

Note

本示例基于 Kubernetes 文档中的 Horizontal pod autoscaler 演练。

测试 Horizontal Pod Autoscaler 安装

1. 使用以下命令部署一个简单的 Apache Web 服务器应用程序。

```
kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
```

   向此 Apache Web 服务器 pod 提供 500 millicpu 的 CPU 限制，并在端口 80 上提供服务。
2. 为 `php-apache` 部署创建 Horizontal Pod Autoscaler 资源。

```
kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
```

   此命令创建为部署定位 50% CPU 利用率的自动缩放器，最少一个 pod，最多十个 pods。当平均 CPU 负载低于 50% 时，Autoscaler 尝试减少部署中的 pods 数量，最低一个。当负载大于 50% 时，自动缩放器尝试增加部署中的 pods 数量，最高十个。有关更多信息，请参阅 Kubernetes 文档中的 HorizontalPodAutoscaler 的工作原理。
3. 使用以下命令描述 Autoscaler 以查看其详细信息。

```
kubectl get hpa
```

   输出示例如下。

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   0%/50%    1         10        1          51s
```

   如您所见，当前 CPU 负载是 `0%`，因为服务器上尚没有负载。pod 计数已处于其最低边界（1 个），因此无法横向缩减。
4. 通过运行容器为 Web 服务器创建负载。

```
kubectl run -i \
    --tty load-generator \
    --rm --image=busybox \
    --restart=Never \
    -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
```

5. 要监视部署的向外扩展情况，请在与执行上一步骤的终端不同的终端上定期运行以下命令。

```
kubectl get hpa php-apache
```

   输出示例如下。

```
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   250%/50%   1         10        5          4m44s
```

   可能需要一分钟以上的时间才能增加副本计数。只要实际 CPU 百分比高于目标百分比，副本计数就会增加（最大值为 10）。在此情况下，此百分比为 `250%`，因此 `REPLICAS` 数会继续增加。

Note

可能需要在几分钟后，您才能看到副本计数达到其最大值。例如，如果只需 6 个副本即可让 CPU 负载小于或等于 50%，则负载将不会超过 6 个副本。

6. 停止负载。在正在生成负载的终端窗口中，按住 Ctrl+C 键停止负载。在观察扩缩的终端中再次运行以下命令，您会看到副本数缩减回 1。

```
kubectl get hpa
```

输出示例如下。

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    0%/50%    1         10        1          25m
```

Note

默认缩减时间范围为 5 分钟，因此，在您看到副本计数再次达到 1 之前也需要一段时间，甚至在当前 CPU 百分比为 0% 时也是如此。时间范围可以修改。有关更多信息，请参阅 Kubernetes 文档中的 Horizontal Pod Autoscaler。

7. 完成示例应用程序的试验之后，删除 php-apache 资源。

```
kubectl delete deployment.apps/php-apache service/php-apache
 horizontalpodautoscaler.autoscaling/php-apache
```

Amazon EKS 上的网络负载均衡

网络流量在 OSI 模型的 L4 上实现负载均衡。若要对 L7 上的应用程序流量进行负载均衡，请部署 Kubernetes ingress，它会预置 Amazon 应用程序负载均衡器。有关更多信息，请参阅Amazon EKS 上的应用程序负载均衡 (p. 332)。要了解更多有关两种负载均衡类型之间差异的信息，请参阅 Amazon 网站上的 Elastic Load Balancing 功能。

创建类型为 LoadBalancer 的 Kubernetes Service 时，Amazon 云提供程序负载均衡器控制器默认创建 Amazon 经典负载均衡器，但也可以创建 Amazon 网络负载均衡器。此控制器将来仅接收关键错误修复。有关如何使用 Amazon 云提供程序负载均衡器的更多信息，请参阅 Kubernetes 文档中的 Amazon 云提供程序负载均衡器控制器。本主题中未涵盖其使用。

对于部署到 1.19 版或更高版本集群的新服务，建议您使用 2.4.3 版本或更高版本的 Amazon Load Balancer Controller (p. 287)，而不是Amazon云提供程序负载均衡器控制器。如果您的集群早于 1.19 版本，则我们建议使用 2.3.1 版本的控制器。Amazon Load Balancer Controller 创建 Amazon 网络负载均衡器，但不创建 Amazon 经典负载均衡器。本主题的其余部分介绍如何使用 Amazon 负载均衡器控制器。

Amazon 网络负载均衡器可以对部署到 Amazon EC2 IP 和实例目标或 Amazon Fargate IP 目标的 pods 的网络流量进行负载均衡。有关更多信息，请参阅 GitHub 上的 Amazon负载均衡器控制器。

先决条件

必须满足以下要求，才能使用 Amazon Load Balancer Controller 对网络流量进行负载均衡。

- 拥有现有集群。如果没有现有集群，请参阅 开始使用 Amazon EKS (p. 4)。如果您需要更新现有集群的版本，请参阅 更新 Amazon EKS 集群 Kubernetes 版本 (p. 28)。
- 在集群上部署 Amazon Load Balancer Controller。有关更多信息，请参阅安装 Amazon Load Balancer Controller 附加组件 (p. 287)。对于 1.19 或更高版本的集群，我们建议使用 2.4.3 或更高版本。如果您的集群早于 1.19 版本，则我们建议使用 2.3.1 版本。

- 至少有一个子网。如果在一个可用区中发现多个标记子网，则该控制器会按子网 ID 的字典顺序选择第一个子网。子网必须具有至少 8 个可用 IP 地址。
- 如果使用 `2.1.1` 或更早版本的 Amazon Load Balancer Controller，则必须按如下方式标记子网。如果使用版本 `2.1.2` 或更高版本，则此标签是可选的。如果您在同一 VPC 中运行多个集群，或者 VPC 中有多个 Amazon 服务共享子网，并且希望对每个集群中预置负载均衡器的位置时效性更多控制，则可能需要标记子网。如果您明确指定子网 ID 作为服务对象上的注释，则 Kubernetes 和 Amazon Load Balancer Controller 会直接使用这些子网来创建负载均衡器。如果您选择使用此方法来预置负载均衡器，则不需要子网标记，您可以跳过以下私有和公有子网标记要求。将 *`my-cluster`* 替换为您的集群名称。
  - 密钥 – `kubernetes.io/cluster/`*`my-cluster`*
  - 值 – `shared` 或 `owned`
- 除非您明确指定子网 ID 作为服务或入口对象上的注释，否则您的公有子网和私有子网必须满足以下要求。如果通过明确指定子网 ID 作为服务或入口对象上的注释来预置负载均衡器，则 Kubernetes 和 Amazon Load Balancer Controller 会直接使用这些子网来创建负载均衡器，并且不需要以下标签。
  - 私有子网：必须采用以下格式标记。这样，Kubernetes 和 Amazon 负载均衡器控制器就会知道子网可用于内部负载均衡器。如果您在 2020 年 3 月 26 日之后使用 `eksctl` 或 Amazon EKS Amazon Amazon CloudFormation 模板创建 VPC，则在创建子网时会对子网进行适当标记。有关 Amazon EKS Amazon Amazon CloudFormation VPC 模板的更多信息，请参阅 为 Amazon EKS 集群创建 VPC (p. 228)。
    - 密钥 – `kubernetes.io/role/internal-elb`
    - 值 – `1`
  - 公有子网：必须采用以下格式标记。这样，Kubernetes 就会知道仅将这些子网用于外部负载均衡器，而不是在每个可用区中选择公有子网（按子网 ID 的词典式顺序选择）。如果您在 2020 年 3 月 26 日之后使用 `eksctl` 或 Amazon EKS Amazon CloudFormation 模板创建 VPC，则在创建子网时会对子网进行适当标记。有关 Amazon EKS Amazon CloudFormation VPC 模板的更多信息，请参阅 为 Amazon EKS 集群创建 VPC (p. 228)。
    - 密钥 – `kubernetes.io/role/elb`
    - 值 – `1`

  如果未显式添加子网角色标记，则 Kubernetes 服务控制器将检查您的集群 VPC 子网的路由表，以确定子网是私有子网还是公有子网。我们建议您不要依赖此行为，而是明确地添加私有或公有角色标记。Amazon Load Balancer Controller 不会检查路由表，并且需要存在私有和公有标签才能成功地自动发现。

注意事项

- 负载均衡器的配置由添加到服务清单中的注释控制。使用 Amazon Load Balancer Controller 和使用 Amazon 云提供程序负载均衡器控制器的服务注释不同。部署服务前，请务必检查 Amazon Load Balancer Controller 的注释。
- 使用 Amazon VPC CNI plugin for Kubernetes (p. 233) 时，Amazon Load Balancer Controller 可以对 Amazon EC2 IP 或实例目标和 Fargate IP 目标进行负载均衡。使用备选的兼容 CNI 插件 (p. 287)时，控制器只能对实例目标进行负载均衡。有关 Network Load Balancer 目标类型的更多信息，请参阅 Network Load Balancer 用户指南中的目标类型
- 如果要在创建负载均衡器时或之后将标记添加到负载均衡器，请在服务规范中添加以下注释。有关更多信息，请参阅 Amazon Load Balancer Controller 文档中的 Amazon 资源标签。

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
```

- 您可以通过添加以下注释，从而将弹性 IP 地址分配到 Network Load Balancer。请将 *`example values`* 替换为弹性 IP 地址的 `Allocation IDs`。`Allocation IDs` 的数量必须与用于负载均衡器的子网数量相匹配。有关更多信息，请参阅 Amazon Load Balancer Controller 文档。

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations:
 eipalloc-xxxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyyy
```

- Amazon EKS 将一条入站规则添加到节点的安全组用于客户端流量，并添加一条规则用于 VPC 中的每个负载均衡器子网，对您创建的每个 Network Load Balancer 进行运行状况检查。如果 Amazon EKS 尝试创建的规则数超出安全组允许的最大规则数配额，则部署 `LoadBalancer` 类型的服务可能会失败。有关更多信息，请参阅 Amazon VPC 用户指南的 Amazon VPC 配额中的安全组。请考虑以下选项，以最大限度地减少超过安全组最大规则数的可能性：
  - 请求增加每个安全组配额的规则数。有关更多信息，请参阅 Service Quotas 用户指南中的请求增加配额。
  - 使用 IP 目标，而不是实例目标。对于 IP 目标，可以为相同目标端口共享规则。可以使用注释手动指定负载均衡器子网。有关更多信息，请参阅 GitHub 上的注释。
  - 使用入口（而不是 `LoadBalancer` 类型的服务）将流量发送到您的服务。Amazon Application Load Balancer 需要的规则数比 Network Load Balancer 少。您可以跨多个入口共享 ALB。有关更多信息，请参阅Amazon EKS 上的应用程序负载均衡 (p. 332)。您无法跨多个服务共享 Network Load Balancer。
  - 将您的集群部署到多个账户。
- 如果您的 pods 在 Amazon EKS 集群中的 Windows 上运行，则采用负载均衡器的单个服务最多可支持 1024 个后端 pods。每个 pod 都有自己的唯一 IP 地址。
- 我们建议您仅使用 Amazon Load Balancer Controller 创建新的网络负载均衡器。尝试替换使用 Amazon 云提供程序负载均衡器控制器创建的现有 Network Load Balancer，将产生多个可能导致应用程序停机的 Network Load Balancer。

创建网络负载均衡器

您可以使用 IP 或实例目标创建网络负载均衡器。

IP targets

您可以将 IP 目标与部署到 Amazon EC2 节点或 Fargate 的 pods 结合使用。您的 Kubernetes 服务必须创建为类型 `LoadBalancer`。有关更多信息，请参阅 Kubernetes 文档中的类型 LoadBalancer。

要创建使用 IP 目标的负载均衡器，请将以下注释添加到服务清单中并部署您的服务。`aws-load-balancer-type` 的 `external` 值是导致 Amazon Load Balancer Controller 而不是 Amazon 云提供程序负载均衡器控制器创建网络负载均衡器的原因。您可以查看带有注释的示例服务清单 (p. 331)。

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

Note

如果要对 `IPv6` pods 进行负载均衡，请添加以下注释。您只能通过 `IPv6` 对 IP 目标进行负载均衡，无法对实例目标进行负载均衡。如果没有此注释，则通过 `IPv4` 进行负载均衡。

```
service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
```

默认情况下，Network Load Balancer 使用 `internal` `aws-load-balancer-scheme` 创建。您可以在集群 VPC 的任意子网中启动网络负载均衡器，包括在创建集群时未指定的子网。

Kubernetes 会检查子网的路由表来确定它们是公有还是私有。公有子网有使用互联网网关直接连接到 Internet 的路由，但私有子网没有。

如果要在公有子网中创建 Network Load Balancer 以将负载均衡到 Amazon EC2 节点（Fargate 只能是私有的），请指定具有以下注释的 `internet-facing`：

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

Note

仍支持 `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` 注释的向后兼容性。但是，我们建议您将以前的注释用于新负载均衡器，而不是 `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"`。

Important

创建服务后，请勿编辑注释。如果需要对其进行修改，请删除相应服务对象，然后使用此注释的所需值重新创建它。

Instance targets

Amazon 云提供程序负载均衡器控制器仅使用实例目标创建 Network Load Balancer。`2.2.0` 版和更高版本的 Amazon 负载均衡器控制器也使用实例目标创建网络负载均衡器。我们建议使用该控制器而不是 Amazon 云提供程序负载均衡器控制器来创建新的 Network Load Balancer。您可以将网络负载均衡器实例目标与部署到 Amazon EC2 节点（而不是部署到 Fargate）的 pods 结合使用。要在部署到 Fargate 的不同 pods 之间对网络流量进行负载均衡，必须使用 IP 目标。

要将 Network Load Balancer 部署到私有子网，您的服务规范须具有以下注释。您可以查看带有注释的示例服务清单 (p. 331)。`aws-load-balancer-type` 的 `external` 值是导致 Amazon 负载均衡器控制器而不是 Amazon 云提供程序负载均衡器控制器创建 Network Load Balancer 的原因。

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
```

默认情况下，Network Load Balancer 使用 `internal` `aws-load-balancer-scheme` 创建。对于内部网络负载均衡器，Amazon EKS 集群必须配置为使用 VPC 中的至少一个私有子网。Kubernetes 会检查子网的路由表来确定它们是公有还是私有。公有子网有使用互联网网关直接连接到 Internet 的路由，但私有子网没有。

如果要在公有子网中创建 Network Load Balancer 以将负载均衡到 Amazon EC2 节点，请指定具有以下注释的 `internet-facing`：

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

Important

创建服务后，请勿编辑注释。如果需要对其进行修改，请删除相应服务对象，然后使用此注释的所需值重新创建它。

（可选）部署示例应用程序

先决条件

- 集群 VPC 中至少有一个公有或私有子网。
- 在集群上部署 Amazon Load Balancer Controller。有关更多信息，请参阅安装 Amazon Load Balancer Controller 附加组件 (p. 287)。建议升级到版本 `2.4.3` 或更高版本。

部署示例应用程序

1. 如果部署到 Fargate，请确保您的 VPC 中有一个可用的私有子网，然后创建 Fargate 配置文件。如果您不部署到 Fargate，请跳过此步骤。您可以通过运行以下命令来创建配置文件，也可以在 Amazon Web Services Management Console (p. 133) 中，使用该命令中 `name` 和 `namespace` 的相同值创建配置文件。将 *`example values`* 替换为您自己的值。

```
eksctl create fargateprofile \
```

```
--cluster my-cluster \
--region region-code \
--name nlb-sample-app \
--namespace nlb-sample-app
```

2. 部署示例应用程序。

   a. 为应用程序创建命名空间。

   ```
   kubectl create namespace nlb-sample-app
   ```

   b. 将以下内容保存到计算机上名为 *sample-deployment*.yaml 文件的文件中。

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: nlb-sample-app
     namespace: nlb-sample-app
   spec:
     replicas: 3
     selector:
       matchLabels:
         app: nginx
     template:
       metadata:
         labels:
           app: nginx
       spec:
         containers:
           - name: nginx
             image: public.ecr.aws/nginx/nginx:1.21
             ports:
               - name: tcp
                 containerPort: 80
   ```

   c. 将清单应用于集群。

   ```
   kubectl apply -f sample-deployment.yaml
   ```

3. 创建具有面向互联网的网络负载均衡器的服务，以便将负载均衡到 IP 目标。

   a. 将以下内容保存到计算机上名为 *sample-service*.yaml 文件的文件中。如果部署到 Fargate 节点，请删除 `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing` 行。

   ```
   apiVersion: v1
   kind: Service
   metadata:
     name: nlb-sample-service
     namespace: nlb-sample-app
     annotations:
       service.beta.kubernetes.io/aws-load-balancer-type: external
       service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
       service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
   spec:
     ports:
       - port: 80
         targetPort: 80
         protocol: TCP
     type: LoadBalancer
     selector:
       app: nginx
   ```

b. 将清单应用于集群。

```
kubectl apply -f sample-service.yaml
```

4. 确认是否已部署相应服务。

```
kubectl get svc nlb-sample-service -n nlb-sample-app
```

输出示例如下。

```
NAME             TYPE           CLUSTER-IP         EXTERNAL-IP
                                                 PORT(S)        AGE
sample-service   LoadBalancer   10.100.240.137
 k8s-nlbsampl-nlbsampl-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com.cn
  80:32400/TCP    16h
```

Note

10.100.240.137 和 *xxxxxxxxxx-xxxxxxxxxxxxxxxx* 的值与示例输出不同（这些值将是您的负载均衡器独有的），并且 *cn-north-1* 可能因集群所在的 Amazon Web Services 区域而有所不同。

5. 打开 Amazon EC2 Amazon Web Services Management Console。在左侧导航窗格中，选择 Target Groups（目标组）（位于 Load Balancing（负载均衡）下）。在 Name（名称）列中，选择目标组的名称，其中 Load balancer（负载均衡器）列的值与上一步输出的 `EXTERNAL-IP` 列中的一部分名称相匹配。例如，如果您的输出与之前的输出相同，则应选择名为 `k8s-default-samplese-`*xxxxxxxxxx* 的目标组。Target type（目标类型）为 `IP`，因为在示例服务清单中指定。
6. 选择 Target group（目标组），然后选择 Targets（目标）选项卡。在 Registered targets（已注册目标）中，您应该可看到上一步中部署的三个副本的三个 IP 地址。在所有目标的状态均为 healthy（正常）之前，请耐心等待，然后再继续。可能需要几分钟时间所有目标的状态才能达到 `healthy`。在更改为 `healthy` 状态之前，目标可能是 `unhealthy` 状态。
7. 将流量发送到服务，将 *xxxxxxxxxx-xxxxxxxxxxxxxxxx* 和 *cn-north-1* 替换为在上一步 (p. 332)输出中为 `EXTERNAL-IP` 返回的值。如果您已部署到私有子网，则需要从 VPC 中的设备（例如堡垒主机）查看页面。有关更多信息，请参阅 Amazon 上的 Linux 堡垒机主机。

```
curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com.cn
```

输出示例如下。

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
```

8. 完成示例部署、服务和命名空间后，将其移除。

```
kubectl delete namespace nlb-sample-app
```

Amazon EKS 上的应用程序负载均衡

当您创建 Kubernetes `ingress` 时，会预置一个 Amazon 应用程序负载均衡器 (ALB, Application Load Balancer)，以实现应用程序流量负载均衡。要了解详情，请参阅应用程序负载均衡器用户指南中的什么是应

用程序负载均衡器？以及 Kubernetes 文档中的入口。ALB 可以与部署到节点或 Amazon Fargate 的 pods 一起使用。您可以将 ALB 部署到公有子网或私有子网。

应用程序流量在 OSI 模型的 `L7` 上实现均衡。要在 `L4` 上对网络流量进行负载均衡，您可以部署 `LoadBalancer` 类型的 Kubernetes `service`。这种类型将预置 Amazon Network Load Balancer。有关更多信息，请参阅Amazon EKS 上的网络负载均衡 (p. 327)。要了解更多有关两种负载均衡类型之间差异的信息，请参阅 Amazon 网站上的 Elastic Load Balancing 功能。

先决条件

在对应用程序的应用程序流量进行负载均衡之前，您必须符合以下要求。

- 拥有现有集群。如果没有现有集群，请参阅 开始使用 Amazon EKS (p. 4)。如果您需要更新现有集群的版本，请参阅 更新 Amazon EKS 集群 Kubernetes 版本 (p. 28)。
- 在集群上部署 Amazon Load Balancer Controller。有关更多信息，请参阅安装 Amazon Load Balancer Controller 附加组件 (p. 287)。对于 `1.19` 或更高版本的集群，我们建议使用 `2.4.3` 或更高版本。如果您的集群早于 `1.19` 版本，则我们建议使用 `2.3.1` 版本。
- 至少两个子网位于不同的可用区。Amazon Load Balancer Controller 从每个可用区中选择一个子网。如果在一个可用区中发现多个标记子网，则该控制器会按子网 ID 的字典顺序选择第子网。每个子网必须具有至少 8 个可用 IP 地址。

  如果您使用的是挂载到 Worker 节点的多个安全组，则必须按如下方式标记一个安全组。将 *`my-cluster`* 替换为您的集群名称。

  - 密钥 – `kubernetes.io/cluster/`*`my-cluster`*
  - 值 – `shared` 或 `owned`
- 如果您使用的是 Amazon Load Balancer Controller 版本 `2.1.1` 或更早版本，则必须按如下格式标记子网。如果使用版本 `2.1.2` 或更高版本，则此标记是可选的。但是，如果出现以下任一情况，我们建议您标记子网。您有多个集群在同一 VPC 中运行，或者有多个 Amazon 服务在 VPC 中共享子网。或者，您希望更好地控制为每个集群配置的负载均衡器的位置。将 *`my-cluster`* 替换为您的集群名称。
  - 密钥 – `kubernetes.io/cluster/`*`my-cluster`*
  - 值 – `shared` 或 `owned`
- 您的公有子网和私有子网必须满足以下要求。除非您明确指定子网 ID 作为服务或入口对象的注释。假设您通过明确指定子网 ID 作为服务或入口对象的注释来预置负载均衡器。在这种情况下，Kubernetes 和 Amazon 负载均衡器控制器会直接使用这些子网创建负载均衡器，并且不需要以下标签。
  - 私有子网：必须采用以下格式标记。这样，Kubernetes 和 Amazon 负载均衡器控制器就会知道子网可用于内部负载均衡器。如果您在 2020 年 3 月 26 日之后使用 `eksctl` 或 Amazon EKS Amazon CloudFormation 模板创建 VPC，则在创建子网时会对子网进行适当标记。有关 Amazon EKS Amazon CloudFormation VPC 模板的更多信息，请参阅 为 Amazon EKS 集群创建 VPC (p. 228)。
    - 密钥 – `kubernetes.io/role/internal-elb`
    - 值 – `1`
  - 公有子网：必须采用以下格式标记。如此一来，Kubernetes 将知道仅使用为外部负载均衡器指定的子网。这样，Kubernetes 就不会在每个可用区中选择一个公有子网（根据其子网 ID 按词典式顺序排列）。如果您在 2020 年 3 月 26 日之后使用 `eksctl` 或 Amazon EKS Amazon CloudFormation 模板创建 VPC，则在创建子网时会对子网进行适当标记。有关 Amazon EKS Amazon CloudFormation VPC 模板的更多信息，请参阅 为 Amazon EKS 集群创建 VPC (p. 228)。
    - 密钥 – `kubernetes.io/role/elb`
    - 值 – `1`

  如果未显式添加子网角色标签，则 Kubernetes 服务控制器将检查您的集群 VPC 子网的路由表。这是为了确定子网是私有还是公有。我们建议您不要依赖此行为。相反，明确添加私有或公有角色标签。Amazon Load Balancer Controller 不检查路由表。它还需要私有标签和公有标签才能成功实现自动发现。

注意事项

- 每当使用 `kubernetes.io/ingress.class: alb` 注释在集群上创建 Kubernetes 入口资源时，Amazon Load Balancer Controller 就会创建 ALB 和必要的支持 Amazon 资源。入口资源会配置 ALB 以便将 HTTP 或 HTTPS 流量路由到集群中的其他 pods。要确保您的入口对象使用 Amazon Load Balancer Controller，请将以下注释添加到您的 Kubernetes 入口规范。有关更多信息，请参阅 GitHub 上的入口规范。

```
annotations:
    kubernetes.io/ingress.class: alb
```

  > **Note**
  >
  > 如果要对 `IPv6` pods 进行负载均衡，请将以下注释添加到您的入口规范。您只能通过 `IPv6` 对 IP 目标进行负载均衡，无法对实例目标进行负载均衡。如果没有此注释，则通过 `IPv4` 进行负载均衡。
  >
  > ```
  > alb.ingress.kubernetes.io/ip-address-type: dualstack
  > ```

- Amazon Load Balancer Controller 支持以下流量模式：
  - 实例 – 将您的集群中的节点注册为 ALB 的目标。传输到 ALB 的流量将路由到您的服务的 `NodePort`，然后转发到您的 pods。这是默认流量模式。您也可以使用 `alb.ingress.kubernetes.io/target-type: instance` 注释明确地指定该模式。

    > **Note**
    >
    > 您的 Kubernetes 服务必须指定 `NodePort` 或“LoadBalancer”类型，才能使用此流量模式。

  - IP – 将 pods 注册为 ALB 的目标。传输到 ALB 的流量将直接路由到您的服务的 pods。您必须指定 `alb.ingress.kubernetes.io/target-type: ip` 注释，才能使用此流量模式。当目标 pods 在 Fargate 上运行时，必须使用 IP 目标类型。
- 要标记由控制器创建的 ALB，请向控制器添加以下注释：`alb.ingress.kubernetes.io/tags`。有关 Amazon Load Balancer Controller 支持的所有可用注释的列表，请参阅 GitHub 上的入口注释。
- 升级或降级 ALB 控制器版本可能会对依赖它的功能带来重大变化。有关在每个版本中引入的突发性更改的更多信息，请参阅 GitHub 上的 ALB 控制器发布说明。

使用 `IngressGroups` 跨多个服务资源共享 Application Load Balancer

要将入口加入到组，请将以下注释添加到 Kubernetes 入口资源规范中。

```
alb.ingress.kubernetes.io/group.name: my-group
```

组名称必须为：

- 长度不超过 63 个字符。
- 它们包含小写字母、数字、- 和 .。
- 以字母或数字开头和结尾。

控制器将自动合并同一入口组中所有入口的入口规则。它用单个 ALB 提供支持。在入口上定义的大多数注释仅适用于由该入口定义的路径。默认情况下，入口资源不属于任何入口组。

> **Warning**
>
> 潜在的安全风险：仅当所有具有创建或修改入口资源的 RBAC 权限的 Kubernetes 用户均在同一信任边界内时，为入口指定一个入口组。如果添加具有组名称的注释，则其他 Kubernetes 用户可创建或修改其入口，以使其属于同一入口组。这样做可能会导致不良行为，例如使用优先级更高的规则覆盖现有规则。

您可以添加入口资源的编号。

```
alb.ingress.kubernetes.io/group.order: '10'
```

该号码可以为 1-1000。首先计算同一入口组中所有入口的最小数目。没有此注释的所有入口将使用零值进行评估。数字越大的重复规则可以覆盖具有较小数字的规则。默认情况下，同一入口组中不同入口之间的规则顺序由入口的命名空间和名称的字典顺序决定。

Important

确保同一入口组中的每个入口都具有唯一的优先级编号。不同入口中不能有重复的编号。

（可选）部署示例应用程序

先决条件

- 集群 VPC 中至少有一个公有或私有子网。
- 在集群上部署 Amazon Load Balancer Controller。有关更多信息，请参阅安装 Amazon Load Balancer Controller 附加组件 (p. 287)。建议升级到版本 `2.4.3` 或更高版本。

部署示例应用程序

您可以在具有 Amazon EC2 节点、Fargate pods 或这两者的集群上运行示例应用程序。

1. 如果您不部署到 Fargate，请跳过此步骤。如果您要部署到 Fargate，请创建一个 Fargate 配置文件。您可以通过运行以下命令来创建配置文件，也可以在 Amazon Web Services Management Console (p. 133) 中，使用该命令中 `name` 和 `namespace` 的相同值创建配置文件。将 *example values* 替换为您自己的值。

   ```
   eksctl create fargateprofile \
       --cluster my-cluster \
       --region region-code \
       --name alb-sample-app \
       --namespace game-2048
   ```

2. 将游戏 2048 部署为示例应用程序，以确认作为入口对象的结果，Amazon Load Balancer Controller 是否会创建 Amazon ALB。完成您要部署到的子网类型的步骤。

   a. 如果要部署到使用 `IPv6` 系列创建的集群中的 pods，请跳至下一步。

   - Public

     ```
     kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.3/docs/examples/2048/2048_full.yaml
     ```

   - 私有

     1. 下载清单。

        ```
        curl -o 2048_full.yaml https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.3/docs/examples/2048/2048_full.yaml
        ```

     2. 编辑文件并找到显示 `alb.ingress.kubernetes.io/scheme: internet-facing` 的行。
     3. 将 *internet-facing* 更改为 **internal** 然后保存文件。
     4. 将清单应用于集群。

```
kubectl apply -f 2048_full.yaml
```

b. 如果要部署到使用 IPv6 系列 (p. 248)创建的集群中的 pods，请完成以下步骤。

1. 下载清单。

```
curl -o 2048_full.yaml https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.3/docs/examples/2048/2048_full.yaml
```

2. 在编辑器中打开文件，并将以下行添加到入口规范的注释中。

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

3. 如果要对内部 pods（而不是面向互联网的 pods）进行负载均衡，请将显示 `alb.ingress.kubernetes.io/scheme: internet-facing` 的行更改为 `alb.ingress.kubernetes.io/scheme: internal`
4. 保存该文件。
5. 将清单应用于集群。

```
kubectl apply -f 2048_full.yaml
```

3. 几分钟后，验证是否已使用以下命令创建入口资源。

```
kubectl get ingress/ingress-2048 -n game-2048
```

输出示例如下。

```
NAME           CLASS    HOSTS   ADDRESS
                   PORTS   AGE
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-code.elb.amazonaws.com.cn   80      2m32s
```

Note

如果您在私有子网中创建了负载均衡器，则之前输出中 ADDRESS 下的值前面加上 internal-。
如果在几分钟后尚未成功创建入口，请运行以下命令以查看 Amazon Load Balancer Controller 日志。这些日志包含可让您诊断部署中问题的错误消息。

```
kubectl logs -n kube-system deployment.apps/aws-load-balancer-controller
```

4. 如果您部署到了公有子网，请打开浏览器并从上一命令输出导航到 ADDRESS URL 以查看示例应用程序。如果您没有看到任何内容，请刷新浏览器并重试。如果您已部署到私有子网，则需要从 VPC 中的设备（例如堡垒主机）查看页面。有关更多信息，请参阅 Linux 上的 Amazon 堡垒机主机。
5. 在完成对示例应用程序的试验后，通过运行以下命令之一将其删除。

- 如果您应用了清单，而不是应用下载的副本，请使用以下命令。

```
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.3/docs/examples/2048/2048_full.yaml
```

- 如果您下载并编辑了清单，请使用以下命令。

```
kubectl delete -f 2048_full.yaml
```

限制可分配给服务的外部 IP 地址

可以通过以下方式从集群内部访问 Kubernetes 服务：

- Kubernetes 自动分配的集群 IP 地址
- 您在服务规格中为 `externalIPs` 属性指定的任何 IP 地址。外部 IP 地址不由 Kubernetes 管理，而是由集群管理员负责。使用 `externalIPs` 指定的外部 IP 地址不同于由云提供商分配给 `LoadBalancer` 类型服务的外部 IP 地址。

要了解有关 Kubernetes 服务的更多信息，请参阅 Kubernetes 文档中的服务。您能够限制可在服务规格中为 `externalIPs` 指定的 IP 地址。

限制在服务规格中可为 **`externalIPs`** 指定的 IP 地址

1. 部署 cert-manager 来管理 Webhook 证书。有关更多信息，请参阅 cert-manager 文档。

```
kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert-manager.yaml
```

2. 验证 cert-manager pods 是否正在运行。

```
kubectl get pods -n cert-manager
```

输出示例如下。

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-58c8844bb8-nlx7q              1/1     Running   0          15s
cert-manager-cainjector-745768f6ff-696h5   1/1     Running   0          15s
cert-manager-webhook-67cc76975b-4v4nk      1/1     Running   0          14s
```

3. 检查您的现有服务，以确保为它们分配的外部 IP 地址都包含在您要将地址限制到的 CIDR 块之中。

```
kubectl get services -A
```

输出示例如下。

```
NAMESPACE                       NAME                                      TYPE
 CLUSTER-IP       EXTERNAL-IP     PORT(S)         AGE
cert-manager                    cert-manager                              ClusterIP
 10.100.102.137   <none>          9402/TCP        20m
cert-manager                    cert-manager-webhook                      ClusterIP
 10.100.6.136     <none>          443/TCP         20m
default                         kubernetes                                ClusterIP
 10.100.0.1       <none>          443/TCP         2d1h
externalip-validation-system    externalip-validation-webhook-service     ClusterIP
 10.100.234.179   <none>          443/TCP         16s
kube-system                     kube-dns                                  ClusterIP
 10.100.0.10      <none>          53/UDP,53/TCP   2d1h
my-namespace                    my-service                                ClusterIP
 10.100.128.10    192.168.1.1     80/TCP          149m
```

如果其中有任何一个 IP 地址值不在要将访问限制到的块之内，则需要将这些地址更改为块的范围内，然后重新部署服务。例如，上面输出中的 `my-service` 服务获得分配的一个外部 IP 地址不在步骤 5 中 CIDR 块示例范围内。

4. 下载外部 IP Webhook 清单。您还可以在 GitHub 上查看 Webhook 的源代码。

5. 指定 CIDR 块。在编辑器中打开下载的文件并删除下面代码行开头的 #。

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

将 10.0.0.0/8 替换为您自己的 CIDR 块。您可以根据需要指定任意数量的块。如果指定多个块，请在块之间添加逗号。

6. 则将文件中的 us-west-2、602401143452 和 amazonaws.com 替换为以下命令。在运行命令之前，为 Amazon Web Services 区域 将 *region-code* 和 *111122223333* 替换为来自 Amazon 容器镜像注册表 (p. 340) 列表中的值。

```
sed -i.bak -e 's|602401143452|111122223333|' externalip-webhook.yaml
sed -i.bak -e 's|us-west-2|region-code|' externalip-webhook.yaml
sed -i.bak -e 's|amazonaws.com|amazonaws.com.cn|' externalip-webhook.yaml
```

7. 将清单应用于集群。

```
kubectl apply -f externalip-webhook.yaml
```

若尝试将服务部署到为 externalIPs 指定的 IP 地址不在指定 CIDR 块 (p. 338)步骤中指定的地址块范围内的集群，则部署将会失败。

将容器镜像从一个存储库复制到另一个存储库

本主题介绍如何从节点无权访问的存储库中提取容器镜像，然后将该镜像推送到节点有权访问的存储库。您可以将镜像推送到 Amazon ECR 或节点有权访问的备用存储库。

先决条件

- 您的计算机上安装和配置的 Docker 引擎。有关说明，请参阅 Docker 文档中的安装 Docker 引擎。
- 您的设备或 Amazon CloudShell 上安装并配置了 2.7.21 版或更高版本，或 1.25.46 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 aws configure 进行快速配置。
- 如果您希望节点通过 Amazon 的网络从私有 Amazon ECR 存储库中提取容器镜像或将容器镜像推送到私有 Amazon ECR 存储库，请为 Amazon ECR 创建接口 VPC 端点。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的为 Amazon ECR 创建 VPC 端点。

请完成以下步骤，以从存储库中提取容器镜像并将其推送到自己的存储库。在本主题提供的以下示例中，将提取 指标帮助程序 (p. 283) 的镜像。按照这些步骤操作时，请确保将 *example values* 替换为您自己的值。

将容器镜像从一个存储库复制到另一个存储库

1. 如果您还没有 Amazon ECR 存储库或其他存储库，请创建一个节点可以访问的存储库。以下命令将创建一个 Amazon ECR 私有存储库。Amazon ECR 私有存储库的名称必须以字母开头。它只能包含小写字母、数字、连字符 (-)、下划线 (_) 和正斜杠 (/)。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的创建私有存储库。

   您可以将 *cni-metrics-helper* 替换为您选择的任何名称。作为最佳实践，请为每个镜像创建单独的存储库。我们建议您这样做，因为镜像标签在存储库中必须唯一。将 *region-code* 替换为 Amazon ECR 支持的 Amazon Web Services 区域。

```
aws ecr create-repository --region region-code --repository-name cni-metrics-helper
```

2. 请确定节点需要提取的镜像的注册表、存储库和标签（可选）。此信息采用 `registry/repository[:tag]` 格式。

   许多关于安装镜像的 Amazon EKS 主题都要求您应用清单文件或使用 Helm 图表安装镜像。但是，在应用清单文件或安装 Helm 图表之前，请先查看清单或图表 `values.yaml` 文件的内容。这样您就可以确定要提取的注册表、存储库和标签。

   例如，您可以在 指标帮助程序 (p. 283) 的清单文件中找到以下行。`602401143452.dkr.ecr.us-west-2.amazonaws.com` 注册表是 Amazon ECR 私有注册表。存储库是 `cni-metrics-helper`。

   ```
   image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.11.3"
   ```

   您可能会看到镜像位置的以下变体：

   - 仅限 `repository-name:tag`。在本例中，`docker.io` 通常是未指定的注册表，因为如果没有指定注册表，默认情况下 Kubernetes 会将它置于存储库名称前面。
   - `repository-name/repository-namespace/repository:tag`。存储库命名空间为可选，但有时由存储库所有者指定以用于对镜像进行分类。例如，Amazon ECR Public Gallery 中的所有 Amazon EC2 镜像都使用 `aws-ec2` 命名空间。

   在使用 Helm 安装镜像之前，请查看 Helm `values.yaml` 文件以确定镜像位置。例如，指标帮助程序 (p. 283) 的 `values.yaml` 文件包括以下行。

   ```
   image:
     region: us-west-2
     tag: v1.11.3
     account: "602401143452"
     domain: "amazonaws.com"
   ```

3. 请提取清单文件中指定的容器镜像。

   a. 如果从公有注册表中提取，例如 Amazon ECR Public Gallery，则可以跳到下一个子步骤，因为不需要身份验证。在此示例中，您可以对包含 CNI 指标帮助程序映像存储库的 Amazon ECR 私有注册表进行身份验证。Amazon EKS 在 Amazon 容器镜像注册表 (p. 340) 中所列的每个注册表中维护映像。通过将 *602401143452918309763551* 和 *region-code* 替换为其他注册表的信息，您可以对任何注册表进行身份验证。支持 Amazon EKS 的每个Amazon Web Services 区域 都存在一个单独的注册表。

   ```
   aws ecr get-login-password --region region-code | docker login --username Amazon --password-stdin 918309763551.dkr.ecr.region-code.amazonaws.com.cn
   ```

   b. 提取镜像。在此示例中，您从上一个子步骤中进行身份验证的注册表中提取。将 *961992271922* 和 *region-code* 替换为您在上一个子步骤中提供的信息。

   ```
   docker pull 918309763551.dkr.ecr.region-code.amazonaws.com.cn/cni-metrics-helper:v1.11.3
   ```

4. 请使用您的注册表、存储库和标签来标记提取的镜像。以下示例假设您从清单文件中提取镜像，然后将其推送到您在第一步中创建的 Amazon ECR 私有存储库。请将 *111122223333* 替换为您的账户 ID。将 *region-code* 替换为您在其中创建 Amazon ECR 私有存储库的 Amazon Web Services 区域。

   ```
   docker tag cni-metrics-helper:v1.11.3 111122223333.dkr.ecr.region-code.amazonaws.com.cn/cni-metrics-helper:v1.11.3
   ```

5. 对注册表进行身份验证。在此示例中，您可以对在第一步中创建的 Amazon ECR 私有注册表进行身份验证。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的注册表身份验证。

```
aws ecr get-login-password --region region-code | docker login --username Amazon --password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com.cn
```

6. 将镜像推送到存储库。在此示例中，您将镜像推送到在第一步中创建的 Amazon ECR 私有存储库。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的推送 Docker 镜像。

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com.cn/cni-metrics-helper:v1.11.3
```

7. 请使用您推送的镜像 `registry/repository:tag` 来更新在上一步中用于确定镜像的清单文件。如果您使用 Helm 图表进行安装，通常可以选择指定 `registry/repository:tag`。安装图表时，请为您推送到存储库的镜像指定 `registry/repository:tag`。

Amazon 容器镜像注册表

当您将 安装 Amazon Load Balancer Controller 附加组件 (p. 287)、Amazon VPC CNI plugin for Kubernetes (p. 238)、`kube-proxy` (p. 305) 或存储驱动程序 (p. 194)等附加组件部署到集群时，您的节点可能会从 Amazon EKS Amazon ECR 私有存储库中提取容器镜像。镜像的注册表、存储库和标签是在清单或 Helm `values.yaml` 文件中指定的，这些文件将在您部署的每个附加组件的主题中引用。

Amazon EKS 会将镜像复制到 Amazon Web Services 区域 支持的每个 Amazon EKS 中的存储库。您的节点可以通过互联网从以下任何注册表中提取容器镜像。或者，如果您在 VPC 中为 Amazon ECR (Amazon PrivateLink) 创建了接口 VPC 端点，则该节点可以通过 Amazon 的网络提取镜像。注册表需要使用 Amazon IAM 账户进行身份验证。您的节点使用 Amazon EKS 节点 IAM 角色 (p. 427) 进行身份验证，该节点在与之关联的 AmazonEC2ContainerRegistryReadOnly 托管 IAM policy 中具有相关权限。

Amazon Web Services 区域	注册表
af-south-1	877085696533.dkr.ecr.af-south-1.amazonaws.com
ap-east-1	800184023465.dkr.ecr.ap-east-1.amazonaws.com
ap-northeast-1	602401143452.dkr.ecr.ap-northeast-1.amazonaws.com
ap-northeast-2	602401143452.dkr.ecr.ap-northeast-2.amazonaws.com
ap-northeast-3	602401143452.dkr.ecr.ap-northeast-3.amazonaws.com
ap-south-1	602401143452.dkr.ecr.ap-south-1.amazonaws.com
ap-southeast-1	602401143452.dkr.ecr.ap-southeast-1.amazonaws.com
ap-southeast-2	602401143452.dkr.ecr.ap-southeast-2.amazonaws.com
ap-southeast-3	296578399912.dkr.ecr.ap-southeast-3.amazonaws.com
ca-central-1	602401143452.dkr.ecr.ca-central-1.amazonaws.com

Amazon Web Services 区域	注册表
cn-north-1	918309763551.dkr.ecr.cn-north-1.amazonaws.com.cn
cn-northwest-1	961992271922.dkr.ecr.cn-northwest-1.amazonaws.com.cn
eu-central-1	602401143452.dkr.ecr.eu-central-1.amazonaws.com
eu-north-1	602401143452.dkr.ecr.eu-north-1.amazonaws.com
eu-south-1	590381155156.dkr.ecr.eu-south-1.amazonaws.com
eu-west-1	602401143452.dkr.ecr.eu-west-1.amazonaws.com
eu-west-2	602401143452.dkr.ecr.eu-west-2.amazonaws.com
eu-west-3	602401143452.dkr.ecr.eu-west-3.amazonaws.com
me-south-1	558608220178.dkr.ecr.me-south-1.amazonaws.com
sa-east-1	602401143452.dkr.ecr.sa-east-1.amazonaws.com
us-east-1	602401143452.dkr.ecr.us-east-1.amazonaws.com
us-east-2	602401143452.dkr.ecr.us-east-2.amazonaws.com
us-gov-east-1	151742754352.dkr.ecr.us-gov-east-1.amazonaws.com
us-gov-west-1	013241004608.dkr.ecr.us-gov-west-1.amazonaws.com
us-west-1	602401143452.dkr.ecr.us-west-1.amazonaws.com
us-west-2	602401143452.dkr.ecr.us-west-2.amazonaws.com

Amazon EKS 附加组件

附加组件是为 Kubernetes 应用程序提供辅助操作功能的软件，但并不特定于应用程序。这包括可观测性代理或 Kubernetes 驱动程序等软件，这些软件允许集群与用于联网、计算和存储的底层 Amazon 资源进行交互。附加组件软件通常由 Kubernetes 社区、Amazon 等云提供商或第三方供应商构建和维护。Amazon EKS 会自动为每个集群安装自我管理的附加组件，例如 Amazon VPC CNI plugin for Kubernetes、`kube-proxy` 和 CoreDNS。您可以更改附加组件的默认配置并在需要时加以更新。

Amazon EKS 附加组件为 Amazon EKS 集群的一组经策管附加组件提供安装和管理。Amazon EKS 附加组件包含最新的安全补丁、错误修复，并经 Amazon 验证能够与 Amazon EKS 一起使用。Amazon EKS 附加组件允许您始终如一地确保您的 Amazon EKS 集群安全稳定，并减少您在安装、配置和更新上所需执行的工作量。如果是类似于 `kube-proxy` 的自我管理附加组件，若已在您的集群上运行并且可作为 Amazon EKS 附加组件使用，那么您便可以安装 `kube-proxy` Amazon EKS 加载项开始享受 Amazon EKS 附加组件的功能所带来的益处。

您可以通过 Amazon EKS API 为 Amazon EKS 附加组件更新特定的 Amazon EKS 托管配置字段。在附加组件启动后，您还可以直接在 Kubernetes 集群中修改并非由 Amazon EKS 管理的配置字段，包括在适用时为附加组件定义特定的配置字段。Amazon EKS 不会覆盖您所做的这些更改。使用 Kubernetes 服务器端应用功能可以实现这一点。有关更多信息，请参阅Amazon EKS 附加组件配置 (p. 342)。

Amazon EKS 附加组件可以与 1.18 版或更高版本的任何 Amazon EKS 集群一起使用。集群可以包括自行管理的节点和 Amazon EKS 托管节点组以及 Fargate。

注意事项

- 要为集群配置附加组件，您的 IAM 用户必须具有 IAM 权限，以使用附加组件。如需了解更多信息，请参阅 Amazon Elastic Kubernetes Service 定义的操作 中的名称中带有 Addon 的操作。
- Amazon EKS 附加组件仅适用于运行 Kubernetes 版本 1.18 及更高版本的 Amazon EKS 集群。
- Amazon EKS 附加组件能在您为集群预置或配置的节点上运行。节点类型包括 Amazon EC2 实例和 Fargate。
- 您可以修改并非由 Amazon EKS 管理的字段以自定义 Amazon EKS 附加组件的安装。有关更多信息，请参阅Amazon EKS 附加组件配置 (p. 342)。
- 如果您使用 Amazon Web Services Management Console 创建集群，则 Amazon EKS kube-proxy、Amazon VPC CNI plugin for Kubernetes 和 CoreDNS Amazon EKS 附加组件会自动添加到您的集群中。如果您借助 eksctl 使用 config 文件创建集群，则 eksctl 还可以使用 Amazon EKS 附加组件创建集群。如果使用不带有 config 文件的 eksctl 或使用任何其他工具创建集群，则自我管理的 kube-proxy、Amazon VPC CNI plugin for Kubernetes 和 CoreDNS 附加组件将会安装，而不会安装 Amazon EKS 附加组件。您可以自行管理，也可以在集群创建后手动添加 Amazon EKS 附加组件。

您可以使用 Amazon EKS API、Amazon Web Services Management Console、Amazon CLI 和 eksctl 添加、更新或删除 Amazon EKS 附加组件。要了解使用 Amazon Web Services Management Console、Amazon CLI 和 eksctl 时的详细步骤，请参阅有关以下附加组件的主题：

- Amazon VPC CNI plugin for Kubernetes (p. 233)
- CoreDNS (p. 294)
- kube-proxy (p. 300)
- ADOT (p. 468)
- Amazon EBS CSI (p. 201)

您也可以使用 Amazon CloudFormation 创建 Amazon EKS 附加组件。

Amazon EKS 附加组件配置

Amazon EKS 使用标准的最佳实践配置，为您的集群安装附加组件。有关 Amazon EKS 附加组件的更多信息，请参阅 Amazon EKS 附加组件 (p. 341)。

您若想要启用高级功能，则建议自定义 Amazon EKS 附加组件的配置。Amazon EKS 使用 Kubernetes 服务器端应用功能来启用 Amazon EKS 对附加组件的管理，而不会覆盖您对并非由 Amazon EKS 管理的设置的配置。有关更多信息，请参阅 Kubernetes 文档中的服务器端应用。为此，Amazon EKS 为其安装的每个附加组件管理最小的字段组。您可以修改并非由 Amazon EKS 或其他 Kubernetes 控制面板进程（例如 kube-controller-manager）管理的所有字段，这样操作不会导致任何问题。

> Important
>
> 修改由 Amazon EKS 管理的字段会阻止 Amazon EKS 管理附加组件，并可能导致在更新附加组件时覆盖您所做的更改。

先决条件

- 现有 1.18 版或更高版本的 Amazon EKS 集群。
- 添加到集群中的 Amazon EKS 附加组件。有关向集群中添加 Amazon EKS 附加组件的更多信息，请参阅 Amazon EKS 附加组件 (p. 341)。

查看字段管理状态

您可以使用 `kubectl` 查看对于任何 Amazon EKS 附加组件，有哪些字段由 Amazon EKS 管理。

查看字段的管理状态

1. 确定要检查的附加组件。要查看部署到集群的所有 `deployments` 和 `DaemonSets`，请参阅 查看 Kubernetes 资源 (p. 458)。
2. 通过运行以下命令查看附加组件的托管字段：

```
kubectl get type/add-on-name -n add-on-namespace -o yaml
```

例如，您可以通过以下命令查看 CoreDNS 附加组件的托管字段。

```
kubectl get deployment/coredns -n kube-system -o yaml
```

字段管理列在所返回的输出中的以下部分中。

```
...
managedFields:
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
...
```

> **Note**
>
> 如果在输出中没有看到 `managedFields`，将 **`--show-managed-fields`** 添加到命令中，然后再次运行。您使用的 `kubectl` 版本决定预设情况下是否返回托管字段。

了解 Kubernetes API 中的字段管理语法

当您查看某个 Kubernetes 对象的详细信息时，输出中会同时返回托管和非托管字段。托管字段可以是以下任一类型：

- 完全托管：该字段的所有密钥都由 Amazon EKS 管理。修改任何值都会导致冲突。
- 部分托管：该字段的部分密钥将由 Amazon EKS 管理。只有对 Amazon EKS 明确管理的密钥进行修改才会导致冲突。

这两种类型的字段都标有 `manager: eks`。

每个键要么是表示字段本身的 `.`（始终映射到空集），要么是表示子字段或项目的字符串。字段管理的输出由以下类型的声明组成：

- `f:`*`name`*，其中 *`name`* 是列表中字段的名称。
- `k:`*`keys`*，其中 *`keys`* 是列表项字段的映射。
- `v:`*`value`*，其中 *`value`* 是列表项的确切 JSON 格式化值。
- `i:`*`index`*，其中 *`index`* 是列表中某个项的位置。

有关更多信息，请参阅 Kubernetes 文档中的 FieldsV1 v1 meta。

CoreDNS 附加组件的以下输出部分说明了上面的声明：

- 完全托管字段：如果托管字段指定了 `f:`（字段），但没有 `k:`（键），则整个字段都是托管的。修改此字段中的任何值都会导致冲突。

  在以下输出中，您可以看到名为 `coredns` 的容器由 `eks` 管理。`args`、`image` 和 `imagePullPolicy` 子字段也由 `eks` 管理。修改此字段中的任何值都会导致冲突。

```
...
f:containers:
  k:{"name":"coredns"}:
  .: {}
  f:args: {}
  f:image: {}
  f:imagePullPolicy: {}
...
manager: eks
...
```

- 部分托管字段：如果托管键具有指定的值，则为该字段管理所声明的键。修改指定的键会导致冲突。

  在以下输出中，您可以看到 `eks` 管理着包含 `name` 键的 `config-volume` 和 `tmp` 卷集。

```
...
f:volumes:
  k:{"name":"config-volume"}:
    .: {}
    f:configMap:
      f:items: {}
      f:name: {}
    f:name: {}
  k:{"name":"tmp"}:
    .: {}
    f:name: {}
...
manager: eks
...
```

- 向部分托管的字段添加键：如果只管理一个特定的键值，则可以安全地向字段添加其他键（如实际参数），而不会导致冲突。如果您添加了其他键，请先确保该字段不是托管字段。添加或修改任何托管值都会导致冲突。

  在以下输出中，您可以看到 `name` 键和 `name` 字段都是托管的。添加或修改任何容器名称都会导致与此托管键发生冲突。

```
...
f:containers:
  k:{"name":"coredns"}:
...
    f:name: {}
...
manager: eks
...
```

使用 Elastic Fabric Adapter 的 Machine Learning 培训

Important

此功能在 Amazon Web Services 区域（中国）不可用。

本主题介绍如何将 Elastic Fabric Adapter（EFA）与部署在 Amazon EKS 集群中的 pods 集成。Elastic Fabric Adapter (EFA) 是 Amazon EC2 实例的网络接口，使您能够在Amazon上运行要求大规模高级别节点间通信的应用程序。其量身定制的操作系统旁路硬件接口增强了实例间通信的性能，这对扩缩这些应用程序至关重要。借助 EFA，采用消息传递接口 (MPI) 的高性能计算 (HPC) 应用程序和采用 NVIDIA Collective Communications Library (NCCL) 的 Machine Learning (ML) 应用程序可以扩展到数千个 CPU 或 GPU。因此，您同时获得了本地 HPC 集群的应用程序性能，以及Amazon云的按需弹性和灵活度。将 EFA 与 Amazon EKS 集群上运行的应用程序集成，可以缩短完成大规模分布式培训工作负载的时间，而无需向集群添加更多实例。有关 EFA 的更多信息，请参阅 Elastic Fabric Adapter。

本主题中描述的 EFA 插件完全支持 Amazon EC2 `P4d` 实例，它代表了云中分布式 Machine Learning 的顶尖技术。每个 `p4d.24xlarge` 实例通过 EFA 拥有八个 NVIDIA A100 GPU 和 400 Gbps 的 GPUDirectRDMA。GPUDirectRDMA 使您能够通过 CPU 旁路在节点之间实现 GPU 到 GPU 的直接通信，从而增加集体通信带宽并降低延迟。Amazon EKS 和 EFA 与 `P4d` 实例的集成提供了一种无缝衔接的方法，可为分布式 Machine Learning 培训利用最高性能的 Amazon EC2 计算实例。

先决条件

- 现有 `1.19` 版 Amazon EKS 集群。如果您没有现有集群，请参考我们的某个 开始使用 Amazon EKS (p. 4) 指南创建一个集群。您的集群必须部署在至少具有一个私有子网的 VPC 中，该子网应当具有足够多的可用 IP 地址以便在其中部署节点。私有子网必须具有由外部设备（如 NAT 网关）提供的出站 Internet 访问。

  如果您计划使用 `eksctl` 创建节点组，`eksctl` 还可以为您创建 `1.19` 版集群。
- 您的设备或 Amazon CloudShell 上安装并配置了 `2.7.21` 版或更高版本，或 `1.25.46` 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 `aws configure` 进行快速配置。
- 您的设备或 Amazon CloudShell 上安装了 `kubectl` 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 `1.22`，则可以将 `kubectl` 版本 `1.21`、`1.22` 或 `1.23` 用于它。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl` (p. 4)。
- 在启动支持多个 Elastic Fabric Adapter 的 Worker 节点（例如 `p4d.24xlarge`）之前，您必须先安装 Amazon VPC CNI plugin for Kubernetes 版本 `1.7.10` 或更高版本。有关更新您的 CNI 版本的更多信息，请参阅 更新 Amazon VPC CNI plugin for Kubernetes 自行管理的附加组件 (p. 238)。

创建节点组

下面的程序通过一个包含 EFA 接口和 GPUDirect RDMA 且受 `p4d.24xlarge` 支持的节点组，帮助您创建节点组，并使用 EFA 针对多节点 NCCL 性能运行一个 NVIDIA Collective Communications Library (NCCL) 示例测试。该示例可用作使用 EFA 在 Amazon EKS 上进行分布式深度学习培训的模板。

1. 确定您想在其中部署节点的 Amazon Web Services 区域 中有哪些支持 EFA 的 Amazon EC2 实例类型可用。将 *`region-code`* 替换为您想要在其中部署节点组的 Amazon Web Services 区域。

   ```
   aws ec2 describe-instance-types --region region-code --filters Name=network-info.efa-supported,Values=true \
       --query "InstanceTypes[*].[InstanceType]" --output text
   ```

   部署节点时，您想要部署的实例类型必须在集群所在的 Amazon Web Services 区域 中可用。
2. 确定您想要部署的实例类型在哪些可用区中可用。本教程使用了 `p4d.24xlarge` 实例类型，必须在上一步所指定的 Amazon Web Services 区域 的输出中返回该实例类型。在生产集群中部署节点时，请将 *`p4d.24xlarge`* 替换为上一步中返回的任何实例类型。

   ```
   aws ec2 describe-instance-type-offerings --region region-code --location-type
    availability-zone --filters Name=instance-type,Values=p4d.24xlarge \
       --query 'InstanceTypeOfferings[*].Location' --output text
   ```

   输出示例如下。

```
cn-north-1a     cn-north-1c     cn-north-1b
```

记下返回的可用区，以供后续步骤使用。将节点部署到集群时，VPC 的子网必须在输出返回的其中一个可用区中具有可用的 IP 地址。

3. 使用 eksctl，或者使用 Amazon CLI 和 Amazon CloudFormation 创建节点组。

   eksctl

   先决条件

   您的设备或 Amazon CloudShell 上安装了 0.110.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 安装或更新 eksctl (p. 10)。

   1. 将以下内容复制到名为 *efa-cluster.yaml* 的文件中。将 *example values* 替换为您自己的值。您可以将 *p4d.24xlarge* 替换为不同的实例，但是如果您进行替换，请确保 availabilityZones 的值为第 1 步中针对实例类型返回的可用区。

      ```
      apiVersion: eksctl.io/v1alpha5
      kind: ClusterConfig

      metadata:
        name: my-efa-cluster
        region: region-code
        version: "1.19"

      iam:
        withOIDC: true

      availabilityZones: ["cn-north-1", "cn-north-1c"]

      managedNodeGroups:
        - name: my-efa-ng
          instanceType: p4d.24xlarge
          minSize: 1
          desiredCapacity: 2
          maxSize: 3
          availabilityZones: ["cn-north-1"]
          volumeSize: 300
          privateNetworking: true
          efaEnabled: true
      ```

   2. 在现有集群中创建托管节点组。

      ```
      eksctl create nodegroup -f efa-cluster.yaml
      ```

      如果您没有现有集群，可以运行以下命令来创建集群和节点组。

      ```
      eksctl create cluster -f efa-cluster.yaml
      ```

      Note

      由于此示例中使用的实例类型具有 GPU，eksctl 会在每个实例上为您自动安装 NVIDIA Kubernetes 设备插件。

Amazon CLI and Amazon CloudFormation

EFA 联网有几个要求，包括创建 EFA 特定的安全组，创建 Amazon EC2 置放群组，创建指定一个或多个 EFA 接口的启动模板，并将 EFA 驱动程序安装作为 Amazon EC2 用户数据的一部分。要了解有关 EFA 要求的更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 EFA 和 MPI 入门。以下步骤为您创建所有这些内容。将所有*###*替换为您自己的值。

1. 设置将在后面的步骤中使用的几个变量。将所有 *example values* 替换为您自己的值。将 *my-cluster* 替换为您的现有集群的名称。后面将使用 `node_group_resources_name` 的值来创建 Amazon CloudFormation 堆栈。后面将使用 `node_group_name` 的值在集群中创建节点组。

```
cluster_name="my-cluster"
cluster_region="region-code"
node_group_resources_name="my-efa-nodegroup-resources"
node_group_name="my-efa-nodegroup"
```

2. 在您的 VPC 中确定一个私有子网，该子网与您想要部署的实例类型位于相同的可用区中。
   a. 检索集群的版本并将其存储在变量中，以便在后续步骤中使用。

```
cluster_version=$(aws eks describe-cluster \
    --name $cluster_name \
    --query "cluster.version" \
    --output text)
```

   b. 检索您的集群所在的 VPC ID，并将其存储在变量中，以便在后续步骤中使用。

```
vpc_id=$(aws eks describe-cluster \
    --name $cluster_name \
    --query "cluster.resourcesVpcConfig.vpcId" \
    --output text)
```

   c. 检索集群的控制层面安全组的 ID，并将其存储在变量中，以便在后续步骤中使用。

```
control_plane_security_group=$(aws eks describe-cluster \
    --name $cluster_name \
    --query "cluster.resourcesVpcConfig.clusterSecurityGroupId" \
    --output text)
```

   d. 获取 VPC 中位于第 1 步返回的可用区中的子网 ID 列表。

```
aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$vpc_id" "Name=availability-zone,Values=cn-north-1" \
    --query 'Subnets[*].SubnetId' \
    --output text
```

   如果没有返回输出，请尝试第 1 步中返回的其他可用区。如果您的子网都不在第 1 步返回的可用区中，则您需要在第 1 步返回的可用区中创建一个子网。如果您的 VPC 中没有空间再创建子网，则可能需要在新 VPC 中创建一个新集群。

   e. 通过检查子网的路由表来确定子网是否为私有子网。

```
aws ec2 describe-route-tables \
    --filter Name=association.subnet-id,Values=subnet-0d403852a65210a29 \
    --query "RouteTables[].Routes[].GatewayId" \
    --output text
```

输出示例如下。

```
local
```

如果输出为 `local igw-02adc64c1b72722e2`，则说明该子网是公有子网。您必须在第 1 步返回的可用区中选择一个私有子网。确定私有子网后，请记下其 ID 以便后面的步骤中使用。

f. 使用上一步中的私有子网 ID 设置一个变量，以便在后续步骤中使用。

```
subnet_id=your-subnet-id
```

3. 下载 Amazon CloudFormation 模板。

```
curl -o efa-p4d-managed-nodegroup.yaml https://raw.githubusercontent.com/aws-samples/aws-efa-eks/main/cloudformation/efa-p4d-managed-nodegroup.yaml
```

4. 将下面的文本复制到计算机上。将 *p4d.24xlarge* 替换为第 1 步中的实例类型。将 *subnet-0d403852a65210a29* 替换为您在第 2.b.v 步中标识的私有子网的 ID。将 *path-to-downloaded-cfn-template* 替换为您在上一步中下载的 `efa-p4d-managed-nodegroup.yaml` 的路径。将 *your-public-key-name* 替换为您的公有密钥的名称。完成替换后，运行修改后的命令。

```
aws cloudformation create-stack \
    --stack-name ${node_group_resources_name} \
    --capabilities CAPABILITY_IAM \
    --template-body file://path-to-downloaded-cfn-template \
    --parameters \
       ParameterKey=ClusterName,ParameterValue=${cluster_name} \
       ParameterKey=ClusterControlPlaneSecurityGroup,ParameterValue=
${control_plane_security_group} \
       ParameterKey=VpcId,ParameterValue=${vpc_id} \
       ParameterKey=SubnetId,ParameterValue=${subnet_id} \
       ParameterKey=NodeGroupName,ParameterValue=${node_group_name} \
       ParameterKey=NodeImageIdSSMParam,ParameterValue=/aws/service/eks/
optimized-ami/${cluster_version}/amazon-linux-2-gpu/recommended/image_id \
       ParameterKey=KeyName,ParameterValue=your-public-key-name \
       ParameterKey=NodeInstanceType,ParameterValue=p4d.24xlarge
```

5. 确定何时部署您在上一步中部署的堆栈。

```
aws cloudformation wait stack-create-complete --stack-name
 $node_group_resources_name
```

上一个命令没有产生输出，但是您的 shell 提示符要直到创建堆栈后才会返回。

6. 使用上一步中 Amazon CloudFormation 堆栈所创建的资源创建您的节点组。

a. 从已部署的 Amazon CloudFormation 堆栈检索信息并将其存储在变量中。

```
node_instance_role=$(aws cloudformation describe-stacks \
    --stack-name $node_group_resources_name \
    --query='Stacks[].Outputs[?OutputKey==`NodeInstanceRole`].OutputValue' \
    --output text)
launch_template=$(aws cloudformation describe-stacks \
    --stack-name $node_group_resources_name \
    --query='Stacks[].Outputs[?OutputKey==`LaunchTemplateID`].OutputValue' \
    --output text)
```

b. 使用在上一步中创建的启动模板和节点 IAM 角色创建一个托管节点组。

```
aws eks create-nodegroup \
    --cluster-name $cluster_name \
    --nodegroup-name $node_group_name \
    --node-role $node_instance_role \
    --subnets $subnet_id \
    --launch-template id=$launch_template,version=1
```

c. 确认已创建节点。

```
aws eks describe-nodegroup \
   --cluster-name ${cluster_name} \
   --nodegroup-name ${node_group_name} | jq -r .nodegroup.status
```

在上一个命令返回的状态变为 `ACTIVE` 之前，请勿继续操作。节点可能需要几分钟才能准备就绪。

7. 如果您使用 GPU 部署了实例类型，请部署 NVIDIA Kubernetes 设备插件。

```
kubetl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/
v0.9.0/nvidia-device-plugin.yml
```

4. 部署 EFA Kubernetes 设备插件。

EFA Kubernetes 设备插件检测 EFA 接口并将其通告为可分配给 Kubernetes 的资源。应用程序可以使用 pod 请求规范中的扩展资源类型 `vpc.amazonaws.com/efa`，就像 CPU 和内存一样。有关更多信息，请参阅 Kubernetes 文档中的使用扩展资源。请求后，该插件会自动为 pod 分配并挂载 EFA 接口。使用该设备插件可以简化 EFA 设置，而且不需要在特权模式下运行 pod。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/aws-efa-eks/main/
manifest/efa-k8s-device-plugin.yml
```

（可选）部署示例 EFA 兼容应用程序

部署 Kubeflow MPI Operator

对于 NCCL 测试，您可以应用 Kubeflow MPI Operator。MPI Operator 可以轻松地在 Kubernetes 上运行 Allreduce 模式的分布式训练。有关更多信息，请参阅 GitHub 上的 MPI Operator。

```
kubectl apply -f https://raw.githubusercontent.com/kubeflow/mpi-operator/master/deploy/
v1alpha2/mpi-operator.yaml
```

运行多节点 NCCL 性能测试以验证 GPUDirectrDMA/EFA

要通过 EFA 使用 GPUDirectRDMA 验证 NCCL 性能，请运行标准 NCCL 性能测试。有关更多信息，请参阅 GitHub 上的官方 NCCL 测试存储库。您可以使用随此测试提供的示例 Dockerfile，它已经针对 CUDA `11.2` 和最新版本 EFA 进行构建。

或者，您也可以下载 Amazon ECR 存储库中提供的 Amazon Docker 镜像。

Important

在 Kubernetes 中采用 EFA 时需要考虑的一个重要因素，是将 Huge Pages 作为集群中的资源进行配置和管理。有关更多信息，请参阅 Kubernetes 文档中的管理 Huge Pages。安装了 EFA 驱动程序的 Amazon EC2 实例预先分配 5128 个 2M 的 Huge Pages，您可以请求将它们作为资源在您的作业规范中使用。

完成以下步骤以运行双节点 NCCL 性能测试。在示例 NCCL 测试作业中，每个工作线程请求八个 GPU、5210Mi 的 Hugepages-2mi、四个 EFA 和 8000Mi 的内存，这意味着每个工作线程都会使用 `p4d.24xlarge` 实例的所有资源。

1. 创建 NCCL 测试作业。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/aws-efa-eks/main/examples/simple/nccl-efa-tests.yaml
```

输出示例如下。

```
已创建 mpijob.kubeflow.org/nccl-tests-efa
```

2. 查看您运行的 pods。

```
kubectl get pods
```

输出示例如下。

```
NAME                                 READY   STATUS     RESTARTS   AGE
nccl-tests-efa-launcher-nbql9   0/1     Init:0/1   0          2m49s
nccl-tests-efa-worker-0           1/1     Running    0          2m49s
nccl-tests-efa-worker-1           1/1     Running    0          2m49s
```

MPI Operator 创建一个启动器 pod 和 2 个工作线程 pods（每个节点上一个）。

3. 查看 `efa-launcher` pod 的日志。将 *wzr8j* 替换为输出中的值。

```
kubectl logs -f nccl-tests-efa-launcher-nbql9
```

有关更多示例，请参阅 GitHub 上的 Amazon EKS EFA 示例存储库。

使用 Amazon Inferentia 进行的 Machine Learning 推理

本主题介绍如何创建一个包含运行 Amazon EC2 Inf1 实例的节点的 Amazon EKS 集群，并（可选地）部署示例应用程序。Amazon EC2 Inf1 实例由 Amazon Inferentia 芯片提供支持，这些芯片由Amazon定制构建，可在云中提供高性能和最低成本的推理。Machine Learning 模型通过 Amazon Neuron 部署到容器，而 Amazon Neuron 是一个专用的软件开发工具包 (SDK)，由编译器、运行时间和分析工具组成，可用于优化 Inferentia 芯片的 Machine Learning 推理性能。AmazonNeuron 支持常用的 Machine Learning 框架，例如 TensorFlow、PyTorch 和 MXNet。

> **Note**
>
> Neuron 设备逻辑 ID 必须是连续的。如果在 `inf1.6xlarge` 或 `inf1.24xlarge` 实例类型（具有多个 Neuron 设备）上调度请求多个 Neuron 设备的 pod，则当 Kubernetes 调度程序选择不连续的设备 ID 时，该 pod 将无法启动。有关更多信息，请参阅 GitHub 上的设备逻辑 ID 必须是连续的。

先决条件

- 在计算机上安装 `eksctl`。如果未安装，请参阅 安装或更新 `eksctl` (p. 10) 查看安装说明。
- 在计算机上安装 `kubectl`。有关更多信息，请参阅安装或更新 `kubectl` (p. 4)。

- （可选）在计算机上安装 python3。如果未安装，请参阅 Python 下载以查看安装说明。

创建集群

创建包含 Inf1 Amazon EC2 实例节点的 Amazon EKS 集群。

1. 创建包含 Inf1 Amazon EC2 实例节点的 Amazon EKS 集群。您可以将 *inf1.2xlarge* 替换为任何 Inf1 实例类型。eksctl 实用程序检测到您在使用 Inf1 实例类型启动节点组，并将使用一个 Amazon EKS 优化版加速型 Amazon Linux AMI 来启动您的节点。

   Note

   您不能将服务账户的 IAM 角色 (p. 395)与 TensorFlow Serving 结合使用。

   ```
   eksctl create cluster \
       --name inferentia \
       --region region-code \
       --nodegroup-name ng-inf1 \
       --node-type inf1.2xlarge \
       --nodes 2 \
       --nodes-min 1 \
       --nodes-max 4 \
       --ssh-access \
       --ssh-public-key your-key \
       --with-oidc
   ```

   Note

   请记住下一个输出行的值。在后面的（可选）步骤中将使用该值。

   ```
   [9]  adding identity "arn:aws:iam::111122223333:role/
   eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09" to auth
    ConfigMap
   ```

   在启动包含 Inf1 实例的节点组时，eksctl 将自动安装 Amazon Neuron Kubernetes 设备插件。此插件将 Neuron 设备作为系统资源传播到 Kubernetes 调度程序，以供容器请求。除了默认的 Amazon EKS 节点 IAM policy 之外，还添加了 Amazon S3 只读访问策略，以便下一个步骤中所述的示例应用程序能够从 Amazon S3 加载经过训练的模型。
2. 确保所有 pods 已正常启动。

   ```
   kubectl get pods -n kube-system
   ```

   缩减的输出：

   ```
   NAME                                   READY   STATUS    RESTARTS   AGE
   ...
   neuron-device-plugin-daemonset-6djhp   1/1     Running   0          5m
   neuron-device-plugin-daemonset-hwjsj   1/1     Running   0          5m
   ```

（可选）部署 TensorFlow Serving 应用程序映像

经过训练的模型必须先编译为 Inferentia 目标，才能部署在 Inferentia 实例上。要继续，您将需要一个在 Amazon S3 中保存的经 Neuron 优化的 TensorFlow 模型。如果您还没有 SavedModel，请按照创建 Neuron 兼容的 Resnet50 模型教程操作，并将生成的 SavedModel 上载到 S3。ResNet-50 是一种常用

的 Machine Learning 模型，用于图像识别任务。有关如何编译 Neuron 模型的更多信息，请参阅 Amazon Deep Learning AMI 开发人员指南中的带有 DLAMI 的 Amazon Inferentia 芯片。

示例部署清单管理一个预构建的推理服务容器，用于由 Amazon Deep Learning Containers 提供的 TensorFlow。容器内部是 Amazon Neuron 运行时间和 TensorFlow 服务应用程序。在 GitHub 上的可用镜像下，维护着一个针对 Neuron 优化的预构建深度学习容器的完整列表。在启动时，DLC 将从 Amazon S3 中提取您的模型，使用保存的模型启动 Neuron TensorFlow Serving，然后等待预测请求。

可以通过更改部署 yaml 中的 `aws.amazon.com/neuron` 资源，来调整分配给您的服务应用程序的 Neuron 设备的数量。请注意，TensorFlow Serving 和 Neuron 运行时间之间通过 GRPC 进行通信，这需要将 `IPC_LOCK` 功能传递给容器。

1. 将 `AmazonS3ReadOnlyAccess` IAM policy 添加到已在 创建集群 (p. 351) 的第 1 步中创建的节点实例角色。必须执行此操作，示例应用程序才能从 Amazon S3 加载经过训练的模型。

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
    --role-name eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09
```

2. 使用以下内容创建名为 `rn50_deployment.yaml` 的文件。更新区域代码和模型路径以匹配您需要的设置。当客户端向 TensorFlow 服务器发出请求时，模型名称用于标识目的。此示例使用模型名称匹配一个示例 ResNet50 客户端脚本，后面的步骤中将使用该脚本发送预测请求。

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
    role: master
spec:
  replicas: 2
  selector:
    matchLabels:
      app: eks-neuron-test
      role: master
  template:
    metadata:
      labels:
        app: eks-neuron-test
        role: master
    spec:
      containers:
        - name: eks-neuron-test
          image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
neuron:1.15.4-neuron-py37-ubuntu18.04
          command:
            - /usr/local/bin/entrypoint.sh
          args:
            - --port=8500
            - --rest_api_port=9000
            - --model_name=resnet50_neuron
            - --model_base_path=s3://your-bucket-of-models/resnet50_neuron/
          ports:
            - containerPort: 8500
            - containerPort: 9000
          imagePullPolicy: IfNotPresent
          env:
            - name: AWS_REGION
              value: "us-east-1"
            - name: S3_USE_HTTPS
              value: "1"
            - name: S3_VERIFY_SSL
```

```
            value: "0"
          - name: S3_ENDPOINT
            value: s3.us-east-1.amazonaws.com
          - name: AWS_LOG_LEVEL
            value: "3"
        resources:
          limits:
            cpu: 4
            memory: 4Gi
            aws.amazon.com/neuron: 1
          requests:
            cpu: "1"
            memory: 1Gi
        securityContext:
          capabilities:
            add:
              - IPC_LOCK
```

3. 部署模型。

```
kubectl apply -f rn50_deployment.yaml
```

4. 使用以下内容创建名为 `rn50_service.yaml` 的文件。这将打开 HTTP 和 gRPC 端口以接受预测请求。

```
kind: Service
apiVersion: v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
spec:
  type: ClusterIP
  ports:
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: eks-neuron-test
    role: master
```

5. 为 TensorFlow 模型 Serving 应用程序创建 Kubernetes 服务。

```
kubectl apply -f rn50_service.yaml
```

（可选）根据 TensorFlow Serving 服务进行预测

1. 要在本地进行测试，请将 gRPC 端口转发到 `eks-neuron-test` 服务。

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. 创建一个名为 `tensorflow-model-server-infer.py` 的 Python 脚本，其中包含以下内容。该脚本通过 GRPC（这是一个服务框架）运行推理过程。

```
import numpy as np
   import grpc
```

```
   import tensorflow as tf
   from tensorflow.keras.preprocessing import image
   from tensorflow.keras.applications.resnet50 import preprocess_input
   from tensorflow_serving.apis import predict_pb2
   from tensorflow_serving.apis import prediction_service_pb2_grpc
   from tensorflow.keras.applications.resnet50 import decode_predictions

   if __name__ == '__main__':
       channel = grpc.insecure_channel('localhost:8500')
       stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
       img_file = tf.keras.utils.get_file(
           "./kitten_small.jpg",
           "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg")
       img = image.load_img(img_file, target_size=(224, 224))
       img_array = preprocess_input(image.img_to_array(img)[None, ...])
       request = predict_pb2.PredictRequest()
       request.model_spec.name = 'resnet50_inf1'
       request.inputs['input'].CopyFrom(
           tf.make_tensor_proto(img_array, shape=img_array.shape))
       result = stub.Predict(request)
       prediction = tf.make_ndarray(result.outputs['output'])
       print(decode_predictions(prediction))
```

3. 运行脚本以将预测数据提交给服务。

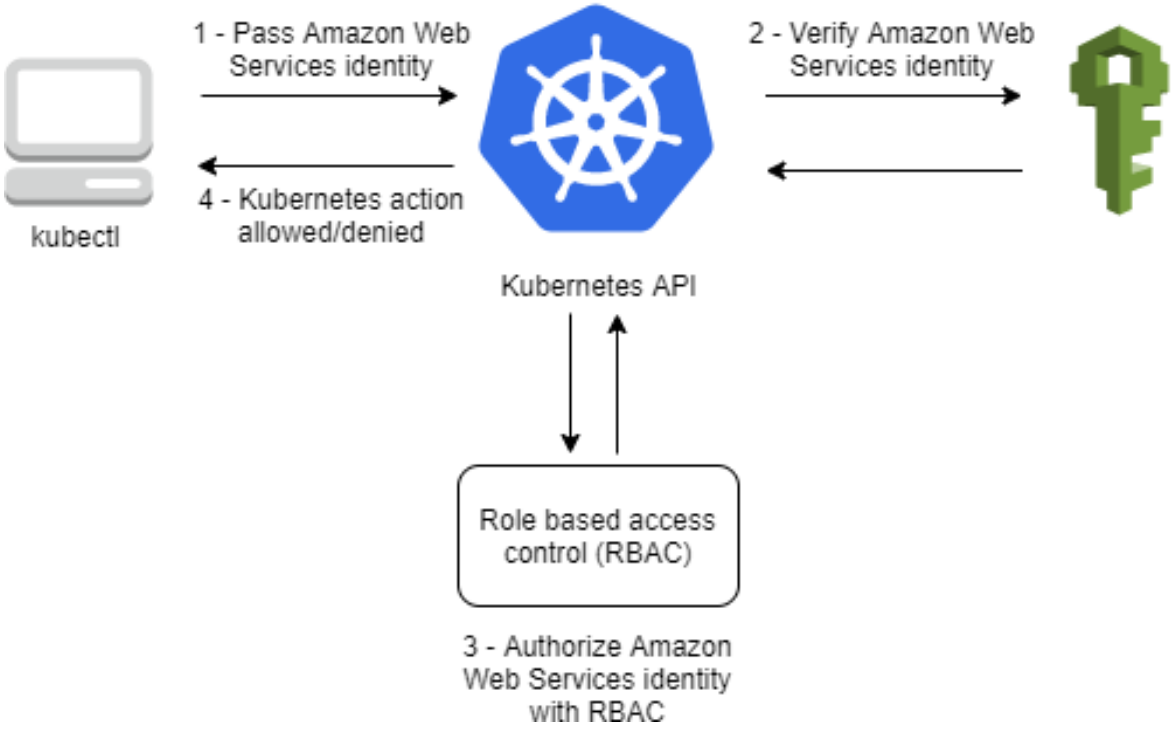
```
python3 tensorflow-model-server-infer.py
```

输出示例如下。

```
[[(u'n02123045', u'tabby', 0.68817204), (u'n02127052', u'lynx', 0.12701613),
 (u'n02123159', u'tiger_cat', 0.08736559), (u'n02124075', u'Egyptian_cat',
 0.063844085), (u'n02128757', u'snow_leopard', 0.009240591)]]
```

集群身份验证

Amazon EKS 使用 IAM 向您的 Kubernetes 集群提供身份验证（通过 Amazon CLI 的 `1.16.156` 版或更高版本中可用的 `aws eks get-token` 命令或者适用于 Kubernetes 的 Amazon IAM 身份验证），但它仍依赖于本机 Kubernetes 基于角色的访问控制 (RBAC, Role based access control) 来进行授权。这表示 IAM 仅用于有效 IAM 实体的身份验证。用于与 Amazon EKS 集群的 Kubernetes API 进行交互的所有权限均通过本机 Kubernetes RBAC 系统进行管理。下图显示了此关系。



Note

Amazon EKS 使用身份验证令牌来执行 `sts:GetCallerIdentity` 调用。因此，来自源 `sts.amazonaws.com` 且名字为 `GetCallerIdentity` 的事件 Amazon CloudTrail 从可以将 Amazon EKS 服务 IP 地址作为其源 IP 地址。

主题

- 让 IAM 用户和角色有权访问您的集群 (p. 355)
- 通过 OpenID Connect 身份提供商对集群的用户进行身份验证 (p. 362)
- 为 Amazon EKS 创建 kubeconfig (p. 365)
- 安装 aws-iam-authenticator (p. 369)
- 默认 Amazon EKS Kubernetes 角色和用户 (p. 373)

让 IAM 用户和角色有权访问您的集群

Amazon EKS 控制面板上运行的 Amazon IAM Authenticator for Kubernetes 支持使用 Amazon Identity and Access Management (IAM) 实体访问集群。身份验证程序从 `aws-auth ConfigMap` 获取配置信息。对于所有 `aws-auth ConfigMap` 设置，请参阅 GitHub 上的完整配置格式。

将 IAM 用户或角色添加到 Amazon EKS 集群

创建 Amazon EKS 集群时，将为 Amazon Identity and Access Management (IAM) 实体用户或角色，如创建集群的联合身份用户，自动授予 Amazon EKS 控制层面中基于集群角色的访问控制 (RBAC) 配置中的 `system:masters` 权限。此 IAM 实体不会显示在任何可见配置中，因此请确保跟踪最初创建集群的 IAM 实体。要授予其他 Amazon 用户或角色与集群进行交互的能力，必须编辑 Kubernetes 中的 `aws-auth ConfigMap`，创建 Kubernetes `rolebinding` 或 `clusterrolebinding`，名为 `aws-auth ConfigMap` 中指定的 `group`。

> **Note**
>
> 有关不同 IAM 实体的更多信息，请参阅 IAM 用户指南中的身份（用户、组和角色）。有关 Kubernetes 基于角色的访问控制 (RBAC, Role based access control) 配置的更多信息，请参阅使用 RBAC 授权。

将 IAM 用户或角色添加到 Amazon EKS 集群

1. 确定 `kubectl` 用来访问集群的凭据。可以在计算机上使用下面的命令查看 `kubectl` 使用的凭据。如果不使用原定设置路径，请将 ***~/.kube/config*** 替换为 `kubeconfig` 文件的路径。

```
cat ~/.kube/config
```

输出示例如下。

```
...
contexts:
- context:
    cluster: my-cluster.region-code.eksctl.io
    user: admin@my-cluster.region-code.eksctl.io
  name: admin@my-cluster.region-code.eksctl.io
current-context: admin@my-cluster.region-code.eksctl.io
...
```

在上一个示例输出中，为 *my-cluster* 集群配置名为 *admin* 的用户凭证。如果这是创建了集群的用户，那么该用户有权访问您的集群。如果不是创建集群的用户，则需要完成剩余步骤才能让其他用户有权访问集群。您可以使用以下命令查看哪些其他角色或用户当前有权访问您的集群：

```
kubectl describe -n kube-system configmap/aws-auth
```

输出示例如下。

```
Name:         aws-auth
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws:iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}


BinaryData
```

```
====

Events:  <none>
```

上一个示例是默认的 aws-auth ConfigMap。只有节点实例角色才有权访问集群。

2. 请确保您拥有现有的 Kubernetes roles 和 rolebindings 或 clusterroles 和 clusterrolebindings，您可以将 IAM 用户或角色映射到其中。有关这些资源的更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

   1. 查看您现有的 Kubernetes roles 或 clusterroles。Roles 范围限定为 namespace，但 clusterroles 范围则限定为集群。

      ```
      kubectl get roles -A
      ```

      ```
      kubectl get clusterroles
      ```

   2. 查看之前的输出中返回的任何 role 或 clusterrole 的详细信息，并确认它具有您希望 IAM 用户在集群中拥有的权限（rules）。

      将 *role-name* 替换为在上一个命令的输出中返回的 role 名称。将 *kube-system* 替换为 role 的命名空间。

      ```
      kubectl describe role role-name -n kube-system
      ```

      将 *cluster-role-name* 替换为在上一个命令的输出中返回的 clusterrole 名称。

      ```
      kubectl describe clusterrole cluster-role-name
      ```

   3. 查看您现有的 Kubernetes rolebindings 或 clusterrolebindings。Rolebindings 范围限定为 namespace，但 clusterrolebindings 范围则限定为集群。

      ```
      kubectl get rolebindings -A
      ```

      ```
      kubectl get clusterrolebindings
      ```

   4. 查看任何 rolebinding 或 clusterrolebinding 的详细信息，并确认它具有在上一步中列为 roleRef 的 role 或 clusterrole，以及为 subjects 列出的组名称。

      将 *role-binding-name* 替换为在上一个命令的输出中返回的 rolebinding 名称。将 *kube-system* 替换为 rolebinding 的 namespace。

      ```
      kubectl describe rolebinding role-binding-name -n kube-system
      ```

      输出示例如下。

      ```
      apiVersion: rbac.authorization.k8s.io/v1
      kind: RoleBinding
      metadata:
        name: eks-console-dashboard-restricted-access-role-binding
        namespace: default
      subjects:
      - kind: Group
        name: eks-console-dashboard-restricted-access-group
        apiGroup: rbac.authorization.k8s.io
      roleRef:
      ```

```
  kind: Role
  name: eks-console-dashboard-restricted-access-role
  apiGroup: rbac.authorization.k8s.io
```

将 *cluster-role-binding-name* 替换为在上一个命令的输出中返回的 clusterrolebinding 名称。

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

输出示例如下。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
- kind: Group
  name: eks-console-dashboard-full-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: eks-console-dashboard-full-access-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

3. 编辑 aws-auth ConfigMap。您可以使用 eksctl 之类的工具更新 ConfigMap，或者可以通过编辑它来进行手动更新。

   Important

   我们建议使用 eksctl 或者其他工具来编辑 ConfigMap。有关您可以使用的其他工具的信息，请参阅《Amazon EKS 最佳实践指南》中的使用工具对 aws-authConfigMap 进行更改。格式不正确的 aws-auth ConfigMap 可能会导致您失去对集群的访问权限。

   eksctl

   先决条件

   您的设备或 Amazon CloudShell 上安装了 0.110.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 安装或更新 eksctl (p. 10)。

   1. 查看 ConfigMap 中的当前映射。将 *my-cluster* 替换为您的集群名称。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

      ```
      eksctl get iamidentitymapping --cluster my-cluster --region=region-code
      ```

      输出示例如下。

      ```
      ARN
                    USERNAME                                GROUPS
            ACCOUNT
      arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
      NodeInstanceRole-1XLS7754U3ZPA     system:node:{{EC2PrivateDNSName}}
       system:bootstrappers,system:nodes
      ```

   2. 为角色添加映射。将 *my-role* 替换为您的角色名称。将 *eks-console-dashboard-full-access-group* 替换为您的 Kubernetes rolebinding 或 clusterrolebinding 中指定的组名称。请将 *111122223333* 替换为您的账户 ID。

      ```
      eksctl create iamidentitymapping \
      ```

```
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:role/my-role \
    --group eks-console-dashboard-full-access-group \
    --no-duplicate-arns
```

Important

角色 ARN 不能包含 role/my-team/developers/my-role 等路径。ARN 的格式必须为 arn:aws:iam::111122223333:role/my-role。在此示例中，my-team/developers/ 需要删除。

输出示例如下。

```
...
2022-05-09 14:51:20 [#]  adding identity "arn:aws:iam::111122223333:role/my-role"
 to auth ConfigMap
```

3. 为用户添加映射。将 *my-user* 替换为您的用户名。将 *eks-console-dashboard-restricted-access-group* 替换为您的 Kubernetes rolebinding 或 clusterrolebinding 中指定的组名称。请将 *111122223333* 替换为您的账户 ID。

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user \
    --group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

输出示例如下。

```
...
2022-05-09 14:53:48 [#]  adding identity "arn:aws:iam::111122223333:user/my-user"
 to auth ConfigMap
```

4. 再次查看 ConfigMap 中的映射。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

输出示例如下。

```
ARN
                USERNAME                                GROUPS
               ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:role/my-role
                                                        eks-console-dashboard-
full-access-group
arn:aws:iam::111122223333:user/my-user
                                                        eks-console-dashboard-
restricted-access-group
```

Edit ConfigMap manually

1. 打开 ConfigMap 文件进行编辑。

```
kubectl edit -n kube-system configmap/aws-auth
```

Note

如果您收到错误指示“Error from server (NotFound): configmaps "aws-auth" not found”，则使用 将 aws-authConfigMap 应用到集群 (p. 361) 中的过程应用库存 ConfigMap。

2. 将 IAM 用户、角色或 Amazon 账户添加到 ConfigMap。无法将 IAM 组添加到 ConfigMap。
   - 添加 IAM 角色（例如，对于联合身份用户）：将角色详细信息添加到 data 下 ConfigMap 的 mapRoles 部分。如果此部分在文件中尚不存在，请添加它。每个条目支持以下参数：
     - rolearn：要添加的 IAM 角色的 ARN。此值不能包含路径。例如，您无法指定 ARN，例如 arn:aws:iam::*111122223333*:role/my-team/developers/*role-name*。ARN 需要为 arn:aws:iam::*111122223333*:role/*role-name*。
     - username：Kubernetes 内要映射到 IAM 角色的用户名。
     - groups：要将角色映射到的组或 Kubernetes 组列表。该组可以是默认组，也可以是 clusterrolebinding 或 rolebinding 中指定的组。有关更多信息，请参阅 Kubernetes 文档中的默认角色和角色绑定。
   - 添加 IAM 用户：将用户详细信息添加到 data 下 ConfigMap 的 mapUsers 部分。如果此部分在文件中尚不存在，请添加它。每个条目支持以下参数：
     - userarn：要添加的 IAM 用户的 ARN。
     - username：Kubernetes 内要映射到 IAM 用户的用户名。
     - groups：要将用户映射到的组或 Kubernetes 组列表。该组可以是默认组，也可以是 clusterrolebinding 或 rolebinding 中指定的组。有关更多信息，请参阅 Kubernetes 文档中的默认角色和角色绑定。

   例如，下面的 YAML 块包含：
   - 一个 mapRoles 部分，此部分将 IAM 节点实例映射到 Kubernetes 组，以便节点可以自行注册到集群，和映射到可以查看所有集群的所有 Kubernetes 资源的 Kubernetes 组的 my-console-viewer-role IAM 角色。有关 my-console-viewer-role IAM 角色所需的 IAM 和 Kubernetes 组权限的列表，请参阅 所需权限 (p. 458)。
   - 一个 mapUsers 部分，此部分将默认 Amazon 账户中的 admin IAM 用户映射到 system:masters Kubernetes 组，以及映射到可以查看特定命名空间的 Kubernetes 资源的 Kubernetes 组的另一个 Amazon 账户的 my-user 用户。有关 my-user IAM 用户所需的 IAM 和 Kubernetes 组权限的列表，请参阅 所需权限 (p. 458)。

   根据需要添加或删除行并将所有 *example values* 替换为您自己的值。

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
 file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::111122223333:role/my-role
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
```

```
    - groups:
      - system:masters
      userarn: arn:aws:iam::111122223333:user/admin
      username: admin
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws:iam::444455556666:user/my-user
      username: my-user
```

3. 保存 文件并退出文本编辑器。

将 aws-authConfigMap 应用到集群

使用 eksctl 创建托管节点组时或创建节点组时自动创建 aws-auth ConfigMap 并应用于集群. 最初创建此 ConfigMap 的目的是允许节点加入您的集群，也可以使用 ConfigMap 为 IAM 用户和角色添加基于角色的访问控制 (RBAC)。如果您尚未启动自行管理节点并将 aws-auth ConfigMap 应用到集群，则可以按照下面的过程执行此操作。

将 **aws-authConfigMap** 应用到集群

1. 检查您是否已经应用 aws-auth ConfigMap。

```
kubectl describe configmap -n kube-system aws-auth
```

如果您收到错误指示“Error from server (NotFound): configmaps "aws-auth" not found”，则继续以下步骤应用库存 ConfigMap。

2. 下载、编辑和应用 Amazon 身份验证器配置映射。

   a. 下载配置映射。

```
curl -o aws-auth-cm.yaml https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/
cloudformation/2020-10-29/aws-auth-cm.yaml
```

   b. 使用文本编辑器打开 文件。将 *<ARN of instance role (not instance profile)>* 替换为与节点关联的 IAM 角色的 Amazon Resource Name (ARN)，然后保存相应文件。请勿修改此文件中的任何其他行。

      Important

      角色 ARN 不能包含 role/my-team/developers/my-role 等路径。ARN 的格式必须为 arn:aws:iam::*111122223333*:role/*my-role*。在此示例中，my-team/developers/ 需要删除。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

您可以检查节点组的 Amazon CloudFormation 堆栈输出，并查找以下值：

- InstanceRoleARN – 对于使用 `eksctl` 创建的节点组
- NodeInstanceRole – 对于在 Amazon Web Services Management Console 中使用 Amazon EKS 提供的 Amazon CloudFormation 模板创建的节点组

c. 应用配置。此命令可能需要几分钟才能完成。

```
kubectl apply -f aws-auth-cm.yaml
```

Note

如果您收到任何授权或资源类型错误，请参阅故障排除部分中的未经授权或访问被拒绝 (kubectl) (p. 478)。

3. 查看节点的状态并等待它们达到 `Ready` 状态。

```
kubectl get nodes --watch
```

通过 OpenID Connect 身份提供商对集群的用户进行身份验证

Amazon EKS 支持使用 OpenID Connect (OIDC) 身份提供商作为对您的集群的用户进行身份验证的方法。OIDC 身份提供商可与 Amazon Identity and Access Management (IAM) 一起使用或作为其替代方法。有关使用 IAM 的更多信息，请参阅让 IAM 用户和角色有权访问您的集群 (p. 355)。配置集群身份验证后，您可以创建 Kubernetes `roles` 和 `clusterroles` 以将权限分配给角色，然后使用 Kubernetes `rolebindings` 和 `clusterrolebindings` 将角色绑定到身份。有关更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

注意事项

- 您可以将一个 OIDC 身份提供商与您的集群关联。
- Kubernetes 没有提供 OIDC 身份提供商。您可以使用现有的公共 OIDC 身份提供商，也可以运行您自己的身份提供商。有关经认证提供商的列表，请参阅 OpenID 网站上的 OpenID 认证。
- OIDC 身份提供商的发布者 URL 必须可公开访问，以便 Amazon EKS 能够发现签名密钥。Amazon EKS 不支持拥有自签名证书的 OIDC 身份提供商。
- 您不能在集群上禁用 Amazon IAM 身份验证器，因为在将节点加入集群时仍需要使用它。有关更多信息，请参阅 GitHub 上的适用于 Kubernetes 的 Amazon IAM 身份验证器。
- Amazon EKS 集群仍必须由 Amazon IAM 用户创建，而不是由 OIDC 身份提供商用户创建。这是因为集群创建者与 Amazon EKS API 进行交互，而不是与 Kubernetes API 进行交互。
- 如果为控制层面启用了 CloudWatch Logs，则将在集群的审核日志中列出经 OIDC 身份提供商验证的用户。有关更多信息，请参阅启用和禁用控制层面日志 (p. 45)。
- 您无法使用来自 OIDC 提供商的账户登录 Amazon Web Services Management Console。您只能通过使用 Amazon Identity and Access Management 账户登录 Amazon Web Services Management Console，从而在控制台中查看 Kubernetes 资源 (p. 458)。

关联 OIDC 身份提供商

您需要提供商提供以下信息，然后才能将 OIDC 身份提供商与集群关联：

- 发布者 URL – OIDC 身份提供商的 URL，该 URL 允许 API 服务器发现用于验证令牌的公共签名密钥。该 URL 必须以 `https://` 开头，并应与提供商的 OIDC ID 令牌中的 `iss` 声明相对应。根据 OIDC 标准，允许使用路径组件，但不允许使用查询参数。通常，URL 只包含一个主机名称，如 `https://`

server.example.org 或 https://example.com。该 URL 应指向 .well-known/openid-configuration 以下的级别，并且必须可通过 Internet 网公开访问。

- 客户端 ID（也称为受众）– 向 OIDC 身份提供商发出身份验证请求的客户端应用程序的 ID。

您可以使用 eksctl 或 Amazon Web Services Management Console 关联身份提供商。

eksctl

使用 **eksctl** 将 OIDC 身份提供商与您的集群关联

1. 使用以下内容创建名为 *associate-identity-provider.yaml* 的文件。将 *example values* 替换为您自己的值。identityProviders 部分中的值是从 OIDC 身份提供商处获取的。仅 identityProviders 下的 name、type、issuerUrl 和 clientId 设置需要相应值。

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: your-region-code

identityProviders:
  - name: my-provider
    type: oidc
    issuerUrl: https://example.com
    clientId: kubernetes
    usernameClaim: email
    usernamePrefix: my-username-prefix
    groupsClaim: my-claim
    groupsPrefix: my-groups-prefix
    requiredClaims:
      string: string
    tags:
      env: dev
```

   Important

   不要为 groupsPrefix 或 usernamePrefix 指定 system: 或该字符串的任何部分。

2. 创建提供商。

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. 要使用 kubectl 与您的集群和 OIDC 身份提供商搭配使用，请参阅 Kubernetes 文档中的使用 kubectl。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 将 OIDC 身份提供商与您的集群关联

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择您的集群。
3. 选择 Configuration（配置）选项卡，然后选择 Authentication（身份验证）选项卡。
4. 在 OIDC Identity Providers（OIDC 身份提供商）页面上，选择 Associate Identity Provider（关联身份提供商）。
5. 在 Associate OIDC Identity Provider（关联 OIDC 身份提供商）页面上，输入或选择以下选项，然后选择 Associate（关联）。

- 对于 Name（名称），为提供商输入一个唯一的名称。
- 对于 Issuer URL（发布者 URL），输入您的提供商的 URL。此 URL 必须可通过 Internet 进行访问。
- 对于 Client ID（客户端 ID），输入 OIDC 身份提供商的客户端 ID（也称为受众）。
- 对于 Username claim（用户名声明），输入要用作用户名的声明。
- 对于 Groups claim（组声明），输入要用作用户的组的声明。
- （可选）选择 Advanced options（高级选项），输入或选择以下信息。
  - Username prefix（用户名前缀）– 输入要加在用户名声明前面的前缀。该前缀加在用户名声明前面，以防止与现有名称发生冲突。如果您未提供值，并且用户名为 `email` 之外的值，则前缀默认为 Issuer URL（发布者 URL）的值。您可以使用值 `-` 来禁用所有前缀。不要指定 `system:` 或该字符串的任何部分。
  - 组前缀 – 输入要加在组声明前面的前缀。该前缀加在组声明前面，以防止与现有名称（如 `system: groups`）发生冲突。例如，值 `oidc:` 会创建组名称，如 `oidc:engineering` 和 `oidc:infra`。不要指定 `system:` 或该字符串的任何部分。
  - Required claims（所需声明）– 选择 Add claim（添加声明），然后在客户端 ID 令牌中输入描述所需声明的一个或多个键值对。这些键值对描述了 ID 令牌中所需的声明。如果设置，则验证每个声明是否存在于具有匹配值的 ID 令牌中。

6. 要使用 `kubectl` 与您的集群和 OIDC 身份提供商搭配使用，请参阅 `kubectl` 文档中的使用 Kubernetes。

解除 OIDC 身份提供商与集群的关联

如果您解除 OIDC 身份提供商与集群的关联，则提供商中包含的用户将无法再访问该集群。但是，您仍然可以使用 Amazon IAM 用户访问该集群。

使用 Amazon Web Services Management Console 解除 OIDC 身份提供商与集群的关联

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 在 OIDC Identity Providers（OIDC 身份提供商）部分中，选择 Disassociate（解除关联），输入身份提供商名称，然后选择 `Disassociate`。

示例 IAM policy

如果要阻止 OIDC 身份提供商与集群关联，请创建以下 IAM policy 并将其与 Amazon EKS 管理员的 IAM 账户关联。有关更多信息，请参阅 IAM 用户指南中的创建 IAM policy 和添加 IAM 身份权限，以及服务授权参考中的 Amazon Elastic Kubernetes Service 的操作、资源和条件键。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "denyOIDC",
            "Effect": "Deny",
            "Action": [
                "eks:AssociateIdentityProviderConfig"
            ],
            "Resource": "arn:aws:eks:cn-north-1.amazonaws.com.cn:111122223333:cluster/*"

        },
        {
            "Sid": "eksAdmin",
            "Effect": "Allow",
            "Action": [
```

```
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

如果 clientID 为 kubernetes 而 issuerUrl 为 https://cognito-idp.cn-north-1amazonaws.com.cn/*，则以下示例策略允许进行 OIDC 身份提供商关联。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCognitoOnly",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:cn-north-1:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotLikeIfExists": {
                    "eks:issuerUrl": "https://cognito-idp.cn-north-1.amazonaws.com.cn/*"
                }
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:cn-north-1:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
                    "eks:clientId": "kubernetes"
                }
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}
```

为 Amazon EKS 创建 kubeconfig

在本节中，将为集群创建一个 kubeconfig 文件（或更新现有文件）。

本节提供两个用于创建或更新 *kubeconfig* 文件的过程。您可以使用 Amazon CLI，通过 Amazon CLI update-kubeconfig 命令快速自动创建或更新 kubeconfig 文件，或者您可以使用 Amazon CLI 或 aws-iam-authenticator 手动创建 kubeconfig 文件。

Amazon EKS 使用 Amazon CLI 的版本 1.16.156 或更高版本中提供的 aws eks get-token 命令或者适用于 Kubernetes 的 Amazon IAM 身份验证器借助 kubectl 进行集群身份验证。如果您的系统上已经安装了 Amazon CLI，则默认情况下，适用于 Kubernetes 的 Amazon IAM 身份验证器使用以下命令返回的相同凭证：

```
aws sts get-caller-identity
```

有关更多信息，请参阅《Amazon Command Line Interface 用户指南》中的配置 Amazon CLI。

自动创建 kubeconfig 文件

使用 Amazon CLI 创建 **kubeconfig**

1. 确保您已安装版本 1.25.46 或 2.7.21 或更高版本。要安装或升级 Amazon CLI，请参阅 Amazon Command Line Interface 用户指南中的安装 Amazon CLI。

   Note

   系统的 Python 版本必须为 2.7.9 或更高版本。否则，在 Amazon CLI 调用 Amazon EKS 时会收到 hostname doesn't match 错误。

   可以使用以下命令来查看您的 Amazon CLI 版本：

   ```
   aws --version
   ```

   Important

   软件包管理器（例如 yum、apt-get 或适用于 macOS 的 Homebrew）通常比 Amazon CLI 落后几个版本。要确保您使用的是最新版本，请参阅 Amazon Command Line Interface 用户指南中的安装 Amazon CLI。

2. 为集群创建或更新 kubeconfig 文件。将 *example values* 替换为您自己的值。

   - 预设情况下，在主目录的原定设置 kubeconfig 路径 (.kube) 中创建得到的配置文件，或者或与该位置的现有 config 合并。您可以使用 **--kubeconfig** 选项指定其他路径。
   - 发出 kubectl 命令时，可以使用 **--role-arn** 选项指定 IAM 角色 ARN 供身份验证使用。否则，将使用默认 Amazon CLI 或开发工具包凭证链中的 IAM 实体。通过运行 aws sts get-caller-identity 命令可以查看默认 Amazon CLI 或开发工具包标识。
   - 有关更多信息，请使用 aws eks update-kubeconfig help 命令参阅帮助页面，或参阅 Amazon CLI 命令参考中的 update-kubeconfig。

   Note

   要运行以下命令，您必须具有对指定的集群使用 eks:DescribeCluster API 操作的权限。有关更多信息，请参阅Amazon EKS 基于身份的策略示例 (p. 415)。

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

3. 测试配置。

   ```
   kubectl get svc
   ```

   Note

   如果您收到任何授权或资源类型错误，请参阅故障排除部分中的未经授权或访问被拒绝 (kubectl) (p. 478)。

   输出示例如下。

   ```
   NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
   ```

手动创建 kubeconfig 文件

手动创建 **kubeconfig** 文件

1. 通过将 *example values* 替换为您自己的值，然后运行修改后的命令来设置几个变量的值。

```
export region_code=region-code
export cluster_name=my-cluster
export account_id=111122223333
```

2. 检索集群的端点并将值存储在变量中。

```
cluster_endpoint=$(aws eks describe-cluster \
    --region $region_code \
    --name $cluster_name \
    --query "cluster.endpoint" \
    --output text)
```

3. 检索与集群通信所需的 Base64 编码证书数据，并将值存储在变量中。

```
certificate_data=$(aws eks describe-cluster \
    --region $region_code \
    --name $cluster_name \
    --query "cluster.certificateAuthority.data" \
    --output text)
```

4. 如果默认 `~/.kube` 目录尚不存在，请创建它。

```
mkdir -p ~/.kube
```

5. 为您的首选客户端令牌方法运行命令（Amazon CLI 或适用于 Kubernetes 的 Amazon IAM 身份验证器）以在 `~/.kube` 目录中创建 `config` 文件。在运行其中一个命令之前，可以通过修改命令以包含以下选项来指定以下内容：

   - IAM 角色 – 删除 `args:` 下的行开头的 `#`。将 *my-role* 替换为您想要使用其执行集群操作的 IAM 角色的名称而不是默认的 Amazon 凭证提供程序链。有关更多信息，请参阅 GitHub 上的设置 `kubectl` 以使用适用于 Kubernetes 的 Amazon IAM 身份验证器提供的身份验证令牌。
   - Amazon CLI 命名配置文件 – 删除 `env:` 行开头的 `#`，然后删除其下的行开头的 `#`。将 *aws-profile* 替换为要使用的配置文件的名称。如果未指定配置文件，系统会使用默认配置文件。有关其他信息，请参阅 GitHub 上的指定凭证并使用 Amazon 配置文件。

   Amazon CLI

   先决条件

   必须在您的设备上安装版本 `1.16.156` 或更高版本的 Amazon CLI。

```
#!/bin/bash
read -r -d '' KUBECONFIG <<EOF
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: $certificate_data
    server: $cluster_endpoint
  name: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
contexts:
- context:
    cluster: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
```

```
    user: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
  name: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
current-context: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
kind: Config
preferences: {}
users:
- name: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      command: aws
      args:
        - --region
        - $region_code
        - eks
        - get-token
        - --cluster-name
        - $cluster_name
        # - "- --role-arn"
        # - "arn:aws:iam::$account_id:role/my-role"
      # env:
        # - name: "Amazon_PROFILE"
        #   value: "aws-profile"
EOF
echo "${KUBECONFIG}" > ~/.kube/config
```

Amazon IAM Authenticator for Kubernetes

先决条件

必须在您的设备上安装适用于 Kubernetes 的 Amazon IAM 身份验证器。要进行安装，请参阅 安装 aws-iam-authenticator (p. 369)。

```
#!/bin/bash
read -r -d '' KUBECONFIG <<EOF
apiVersion: v1
clusters:
- cluster:
    server: $cluster_endpoint
    certificate-authority-data: $certificate_data
  name: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
contexts:
- context:
    cluster: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
    user: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
  name: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
current-context: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
kind: Config
preferences: {}
users:
- name: arn:aws:eks:$region_code:$account_id:cluster/$cluster_name
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1beta1
      command: aws-iam-authenticator
      args:
        - "token"
        - "-i"
        - "$cluster_name"
        # - "- --role-arn"
        # - "arn:aws:iam::$account_id:role/my-role"
      # env:
        # - name: "Amazon_PROFILE"
        #   value: "aws-profile"
```

```
EOF
echo "${KUBECONFIG}" > ~/.kube/config
```

6. 将该文件路径添加到 KUBECONFIG 环境变量，以便 kubectl 了解在哪里查找集群配置。

   - 对于 macOS 或 Linux 上的 Bash shell：

   ```
   export KUBECONFIG=$KUBECONFIG:~/.kube/config
   ```

   - 对于 Windows 上的 PowerShell：

   ```
   $ENV:KUBECONFIG="{0};{1}" -f  $ENV:KUBECONFIG, "$ENV:userprofile\.kube\config"
   ```

7. (可选) 将配置添加到您的 shell 初始化文件，以便在打开 shell 时对其进行配置。

   - 对于 macOS 上的 Bash shell：

   ```
   echo 'export KUBECONFIG=$KUBECONFIG:~/.kube/config' >> ~/.bash_profile
   ```

   - 对于 Linux 上的 Bash shell：

   ```
   echo 'export KUBECONFIG=$KUBECONFIG:~/.kube/config' >> ~/.bashrc
   ```

   - 对于 Windows 上的 PowerShell：

   ```
   [System.Environment]::SetEnvironmentVariable('KUBECONFIG', $ENV:KUBECONFIG,
    'Machine')
   ```

8. 测试配置。

   ```
   kubectl get svc
   ```

   Note

   如果您收到任何授权或资源类型错误，请参阅故障排除部分中的未经授权或访问被拒绝 (kubectl) (p. 478)。

   输出示例如下。

   ```
   NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
   ```

安装 aws-iam-authenticator

Amazon EKS 使用 IAM 向您的 Kubernetes 集群提供身份验证（通过适用于 Kubernetes 的 Amazon IAM 身份验证器）。您可以将库存 kubectl 客户端配置为与 Amazon EKS 搭配使用，方法为：安装适用于 Kubernetes 的 Amazon IAM 身份验证器，然后修改 kubectl 配置文件 (p. 365)，从而用其进行身份验证。

Note

如果您正在运行 Amazon CLI 版本 1.16.156 或更高版本，则无需安装身份验证器。相反，您可以使用 aws eks get-token 命令。有关更多信息，请参阅手动创建 kubeconfig 文件 (p. 367)。

如果您无法使用 Amazon CLI 版本 1.16.156 或更高版本创建 kubeconfig 文件，则您可以在 macOS、Linux 或 Windows 上安装适用于 Kubernetes 的 Amazon IAM 身份验证器。

macOS

使用 Homebrew 安装 **aws-iam-authenticator**

安装 aws-iam-authenticator 的最简单方法是使用 Homebrew。

1. 如果您的 Mac 上尚未安装 Homebrew，请使用以下命令安装它。

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
master/install.sh)"
```

2. 使用以下命令安装 aws-iam-authenticator。

```
brew install aws-iam-authenticator
```

3. 测试 aws-iam-authenticator 二进制文件的工作原理。

```
aws-iam-authenticator help
```

在 macOS 上安装 **aws-iam-authenticator**

也可以按照以下步骤安装 Amazon 提供的 aws-iam-authenticator 版本。

1. 从 Amazon S3 中下载 Amazon EKS 提供的 aws-iam-authenticator 二进制文件。

```
curl -o aws-iam-authenticator https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.21.2/2021-07-05/bin/darwin/amd64/aws-iam-authenticator
```

2. （可选）使用同一存储桶前缀中提供的 SHA-256 总和验证下载的二进制文件。
   a. 下载系统的 SHA-256 总和。

```
curl -o aws-iam-authenticator.sha256 https://s3.cn-north-1.amazonaws.com.cn/
amazon-eks/1.21.2/2021-07-05/bin/darwin/amd64/aws-iam-authenticator.sha256
```

   b. 检查下载的二进制文件的 SHA-256 总和。

```
openssl sha1 -sha256 aws-iam-authenticator
```

   c. 将命令输出中生成的 SHA-256 总和与下载的 aws-iam-authenticator.sha256 文件进行比较。这两者应该匹配。
3. 将执行权限应用于二进制文件。

```
chmod +x ./aws-iam-authenticator
```

4. 将二进制文件复制到您的 $PATH 中的文件夹。我们建议创建一个 $HOME/bin/aws-iam-authenticator 并确保 $HOME/bin 首先显示在您的 $PATH 中。

```
mkdir -p $HOME/bin && cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator &&
 export PATH=$PATH:$HOME/bin
```

5. 将 $HOME/bin 添加到 PATH 环境变量。

```
echo 'export PATH=$PATH:$HOME/bin' >> ~/.bash_profile
```

6. 测试 aws-iam-authenticator 二进制文件的工作原理。

```
aws-iam-authenticator help
```

Linux

在 Linux 上安装 **aws-iam-authenticator**

1. 从 Amazon S3 中下载 Amazon EKS 为您的硬件平台提供的 aws-iam-authenticator 二进制文件。

```
curl -o aws-iam-authenticator https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.21.2/2021-07-05/bin/linux/amd64/aws-iam-authenticator
```

```
curl -o aws-iam-authenticator https://s3.cn-north-1.amazonaws.com.cn/amazon-
eks/1.21.2/2021-07-05/bin/linux/arm64/aws-iam-authenticator
```

2. （可选）使用您的硬件平台的同一存储桶前缀中提供的 SHA-256 总和验证下载的二进制文件。

   a. 下载系统的 SHA-256 总和。

```
curl -o aws-iam-authenticator.sha256 https://s3.cn-north-1.amazonaws.com.cn/
amazon-eks/1.21.2/2021-07-05/bin/linux/amd64/aws-iam-authenticator.sha256
```

```
curl -o aws-iam-authenticator.sha256 https://s3.cn-north-1.amazonaws.com.cn/
amazon-eks/1.21.2/2021-07-05/bin/linux/arm64/aws-iam-authenticator.sha256
```

   b. 检查下载的二进制文件的 SHA-256 总和。

```
openssl sha1 -sha256 aws-iam-authenticator
```

   c. 将命令输出中生成的 SHA-256 总和与下载的 aws-iam-authenticator.sha256 文件进行比较。这两者应该匹配。

3. 将执行权限应用于二进制文件。

```
chmod +x ./aws-iam-authenticator
```

4. 将二进制文件复制到您的 $PATH 中的文件夹。我们建议创建一个 $HOME/bin/aws-iam-authenticator 并确保 $HOME/bin 首先显示在您的 $PATH 中。

```
mkdir -p $HOME/bin && cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator &&
 export PATH=$PATH:$HOME/bin
```

5. 将 $HOME/bin 添加到 PATH 环境变量。

```
echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
```

6. 测试 aws-iam-authenticator 二进制文件的工作原理。

```
aws-iam-authenticator help
```

Windows

要在安装有 Chocolatey 的 Windows 上安装 **aws-iam-authenticator**

1. 如果您尚未在 Windows 系统上安装 Chocolatey，请参阅安装 Chocolatey。
2. 打开 PowerShell 终端窗口并使用以下命令安装 aws-iam-authenticator 程序包：

```
choco install -y aws-iam-authenticator
```

3. 测试 aws-iam-authenticator 二进制文件的工作原理。

```
aws-iam-authenticator help
```

在 Windows 上安装 **aws-iam-authenticator**

1. 打开一个 PowerShell 终端窗口并从 Amazon S3 下载 Amazon EKS 提供的 aws-iam-authenticator 二进制文件。

```
curl -o aws-iam-authenticator.exe https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.21.2/2021-07-05/bin/windows/amd64/aws-iam-authenticator.exe
```

2. （可选）使用同一存储桶前缀中提供的 SHA-256 总和验证下载的二进制文件。
   a. 下载系统的 SHA-256 总和。

```
curl -o aws-iam-authenticator.sha256 https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/1.21.2/2021-07-05/bin/windows/amd64/aws-iam-authenticator.exe.sha256
```

   b. 检查下载的二进制文件的 SHA-256 总和。

```
Get-FileHash aws-iam-authenticator.exe
```

   c. 将命令输出中生成的 SHA-256 总和与下载的 SHA-256 文件进行比较。二者应匹配，不过 PowerShell 输出是大写。
3. 将二进制文件复制到您的 PATH 中的文件夹。如果 PATH 中有现有目录可用于命令行实用程序，请将二进制文件复制到该目录。否则，请完成以下步骤。
   a. 为命令行二进制文件创建一个新目录，例如 C:\bin。
   b. 将 aws-iam-authenticator.exe 二进制文件复制到新目录。
   c. 编辑用户或系统 PATH 环境变量，将新目录添加到 PATH 中。
   d. 关闭 PowerShell 终端并打开一个新的来选取新的 PATH 变量。
4. 测试 aws-iam-authenticator 二进制文件的工作原理。

```
aws-iam-authenticator help
```

如果您现有一个 Amazon EKS 集群，则为该集群创建一个 kubeconfig 文件。有关更多信息，请参阅为 Amazon EKS 创建 kubeconfig (p. 365)。否则，请参阅 创建 Amazon EKS 集群 (p. 21) 以创建新的 Amazon EKS 集群。

默认 Amazon EKS Kubernetes 角色和用户

在 1.20 和更高版本的集群中，Amazon EKS 会为每个 Kubernetes 组件创建身份。Amazon EKS 团队已创建以下对象来为组件提供 RBAC 权限：

角色

- `eks:authenticator`
- `eks:certificate-controller-approver`
- `eks:certificate-controller`
- `eks:cluster-event-watcher`
- `eks:fargate-scheduler`
- `eks:k8s-metrics`
- `eks:nodewatcher`
- `eks:pod-identity-mutating-webhook`

角色绑定

- `eks:authenticator`
- `eks:certificate-controller-approver`
- `eks:certificate-controller`
- `eks:cluster-event-watcher`
- `eks:fargate-scheduler`
- `eks:k8s-metrics`
- `eks:nodewatcher`
- `eks:pod-identity-mutating-webhook`

用户

- `eks:authenticator`
- `eks:certificate-controller`
- `eks:cluster-event-watcher`
- `eks:fargate-scheduler`
- `eks:k8s-metrics`
- `eks:nodewatcher`
- `eks:pod-identity-mutating-webhook`

除了这些对象之外，Amazon EKS 还在集群引导过程中对 `kubectl` 操作使用特殊的用户身份 `eks:cluster-bootstrap`。Amazon EKS 还对集群管理操作使用特殊的用户身份 `eks:support-engineer`。所有用户身份都会出现在 `kube` 审计日志中，可通过 CloudWatch 提供给客户。

运行 **`kubectl describe clusterrole`** ***`rolename`*** 以查看每个角色的权限。

集群管理

本章包含可以帮助您管理集群的以下主题。您还可以使用 Amazon Web Services Management Console 查看有关 Kubernetes 资源 (p. 458)的信息。

- 教程：部署 Kubernetes 控制面板 (Web UI) (p. 374) – 了解如何安装控制面板，一个适用于 Kubernetes 集群和应用程序的基于 Web 的用户界面。
- 安装 Kubernetes Metrics Server (p. 380) – Kubernetes Metrics Server 是集群中资源使用情况数据的聚合器。默认情况下，它不会部署在集群中，而是由 Kubernetes 附加组件（例如 Kubernetes 控制面板和 Horizontal Pod Autoscaler (p. 325)）使用。在本主题中，您将了解如何安装 Metrics Server。
- Prometheus 的控制层面指标 (p. 381) – Kubernetes API 服务器公开了大量可用于监控和分析的指标。本主题介绍如何部署 Prometheus，以及您可以用来查看和分析集群操作的方法。
- 将 Helm 与 Amazon EKS 结合使用 (p. 384) – Kubernetes 的 Helm 程序包管理器帮助您在 Kubernetes 集群上安装和管理应用程序。本主题可帮助您安装并运行 Helm 二进制文件，以便您可以在本地计算机中使用 Helm CLI 安装和管理图表。
- 为您的 Amazon EKS 资源添加标签 (p. 385) – 为了帮助您管理 Amazon EKS 资源，您可以以标签的形式为每个资源分配您自己的元数据。本主题介绍标签并演示如何创建标签。
- Amazon EKS 服务配额 (p. 389) – 您的Amazon账户对于每项Amazon服务都有默认配额（以前称为限制）。了解 Amazon EKS 的配额以及如何增加这些配额。

教程：部署 Kubernetes 控制面板 (Web UI)

本教程将指导您完成将 Kubernetes 控制面板部署到 Amazon EKS 集群的过程，包括 CPU 和内存指标。它还帮助您创建可用于安全地连接到控制面板以查看和控制集群的 Amazon EKS 管理员服务账户。如果您在使用控制面板时遇到问题，可以在项目的 GitHub 存储库中创建签发或拉取请求。

先决条件

本教程假定：

- 您已通过执行 开始使用 Amazon EKS (p. 4) 中的步骤创建了一个 Amazon EKS 集群。
- 您已安装 Kubernetes Metrics Server。有关更多信息，请参阅安装 Kubernetes Metrics Server (p. 380)。
- 控制层面弹性网络接口和节点的安全组采用 Amazon EKS 安全组要求和注意事项 (p. 231) 中的推荐设置。
- 您使用的是配置为与 Amazon EKS 集群通信 (p. 17)的 `kubectl` 客户端。

步骤 1：部署 Kubernetes 控制面板

下载、修改控制面板清单并将其应用到您的集群。

1. 使用针对集群版本的命令将控制面板清单文件下载到您的设备。
   - 版本 1.22

     可用版本的某些功能可能无法在此 Kubernetes 版本上正常使用。有关更多信息，请参阅 GitHub 上的版本。
   - 版本 1.20 和 1.21

```
curl -o recommended.yaml https://raw.githubusercontent.com/kubernetes/dashboard/v2.4.0/aio/deploy/recommended.yaml
```

- 版本 1.19

```
curl -o recommended.yaml https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.5/aio/deploy/recommended.yaml
```

2. 修改清单文件。
   a. 识别清单中的容器映像。

```
cat recommended.yaml | grep image:
```

   输出示例如下。

```
image: kubernetesui/dashboard:v2.4.0
image: kubernetesui/metrics-scraper:v1.0.7
```

   b. 提取上一步中返回的映像，并将其推送到节点有权访问的存储库。有关如何提取、标记和推送镜像到您自己的存储库的更多信息，请参阅 将容器镜像从一个存储库复制到另一个存储库 (p. 338)。
   c. 将以下命令中的 *your-registry* 替换为您的注册表，然后运行修改后的命令，将文件中的 kubernetesui 替换为您的注册表。

```
sed -i.bak -e 's|kubernetesui|your-registry|' recommended.yaml
```

   d. 将以下命令中的 *your-repository* 和 *your-tag* 替换为您的存储库和标签，然后运行修改后的命令，将文件中的 *dashboard*:v*2.4.0* 替换为您的注册表和标签。

```
sed -i.bak -e 's|dashboard:v2.4.0|your-repository:your-tag|' recommended.yaml
```

   e. 将以下命令中的 *your-repository* 和 *your-tag* 替换为您的存储库和标签，然后运行修改后的命令，将文件中的 *dashboard*:*v2.4.0* 替换为您的注册表和标签。

```
sed -i.bak -e 's|metrics-scraper:v1.0.7|your-repository:your-tag|' recommended.yaml
```

3. 将 Kubernetes 控制面板清单应用于集群。

```
kubectl apply -f recommended.yaml
```

   输出示例如下。

```
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

第 2 步：创建 eks-admin 服务账户和集群角色绑定

默认情况下，Kubernetes 控制面板用户的权限是有限的。在此部分中，您创建一个可用于使用管理员级别权限安全地连接到控制面板的 eks-admin 服务账户和集群角色绑定。有关更多信息，请参阅 Kubernetes 文档中的管理服务账户。

创建 **eks-admin** 服务账户和集群角色绑定

> Important
>
> 使用此过程创建的示例服务账户在集群上具有完整的 cluster-admin (超级用户) 特权。有关更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

1. 运行以下命令以创建包含以下文本的名为 eks-admin-service-account.yaml 的文件。此清单定义一个名为 eks-admin 的服务账户和集群角色绑定。

```
cat >eks-admin-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eks-admin
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: eks-admin
  namespace: kube-system
EOF
```

2. 将此服务账户和集群角色绑定应用到您的集群。

```
kubectl apply -f eks-admin-service-account.yaml
```

输出示例如下。

```
serviceaccount "eks-admin" created
clusterrolebinding.rbac.authorization.k8s.io "eks-admin" created
```

第 3 步：连接到控制面板

现在，已将 Kubernetes 控制面板部署到集群，并且您已具有可用于查看和控制集群的管理员服务账户，您可使用该服务账户连接到控制面板。

连接到 Kubernetes 控制面板

1. 检索 eks-admin 服务账户的身份验证令牌。

```
kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep eks-admin | awk '{print $1}')
```

输出示例如下。

```
Name:         eks-admin-token-b5zv4
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name=eks-admin
              kubernetes.io/service-account.uid=bcfe66ac-39be-11e8-97e8-026dce96b6e8

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1025 bytes
namespace:  11 bytes
token:      authentication-token
```

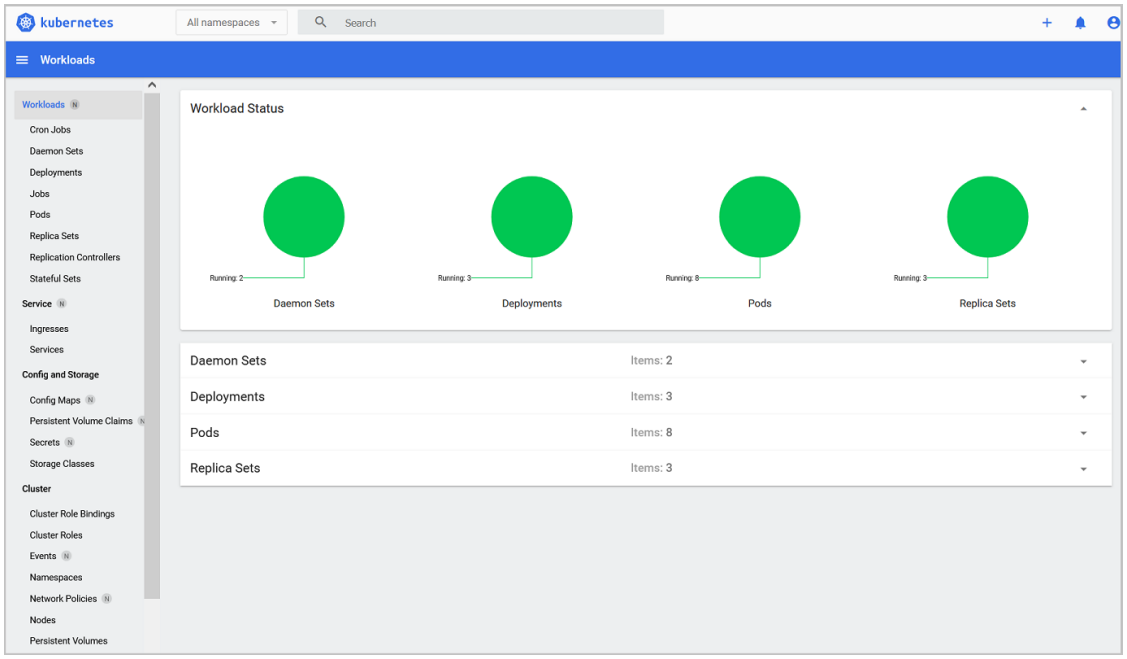
从输出中复制 *authentication-token* 值。您可以在稍后的步骤中使用此令牌连接到控制面板。

2. 启动 `kubectl proxy`。

```
kubectl proxy
```

3. 要访问控制面板终端节点，请使用 Web 浏览器打开以下链接：http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#!/login。
4. 选择 Token（令牌），将上一个命令中的 *authentication-token* 输出粘贴到 Token（令牌）字段中，然后选择 SIGN IN（登录）。

登录之后，您会在 Web 浏览器中看到控制面板。



有关使用控制面板的更多信息，请参阅 Kubernetes 文档中的部署和访问 Kubernetes 控制面板。

成本监控

Amazon EKS 支持 Kubecost，您可以使用该功能监控按 Kubernetes 资源（包括 pods、节点、命名空间和标签）细分的成本。作为 Kubernetes 平台管理员和财务主管，您可以使用 Kubecost 可视化 Amazon EKS 费用明细、分配成本以及向应用程序团队等组织部门退款。您可以根据内部团队和业务部门的实际 Amazon 账单为其提供透明、准确的成本数据。此外，您还可以根据他们的基础设施环境及其集群内的使用模式获得定制的成本优化建议。有关 Kubecost 的详细信息，请参阅 Kubecost 文档。

Amazon EKS 提供 Amazon 优化的 Kubecost 捆绑包，以实现集群成本可视化。您可以使用现有的 Amazon 支持协议获取支持。

先决条件

- 现有 版 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS (p. 4)。
- 您的设备或 Amazon CloudShell 上安装了 `kubectl` 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 `1.22`，则可以将 `kubectl` 版本 `1.21`、`1.22` 或 `1.23` 用于它。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl` (p. 4)。
- 您的设备或 Amazon CloudShell 上配置了 3.9.0 版或更高版本的 Helm。要安装或更新 Helm，请参阅 the section called “使用 Helm” (p. 384)。
- 如果您的集群是 `1.23` 版或更高版本，您必须在集群上安装 the section called “Amazon EBS CSI 驱动程序” (p. 195)。

> **Note**
>
> 您无法在 Fargate 节点上运行 Kubecost。

安装 Kubecost

1. 使用以下命令安装 Kubecost。您可以将 *`1.96.0`* 替换为更高版本。您可以在 Amazon ECR Public Gallery 中的 kubecost/cost-analyzer 上查看可用的版本。

```
helm upgrade -i kubecost oci://public.ecr.aws/kubecost/cost-analyzer --version 1.96.0 \
    --namespace kubecost --create-namespace \
    -f https://raw.githubusercontent.com/kubecost/cost-analyzer-helm-chart/develop/
cost-analyzer/values-eks-cost-monitoring.yaml
```

   Kubecost 定期发布新版本。您可以使用 helm upgrade 更新您的版本。安装包括本地 Prometheus 服务器和 `kube-state-metrics`。如果您的环境已满足某些先决条件，则可以自定义您的部署。有关更多信息，请参阅 GitHub 上的示例配置文件。

2. 确保所需的容器组（pod）正在运行。

```
kubectl get pods -n kubecost
```

   输出示例如下。

```
NAME                                          READY   STATUS    RESTARTS   AGE
kubecost-cost-analyzer-b9788c99f-5vj5b        2/2     Running   0          3h27m
kubecost-kube-state-metrics-99bb8c55b-bn2br   1/1     Running   0          3h27m
kubecost-prometheus-server-7d9967bfc8-9c8p7   2/2     Running   0          3h27m
```

3. 在您的设备上，启用端口转发以公开 Kubecost 控制面板。

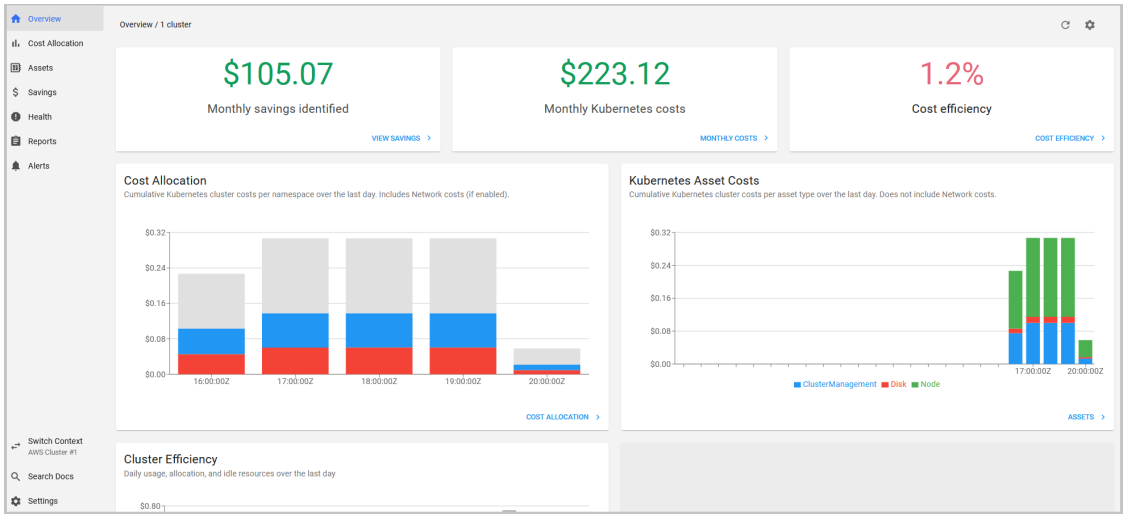
```
kubectl port-forward --namespace kubecost deployment/kubecost-cost-analyzer 9090
```

或者，您可以使用 Amazon Load Balancer Controller (p. 287) 公开 Kubecost 并使用 Amazon Cognito 用于身份验证、授权和用户管理。有关更多信息，请参阅如何使用应用程序负载均衡器和 Amazon Cognito 为您的 Kubernetes Web 应用程序用户进行身份验证。

4. 在完成上一步的同一台设备上，打开 Web 浏览器并输入以下地址。

```
http://localhost:9090
```

浏览器中将显示 Kubecost 概述页面。Kubecost 收集指标可能需要 5-10 分钟。您可以查看 Amazon EKS 支出，包括累计的集群成本、关联的 Kubernetes 资产成本和每月聚合支出。



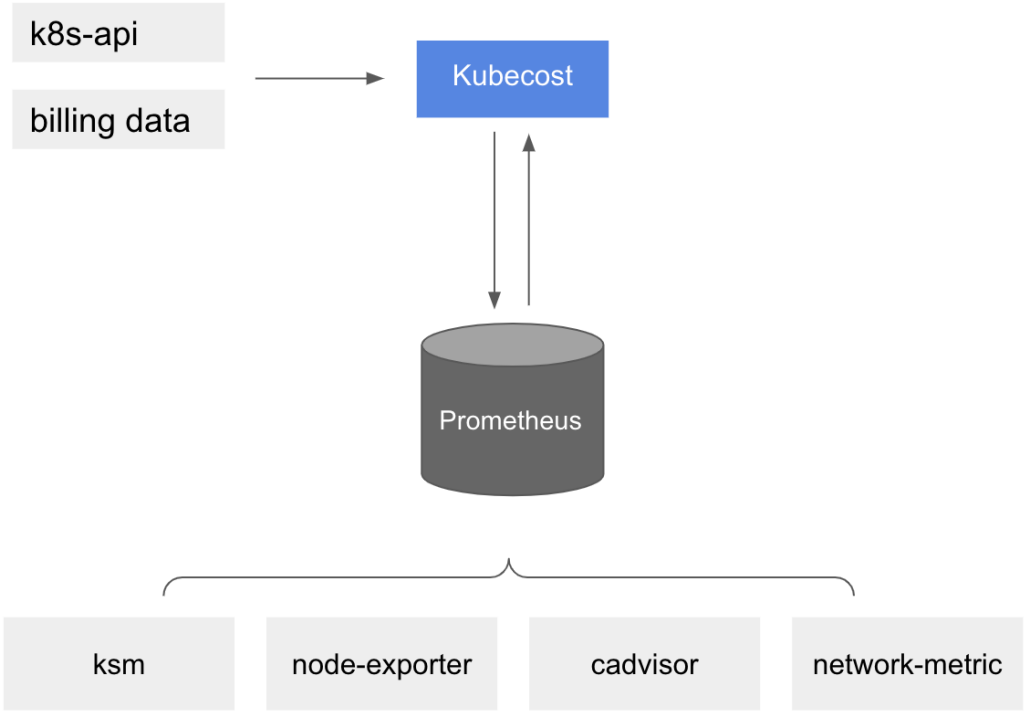
5. 要跟踪集群级别的成本，请标记您的 Amazon EKS 资源以进行计费。有关更多信息，请参阅the section called “标记资源以便于计费” (p. 387)。

在控制面板的左侧窗格中，您还可以选择以下信息进行查看：

- 成本分配 – 查看过去七天内每个命名空间和其他维度的 Amazon EKS 月度成本和累计成本。这有助于了解应用程序的哪些部分产生 Amazon EKS 支出。
- 资产 – 查看与您的 Amazon EKS 资源关联的 Amazon 基础设施资产的成本。

其他功能

- 导出成本指标 – Amazon EKS 优化的成本监控使用 Kubecost 和 Prometheus 部署，这是一个开源监控系统和时间序列数据库。Kubecost 从 Prometheus 读取指标，然后执行成本分配计算并将指标写回 Prometheus。Kubecost 前端从 Prometheus 中读取指标并在 Kubecost 用户界面上显示这些指标。此架构如下图所示。



预装 Prometheus 后，您可以编写查询，以将 Kubecost 数据提取到当前的商业智能系统，进行进一步分析。您也可以将其用作当前 Grafana 控制面板的数据来源，以显示您的内部团队熟悉的 Amazon EKS 集群成本。要了解有关如何编写 Prometheus 查询的更多信息，请参阅 GitHub 上的 Prometheus 配置 `readme` 文件，或者使用在 Kubecost Github 存储库 中作为引用的示例 Grafana JSON 模型。

- Amazon Cost and Usage Report 集成 – 要为您的 Amazon EKS 集群执行成本分配计算，Kubecost 会从 Amazon 价目表 API 中检索 Amazon Web Services 和 Amazon 资源的公开定价信息。您还可以将 Kubecost 和 Amazon Cost and Usage Report 集成，以提高特定于 Amazon Web Services 账户 的定价信息的准确性。这些信息包括企业折扣计划、预留实例使用情况、Savings Plans 以及 Spot 使用情况。要了解有关 Amazon Cost and Usage Report 集成工作原理的更多信息，请参阅 Kubecost 文档中的 Amazon 云集成。

删除 Kubecost

您可以使用以下命令从您的集群删除 Kubecost。

```
helm uninstall kubecost --namespace kubecost
kubectl delete ns kubecost
```

安装 Kubernetes Metrics Server

Kubernetes Metrics Server 是集群中资源使用数据的聚合器，它在 Amazon EKS 集群中默认不部署。有关更多信息，请参阅 GitHub 上的 Kubernetes Metrics Server。Metrics Server 通常由其他 Kubernetes 附加组件使用，例如 Horizontal Pod Autoscaler (p. 325) 或 Kubernetes 控制面板 (p. 374)。有关详细信息，请

参阅 Kubernetes 文档中的资源指标管道。本主题介绍了如何在您的 Amazon EKS 集群上部署 Kubernetes Metrics Server。

Important

当您需要准确的资源使用指标来源时，请勿使用 Metrics Server，也不要将其用作监控解决方案。

部署 Metrics Server

1. 使用下面的命令下载清单：

```
curl -o components.yaml https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

2. 查看您下载的一个或多个清单文件并记下镜像的名称。使用下面的命令将镜像下载到本地。

```
docker pull image:tag
```

3. 使用以下命令标记要推送到中国 Amazon Elastic Container Registry 存储库的镜像。

```
docker tag image:tag 111122223333.dkr.ecr.cn-north-1.amazonaws.com.cn/image:tag
```

4. 通过以下命令将镜像推送到中国 Amazon ECR 存储库。

```
docker push image:tag 111122223333.dkr.ecr.cn-north-1.amazonaws.com.cn/image:tag
```

5. 更新 Kubernetes 的一个或多个清单文件，以引用您所在 Amazon Web Services 区域 的 Amazon ECR 镜像 URL。
6. 使用以下命令将已修改的清单应用于集群。

```
kubectl apply -f components.yaml
```

7. 使用以下命令验证 `metrics-server` 部署是否运行所需数量的 pods。

```
kubectl get deployment metrics-server -n kube-system
```

输出示例如下。

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           6m
```

Prometheus 的控制层面指标

Kubernetes API 服务器公开了大量可用于监控和分析的指标。这些指标是通过引用 `/metrics` HTTP API 的指标终端节点内部公开的。与其他端点一样，此端点在 Amazon EKS 控制层面上公开。本主题介绍了一些您可以使用此终端节点查看和分析集群所执行的操作的方法。

查看原始指标

要查看原始指标输出，请使用带有 `--raw` 标志的 `kubectl`。此命令允许您传递任何 HTTP 路径，并返回原始响应。

```
kubectl get --raw /metrics
```

输出示例如下。

```
...
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
 method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

此原始输出逐字返回 API 服务器公开的内容。这些指标以 Prometheus 格式表示。此格式允许 API 服务器公开按行拆分的不同指标。每一行包含一个指标名称、标签和值。

```
metric_name{"tag"="value"[,...]} value
```

虽然此终端节点在您查找特定指标时很有用，但您通常需要在一段时间内分析这些指标。要执行此操作，您可以在集群中部署 Prometheus。Prometheus 是一个监控和时间序列数据库，它擦除公开的终端节点并聚合数据，让您可以筛选、用图表表示以及查询结果。

部署 Prometheus

本主题可帮助您使用 Helm V3 在集群中部署 Prometheus。如果您已安装有 Helm，则可使用 `helm version` 命令来检查您的版本。Helm 是 Kubernetes 集群的包管理器。有关 Helm 及其安装方法的更多信息，请参阅 将 Helm 与 Amazon EKS 结合使用 (p. 384)。

在为您的 Amazon EKS 集群配置 Helm 后，您可以使用它来部署 Prometheus，步骤如下。

使用 Helm 部署 Prometheus

1. 创建 Prometheus 命名空间。

   ```
   kubectl create namespace prometheus
   ```

2. 添加 `prometheus-community` 图表存储库。

   ```
   helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
   ```

3. 部署 Prometheus。

   ```
   helm upgrade -i prometheus prometheus-community/prometheus \
       --namespace prometheus \
       --set
    alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume.storageClass="gp2"
   ```

   Note

   如果您在执行此命令时收到错误 `Error: failed to download "stable/prometheus" (hint: running `helm repo update` may help)`，请运行 `helm repo update`，然后尝试再次运行第 2 步命令。

如果您收到错误 `Error: rendered manifests contain a resource that already exists`，请运行 `helm uninstall your-release-name -n namespace`，然后尝试再次运行第 3 步命令。

4. 确认 `prometheus` 命名空间中的所有 pods 均处于 `READY` 状态。

```
kubectl get pods -n prometheus
```

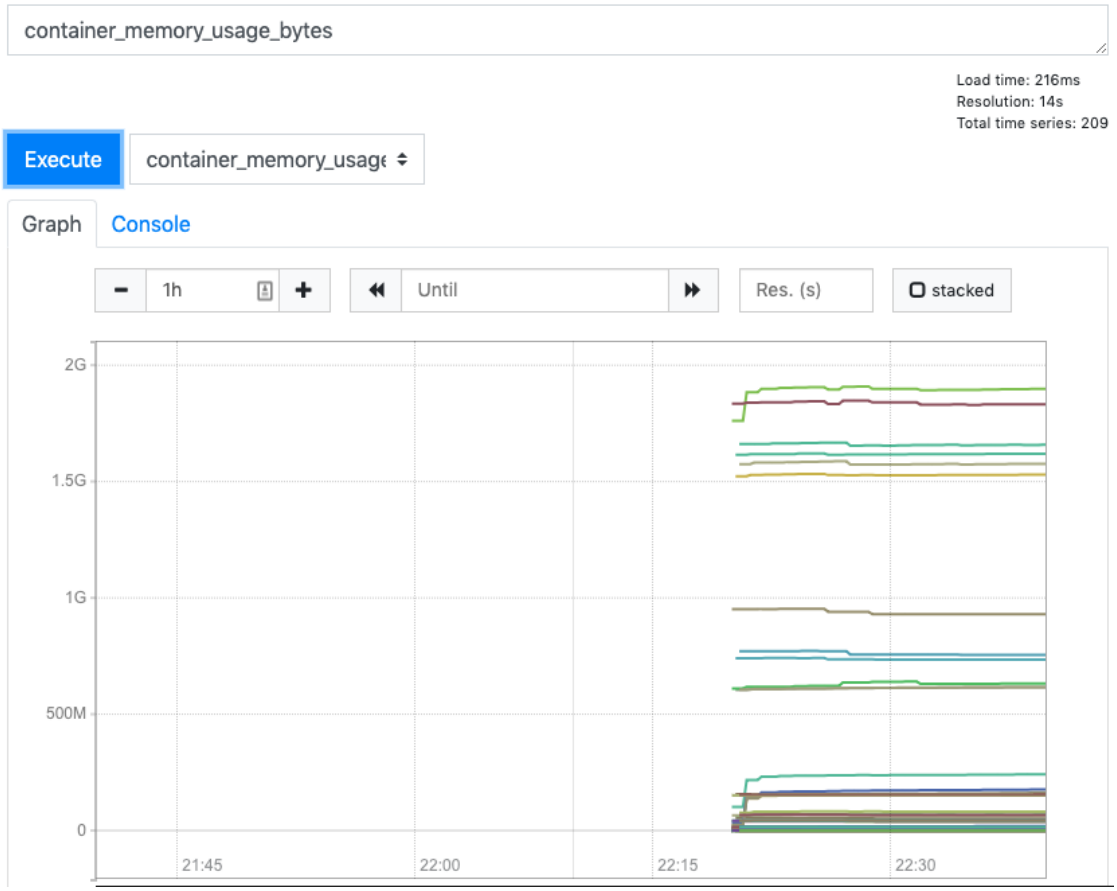
输出示例如下。

```
NAME                                             READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-59b4c8c744-r7bgp         1/2     Running   0          48s
prometheus-kube-state-metrics-7cfd87cf99-jkz2f   1/1     Running   0          48s
prometheus-node-exporter-jcjqz                   1/1     Running   0          48s
prometheus-node-exporter-jxv2h                   1/1     Running   0          48s
prometheus-node-exporter-vbdks                   1/1     Running   0          48s
prometheus-pushgateway-76c444b68c-82tnw          1/1     Running   0          48s
prometheus-server-775957f748-mmht9               1/2     Running   0          48s
```

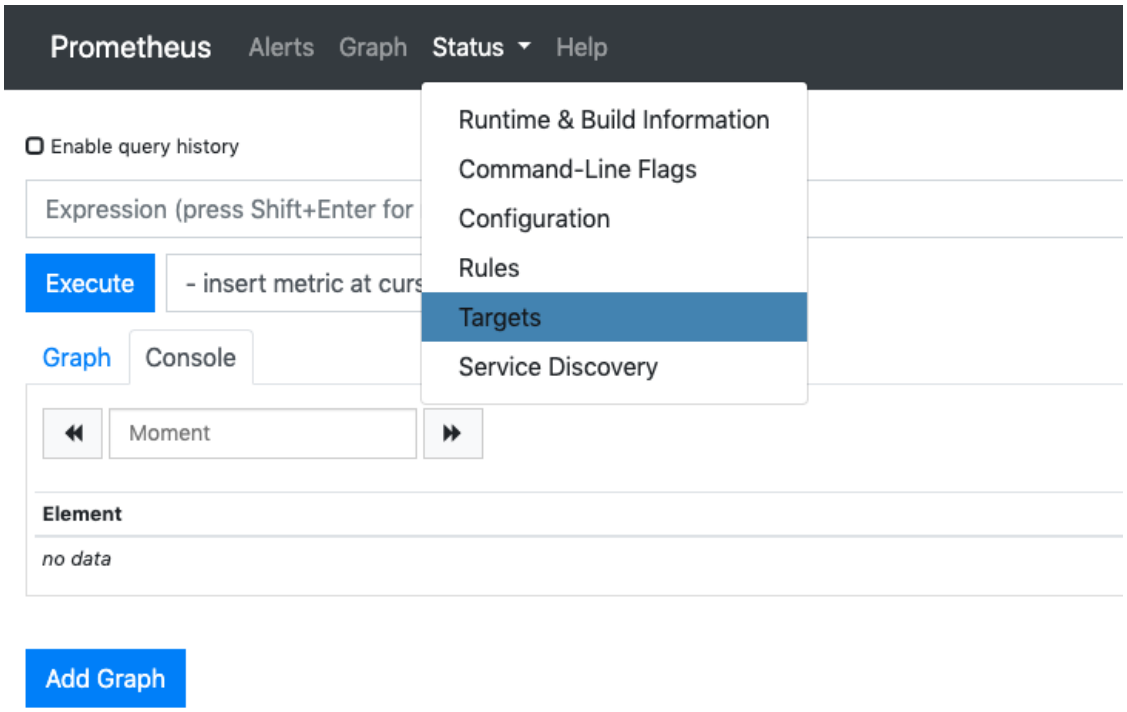
5. 使用 `kubectl` 将 Prometheus 控制台端口转发到本地计算机。

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. 将 Web 浏览器指向 localhost:9090 来查看 Prometheus 控制台。
7. 从 - insert metric at cursor (- 在光标位置插入指标) 菜单选择一个指标，然后选择 Execute (执行)。选择 Graph (图表) 选项卡显示一段时间内的指标。下图显示了一段时间内的 `container_memory_usage_bytes`。



8. 在顶部导航栏中，选择 Status (状态)，然后选择 Targets (目标)。



显示使用服务发现连接到 Prometheus 的所有 Kubernetes 端点。

将您的 Prometheus 指标存储在 Amazon Managed Service for Prometheus

Amazon Managed Service for Prometheus 是一项与 Prometheus 兼容的监控和警报服务，可以轻松实现对容器化应用程序和基础设施的大规模监控。这是一项完全托管的服务，可自动扩展指标的提取、存储、查询和警报。这项服务还集成了 Amazon 安全服务，可以快速安全地访问您的数据。您可以使用开源 ProMQL 查询语言来查询指标并发出警报。

有关更多信息，请参阅开始使用 Amazon Managed Service Managed Service Prometheus。

将 Helm 与 Amazon EKS 结合使用

Kubernetes 的 Helm 程序包管理器帮助您在 Kubernetes 集群上安装和管理应用程序。有关更多信息，请参阅 Helm 文档。本主题可帮助您安装并运行 Helm 二进制文件，以便您可以在本地系统中使用 Helm CLI 安装和管理图表。

Important

您必须先将 `kubectl` 配置为用于 Amazon EKS，然后才能在 Amazon EKS 集群上安装 Helm Chart。如果您尚未执行此操作，请参阅为 Amazon EKS 创建 `kubeconfig` (p. 365)后再继续。如果集群的以下命令成功，说明您已正确配置。

```
kubectl get svc
```

在本地系统上安装 Helm 二进制文件

1. 运行适用于您的客户端操作系统的命令。

   - 如果您将 macOS 与 Homebrew 配合使用，请使用以下命令安装二进制文件。

     ```
     brew install helm
     ```

   - 如果您将 Windows 与 Chocolatey 配合使用，请使用以下命令安装二进制文件。

     ```
     choco install kubernetes-helm
     ```

   - 如果您正在使用 Linux，请使用以下命令来安装二进制文件。

     ```
     curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 >
      get_helm.sh
     chmod 700 get_helm.sh
     ./get_helm.sh
     ```

     Note

     如果您收到一条消息，显示必须首先安装 `openssl`，则可以使用以下命令进行安装。

     ```
     sudo yum install openssl
     ```

2. 要在 `PATH` 中选择新的二进制文件，请关闭当前的终端窗口，然后打开一个新窗口。
3. 查看您已安装的 Helm 版本。

   ```
   helm version --short | cut -d + -f 1
   ```

   输出示例如下。

   ```
   v3.9.0
   ```

4. 此时，您可以运行任何 Helm 命令（例如 `helm install` *chart-name*），以便安装、修改、删除或查询您的集群中的 Helm Chart。如果您刚接触 Helm，并且没有要安装的特定图表，您可以：

   - 通过安装示例图表来进行试验。请参阅 Helm 快速入门指南中的安装示例图表。
   - 创建示例图表并将其推送至 Amazon ECR。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的推送 Helm Chart。
   - 从 eks-charts GitHub 存储库或从 ArtifactHub 中安装 Amazon EKS 图表。

为您的 Amazon EKS 资源添加标签

您可以使用标签帮助您管理 Amazon EKS 资源。本主题提供了标签功能的概述，并说明如何创建标签。

主题

- 有关标签的基本知识 (p. 386)
- 给您的 资源加标签 (p. 386)
- 标签限制 (p. 386)
- 标记资源以便于计费 (p. 387)
- 通过控制台使用标签 (p. 387)
- 通过 CLI、API 或 eksctl 使用标签 (p. 388)

Note

标签是一种与Kubernetes 标签和注释分开的元数据。有关这些其他元数据类型的更多信息，请参阅 Kubernetes 文档中的以下各节：

- 标签和选择器
- 注释

有关标签的基本知识

标签是为 Amazon 资源分配的标记。每个标签都包含一个键和一个可选值。您可以同时定义两者。

借助标签，您可以对 Amazon 资源进行分类。例如，您可以按用途、所有者或环境对资源进行分类。在您具有相同类型的许多资源时，可以使用分配给特定资源的标签来快速识别该资源。例如，您可以为 Amazon EKS 集群定义一组标签，以帮助您跟踪每个集群的拥有者和堆栈级别。我们建议您为每个资源类型设计一组一致的标签键。然后，您可以根据添加的标签搜索和筛选资源。

添加标签后，您可以编辑标签键和值，还可以随时删除资源的标签。如果删除资源，资源的所有标签也会被删除。

标签对 Amazon EKS 没有任何语义意义，应严格按字符串进行解析。您可以将标签值设置为空字符串。但是，您不能将标签值设置为 null。如果您添加的标签的键与该资源上现有标签的键相同，则新值会覆盖旧值。

如果您使用的是 Amazon Identity and Access Management（IAM），则可以控制您的Amazon账户中的哪些用户拥有管理标签的权限。

给您的 资源加标签

以下 Amazon EKS 资源支持标签：

- 集群
- 托管节点组
- Fargate 配置文件

您可以使用以下内容标记这些资源：

- 如果您使用的是 Amazon EKS 控制台，可以随时对新的或现有的资源应用标签。您可以使用相关资源页面上的标签选项卡执行此操作。有关更多信息，请参阅通过控制台使用标签 (p. 387)。
- 如果您使用的是 `eksctl`，可以在使用 `--tags` 选项创建资源时为资源应用标签。
- 如果您使用的是 Amazon CLI、Amazon EKS API 或 Amazon 开发工具包，则可以使用相关 API 操作上的 `tags` 参数对新资源应用标签。您也可以通过使用 `TagResource` API 操作将标签应用于现有资源。有关更多信息，请参阅 TagResource。

在您使用一些资源创建操作时，您还可以在创建资源的同时为资源指定标签。如果在创建资源期间无法应用标签，则创建资源会失败。此机制可确保对于您希望标记的资源，要么使用您指定的标签创建，要么完全不创建。如果您在创建资源时标记这些资源，则无需在创建资源后运行自定义标记脚本。

标签不会传播到与您创建的资源关联的其他资源。例如，Fargate 配置文件标签不会传播到与 Fargate 配置文件关联的其他资源，例如使用配置文件调度的 pods。

标签限制

以下限制适用于标签：

- 一个资源最多可以关联 50 个标签。
- 不能对一个资源重复使用标签键。每个标签键必须具有唯一性，而且只能有一个值。
- 键最长可达 128 个字符（采用 UTF-8 格式）。
- 值最长可达 256 个字符（采用 UTF-8 格式）。
- 如果有多个 Amazon Web Services 和资源使用您的标记方案，请限制您使用的字符类型。某些服务可能对允许使用的字符有限制。通常允许使用的字符包括字母、数字、空格以及以下字符：`+ - = . _ : / @`。
- 标签键和值区分大小写。
- 请不要使用 `aws:`、`AWS:` 或任何大写或小写组合（例如，键或值的前缀）。它们保留供 Amazon 使用。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。

标记资源以便于计费

将标签应用于 Amazon EKS 集群时，您可以使用它们在成本和使用情况报告中进行成本分配。成本和使用情况报告中的计量数据显示了所有 Amazon ECS 集群的使用情况。有关更多信息，请参阅 Amazon Billing 用户指南中的 Amazon 成本和使用情况报告。

Amazon 生成的成本分配标签，特别是 `aws:eks:cluster-name`，会自动添加到参与 Amazon EKS 集群的 Amazon EC2 实例中。即使资源不是通过 Amazon EKS 托管节点组启动的，也会添加此标签。此标签不计入您可以与资源关联的 50 个标签限制。要按单个 Amazon EKS 集群细分 Amazon EC2 成本，账户所有者必须在 Amazon Billing 控制台中或通过使用 API 激活此标签。当 Amazon Organizations 管理账户所有者激活此标签时，将同时为所有组织成员账户激活此标签。

您还可以根据具有相同标签键值的资源组织您的账单信息。例如，您可以将特定的应用程序名称用作几个资源的标签，然后组织您的账单信息。这样，您可以查看多个服务中使用该应用程序的总成本。有关设置带有标签的成本分配报告的更多信息，请参阅 Amazon Billing 用户指南中的月度成本分配报告。

> Note
>
> 如果您刚刚启用报告，则可以在 24 小时后查看当月的数据。

Cost Explorer 是一个报告工具，作为 Amazon 免费套餐的一部分提供。您可以使用 Cost Explorer 查看过去 13 个月的 Amazon EKS 资源图表。您还可以预测您在接下来三个月内可能产生的费用。您可以查看您在 Amazon 资源上的花费随时间变化的模式。如，您可以使用它来确定需要进一步查询的方面，并查看可用于了解成本的趋势。您还可以指定数据的时间范围，并按天或按月查看时间数据。

通过控制台使用标签

通过使用 Amazon EKS 控制台，您可以管理与新的或现有的集群和托管节点组关联的标签。

当您在 Amazon EKS 控制台中选择特定资源页面时，该页面会显示这些资源的列表。例如，如果您从左侧导航窗格中选择 Clusters（集群），则控制台会显示 Amazon EKS 集群列表。当您从其中一个列表中选择一种支持标签的资源（例如，特定集群）时，您可以在标签选项卡上查看和管理其标签。

您还可以在 Amazon Web Services Management Console 中使用标签编辑器，它为管理标签提供了统一的方法。有关更多信息，请参阅《Amazon Resource Groups 和标签用户指南》中的使用标签编辑器。

在创建时为资源添加标签

您可以在创建 Amazon EKS 集群、托管节点组和 Fargate 配置文件时，为它们添加标签。有关更多信息，请参阅创建 Amazon EKS 集群 (p. 21)。

为资源添加和删除标签

您可以直接从资源的页面中添加或删除与集群关联的标签。

添加或删除单个资源上的标签

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 在导航栏中，选择要使用的 Amazon Web Services 区域。
3. 在左侧导航窗格中，选择集群。
4. 选择特定集群。
5. 选择标签选项卡，然后选择管理标签。
6. 在 Manage tags（管理标签）页面上，根据需要添加或删除标签。
   - 要添加标签，请选择 Add tag（添加标签）。然后，指定每个标签的键和值。
   - 要删除标签，请选择 Remove tag（删除标签）。
7. 对要添加或删除的每个标签重复此过程。
8. 选择 Update (更新) 完成操作。

通过 CLI、API 或 eksctl 使用标签

使用以下 Amazon CLI 命令或 Amazon EKS API 操作来添加、更新、列出和删除资源的标签。您只能使用 eksctl 在使用一个命令同时创建新资源的同时添加标签。

Amazon EKS 资源标记支持

任务	Amazon CLI	Amazon Tools for Windows PowerShell	API 操作
添加或覆盖一个或多个标签。	tag-resource	Add-EKSResourceTag	TagResource
删除一个或多个标签。	untag-resource	Remove-EKSResourceTag	UntagResource

以下示例说明如何使用 Amazon CLI 标记或取消标记资源。

示例 1：标记现有集群

以下命令标记现有集群。

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

示例 2：取消标记现有集群

以下命令从现有集群删除标签。

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

示例 3：列出资源的标签

以下命令列出与现有资源关联的标签。

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

在您使用一些资源创建操作时，您可以在创建资源的同时指定标签。以下操作支持在创建资源时指定标签。

任务	Amazon CLI	Amazon Tools for Windows PowerShell	API 操作	**eksctl**
创建 集群	create-cluster	New-EKSCluster	CreateCluster	create cluster
创建托管节点组*	create-nodegroup	New-EKSNodegroup	CreateNodegroup	create nodegroup
创建 Fargate 配置文件	create-fargate-profile	New-EKSFargateProfile	CreateFargateProfile	create fargateprofile

* 如果您还想在创建托管节点组时为 Amazon EC2 实例添加标签，请使用启动模板创建托管节点组。有关更多信息，请参阅为 Amazon EC2 实例添加标签 (p. 102)。如果您的实例已经存在，您可以手动为实例添加标签。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的标记您的资源。

Amazon EKS 服务配额

Amazon EKS 已与服务限额集成，后者是一项 Amazon 服务，您可以使用该服务从中心位置查看和管理您的限额。有关更多信息，请参阅 Service Quotas 用户指南中的什么是 Service Quotas？ 借助服务限额集成，您可以使用 Amazon Web Services Management Console 和 Amazon CLI 快速查找 Amazon EKS 和 Amazon Fargate 服务限额的值。

Amazon Web Services Management Console

要使用 Amazon Web Services Management Console 进一步处理服务限额，请参阅服务限额用户指南。要请求提高配额，请参阅 Service Quotas 用户指南中的请求增加配额。

Amazon CLI

使用 Amazon CLI 查看 Amazon EKS 和 Fargate 服务配额

运行以下命令查看您的 Amazon EKS 配额。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code eks \
    --output table
```

运行以下命令查看您的 Fargate 配额。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

Note

返回的限额是当前 Amazon Web Services 区域 中此账户下可在 Fargate 上并发运行的 Amazon ECS 任务或 Amazon EKS pods 的数量。

要使用 Amazon CLI 进一步处理服务配额，请参阅 Service Quotas Amazon CLI 命令参考。要请求提高配额，请参阅 Amazon CLI 命令参考中的 `request-service-quota-increase` 命令。

服务限额

名称	默认值	可调整	描述
集群	每个支持的区域：100 个	是	当前区域中的此账户中的 EKS 集群的最大数量。
每个集群的控制面板安全组	每个支持的区域：4 个	否	每个集群的控制面板安全组的最大数量（在创建集群时指定）。
每个集群的 Fargate 配置文件	每个支持的区域：10 个	是	每个集群的最大 Fargate 配置文件数量。
每个 Fargate 配置文件选择器的标签对数	每个支持的区域：5 个	是	每个 Fargate 配置文件选择器的最大标签对数量。
每个集群的托管式节点组数	每个支持的区域：30 个	是	每个集群的最大托管节点组数。
每个托管式节点组的节点数	每个支持的区域：450 个	是	每个托管节点组的最大节点数。
每个集群的公有终端节点访问 CIDR 范围数	每个支持的区域：40 个	否	每个集群的最大公有端点访问 CIDR 范围数（在创建或更新集群时指定）。
已注册集群	每个支持的区域：10 个	是	当前区域中的此账户中的注册集群的最大数量。
每个 Fargate 配置文件的选择器数	每个支持的区域：5 个	是	每个 Fargate 配置文件的最大选择器数量。

Note

默认值是由 Amazon 设置的初始限额。这些默认值与实际应用的限额值和最大可能的服务限额是分开的。有关更多信息，请参阅《服务限额用户指南》中的服务限额中的术语。

Amazon Fargate 服务配额

本节介绍 Amazon EKS 的 Amazon Fargate 服务限额。您可以配置警报，以在用量接近服务配额时向您发出警报。有关更多信息，请参阅创建 CloudWatch 警报以监控 Fargate 资源使用情况指标 (p. 138)。

Amazon Fargate 将从基于容器组（pod）的限额过渡到基于 vCPU 的限额。下面是与基于 vCPU 的新限额相关的重要日期。

- 2022 年 9 月 8 日 – 在自动迁移之前，您现在可以选择加入以使用基于 vCPU 的新限额。选择加入后，您的账户将由基于 vCPU 的限额控制，而不是之前基于容器组（pod）的限额控制。对于未选择加入的账户，基于容器组（pod）的限额仍然是默认限额。

  Note

  要在 2022 年 10 月 3 日之前对 Amazon ECS 使用基于 vCPU 的限额，您必须选择加入。要选择加入，请使用 Amazon Support Center Console 创建一个 Service limit increase（提高服务限额）案例。对于 Limit type（限额类型），选择 Fargate，对于 Limit（限额），选择

Fargate account vCPU limit opt-in（Fargate 账户 vCPU 限额选择加入）。有关更多信息，请参阅 Amazon Web Services Support 用户指南中的创建支持案例。

- 2022 年 10 月 3 日至 2022 年 10 月 21 日 – 所有新账户和现有账户都将分阶段切换到基于 vCPU 的限额。

  Note

  要继续使用基于容器组（pod）的限额，您必须选择退出。
  要选择退出，请使用 Amazon Support Center Console 创建一个 Service limit increase（提高服务限额）案例。对于 Limit type（限额类型），选择 Fargate，对于 Limit（限额），选择 Fargate account opt-out of vCPU limit（Fargate 账户选择退出 vCPU 限额）。有关更多信息，请参阅 Amazon Web Services Support 用户指南中的创建支持案例。
- 2022 年 10 月 31 日 – 这是您仍然可以选择退出基于 vCPU 的限额的最后一天。
- 2022 年 11 月 1 日至 2022 年 11 月 15 日 – 选择退出选项结束，所有账户都会迁移到基于 vCPU 的限额。基于容器组（pod）的限额将不再可用。

下表首先列出了基于 vCPU 的新限额，然后列出了现有基于容器组（pod）的限额。这些服务限额包括在服务限额控制台中的 Amazon Fargate 服务下列出的服务限额中。下表仅描述了同样适用于 Amazon EKS 的限额。

您可以通过查看服务限额控制台来确认生效的限额类型。如果生效的是 vCPU 限额，则对于 Applied quota value（应用的限额值），基于 Fargate 按需型容器组（pod）计数的限额将显示 0。任何其他值均表示生效的是容器组（pod）计数限额。

新 Amazon Web Services 账户 的初始限额可能较低，但会随着时间的推移而增加。Fargate 会持续监控每个 Amazon Web Services 区域 内的账户使用情况，然后根据使用情况自动增加限额。对于显示为可调整的值，您还可以请求提高限额。有关更多信息，请参阅 Service Quotas 用户指南 中的请求增加配额。

名称	默认值	可调整	描述
Fargate 按需型 vCPU 资源计数	6	是	此账户在当前区域中可作为 Fargate 按需型实例并发运行的 Fargate vCPU 数量。
Fargate 按需资源计数	2	是	此账户在当前区域中可在 Fargate 上并发运行的 Amazon ECS 任务和 Amazon EKS 容器组（pod）的数量。

Note

默认值是由 Amazon 设置的初始限额。这些默认值与实际应用的限额值和最大可能的服务限额是分开的。有关更多信息，请参阅《服务限额用户指南》中的服务限额中的术语。

Note

此外，Fargate 还强制执行 Amazon ECS 任务和 Amazon EKS 容器组（pod）启动率限额。有关更多信息，请参阅适用于 Amazon Fargate 的 Amazon Elastic Container Service 开发人员指南中的 Fargate 节流限额。

Amazon EKS 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。责任共担模型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。对于 Amazon EKS，Amazon 负责 Kubernetes 控制面板，其中包括控制面板节点和 `etcd` 数据库。作为 Amazon 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用于 Amazon EKS 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 – 您的责任包括以下各个方面。
  - 数据层面的安全配置，包括配置安全组以允许流量从 Amazon EKS 控制层面传入客户 VPC
  - 节点和容器本身的配置
  - 节点操作系统（包括更新和安全补丁）
  - 其他关联的应用程序软件：
    - 设置和管理网络控制功能，例如防火墙规则
    - 使用 IAM 或其他服务管理平台级身份和访问管理
  - 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 EKS 时应用责任共担模式。以下主题说明如何配置 Amazon EKS 以实现您的安全性和合规性目标。您还会了解如何使用其他Amazon服务以帮助您监控和保护 Amazon EKS 资源。

> **Note**
>
> Linux 容器由控制组 (cgroup) 和命名空间组成，这些控制组和命名空间有助于限制容器可以访问的内容，但所有容器都与主机 Amazon EC2 实例共享相同的 Linux 内核。非常不建议以根用户 (UID 0) 身份运行容器或授予容器对主机资源或命名空间（如主机网络或主机 PID 命名空间）的访问权限，因为这样做会降低容器提供的隔离的有效性。

主题

- 证书签名 (p. 392)
- Kubernetes 服务账户 (p. 394)
- 适用于 Amazon EKS 的身份和访问管理 (p. 409)
- Amazon Elastic Kubernetes Service 的合规性验证 (p. 451)
- Amazon EKS 中的恢复能力 (p. 451)
- Amazon EKS 中的基础设施安全性 (p. 452)
- Amazon EKS 中的配置和漏洞分析 (p. 452)
- Amazon EKS 的安全最佳实践 (p. 453)
- Pod 安全策略 (p. 453)
- 将 Kubernetes 与 Amazon Secrets Manager 密钥结合使用 (p. 456)
- Amazon EKS Connector 注意事项 (p. 456)

证书签名

Kubernetes 证书 API 支持自动化 X.509 凭证预置。它具有一个命令行界面，供 Kubernetes API 客户端从证书颁发机构 (CA, Certificate Authority) 请求和获取 X.509 证书。`CertificateSigningRequest`（CSR）

资源用于请求指示签署人对证书进行签名。然后，请求在签署前被批准或拒绝。Kubernetes 支持内置签署人和具有明确定义行为的自定义签署人。这样，客户端就可以预测 CSR 会发生什么。要了解有关证书签名的更多信息，请参阅签名请求。

内置签署人之一是 `kubernetes.io/legacy-unknown`。CSR 资源的 `v1beta1` API 遵循这个旧版未知的签署人。但是，CSR 的稳定 v1 API 不允许 `signerName` 被设置为 `kubernetes.io/legacy-unknown`。

Amazon EKS 版本 `1.21` 和早期版本允许将值 `legacy-unknown` 作为 v1beta1 CSR API 中的 `signerName`。此 API 使 Amazon EKS 证书颁发机构（CA）能够生成证书。但是，在 Kubernetes 版本 `1.22` 中，v1beta1 CSR API 被 v1 CSR API 替换。此 API 不支持“旧版未知”的 signerName。鉴于此，如果您想使用 Amazon EKS CA 在版本 `1.22` 和更高版本上生成证书，您需要使用自定义签署人。它已在 Amazon EKS 版本 `1.22` 中引入。要使用 CSR v1 API 版本并生成新证书，必须迁移任何现有清单和 API 客户端。使用旧版本的 `v1beta1` API 创建的现有证书在证书到期之前有效且正常运行。这包括以下这些：

1. 信任分配：无。在 Kubernetes 集群中，此签署人没有标准的信任或分配。
2. 允许的主题：任何
3. 允许的 x509 扩展：遵循 subjectAltName 和密钥使用扩展，并丢弃其他扩展
4. 允许的密钥用法：不得包括 [“密钥加密”、“数字签名”、“服务器身份验证”] 以外的用法
5. 到期/证书使用寿命：1 年（默认值和最大值）
6. 允许/不允许 CA 位：不允许

使用 signerName 生成 CSR 示例

这些步骤介绍如何使用 `signerName: beta.eks.amazonaws.com/app-serving` 为 DNS 名称 `myserver.default.svc` 生成服务证书。将此用作您自己环境的指南。

1. 运行 `openssl genrsa -out myserver.key 2048` 命令以生成 RSA 私有密钥。

```
openssl genrsa -out myserver.key 2048
```

2. 运行以下命令以生成证书请求。

```
openssl req -new -key myserver.key -out myserver.csr -subj "/CN=myserver.default.svc"
```

3. 生成 CSR 请求的 `base64` 值。稍后，您将使用此值作为 CSR 中的 `request` 值。

```
cat myserver.csr | base64 -w 0 | tr -d "\n"
```

4. 使用以下内容创建名为 `mycsr.yaml` 的文件。在以下示例中，`beta.eks.amazonaws.com/app-serving` 是 `signerName`。将 *base64-value* 替换为上一步中返回的值。

```
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: base64-value
  signerName: beta.eks.amazonaws.com/app-serving
  usages:
    - digital signature
    - key encipherment
    - server auth
```

5. 提交 CSR。

```
kubectl apply -f mycsr.yaml
```

6. 批准服务证书。

```
kubectl certificate approve myserver
```

7. 验证证书是否已颁发。

```
kubectl get csr myserver
```

输出示例如下。

```
NAME        AGE     SIGNERNAME                                  REQUESTOR           CONDITION
myserver    3m20s   beta.eks.amazonaws.com/app-serving   kubernetes-admin
 Approved,Issued
```

8. 导出已颁发的证书：

```
kubectl get csr myserver -o jsonpath='{.status.certificate}'| base64 -d > myserver.crt
```

Kubernetes 服务账户

Kubernetes 服务账户为在 pod 中运行的进程提供身份。有关更多信息，请参阅 Kubernetes 文档中的管理服务账户。如果您的 pod 需要访问 Amazon 服务，您可以将服务账户映射到 Amazon Identity and Access Management 身份来授予该访问权限。有关更多信息，请参阅服务账户的 IAM 角色 (p. 395)。

服务账户令牌

Kubernetes 版本 `1.21` 及更高版本中默认启用 `BoundServiceAccountTokenVolume` 功能。此功能允许在 Kubernetes 上运行工作负载，以请求与受众、时间和密钥绑定的 JSON Web 令牌，从而提高了服务账户令牌的安全性。服务账户令牌有效期为 1 小时。在较早的 Kubernetes 版本中，令牌没有过期时间。这意味着依赖这些令牌的客户端必须在一小时内刷新令牌。以下 Kubernetes 客户端开发工具包会在要求的时间范围内自动刷新令牌：

- Go 版本 `0.15.7` 和更高版本
- Python 版本 `12.0.0` 和更高版本
- Java 版本 `9.0.0` 和更高版本
- JavaScript 版本 `0.10.3` 和更高版本
- Ruby `master` 分支
- Haskell 版本 `0.3.0.0`
- C# 版本 `7.0.5` 和更高版本

如果您的工作负载使用的是旧版本的客户端，则必须予以更新。为了使客户端顺利迁移到更新的有时限的服务账户令牌，Kubernetes 版本 `1.21` 及更高版本在默认一小时内向服务账户令牌添加延长的到期期限。对于 Amazon EKS 集群，延长到期期限为 90 天。Amazon EKS 集群的 Kubernetes API 服务器拒绝令牌超过 90 天的请求。我们建议您检查应用程序及其依赖项，以确保 Kubernetes 客户端开发工具包版本等于或高于之前列出的版本。

当 API 服务器收到使用时间超过一小时的令牌的请求时，它会使用 `annotations.authentication.k8s.io/stale-token` 注释 API 审核日志事件。注释的值与以下示例类似：

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold: 4185802.
```

如果您的集群启用了控制面板日志记录 (p. 44)，则注释在审计日志内。您可以使用以下 CloudWatch Logs Insights 查询，以识别 Amazon EKS 集群中使用过时令牌的所有 pods：

```
fields @timestamp
| filter @logStream like /kube-apiserver-audit/
| filter @message like /seconds after warning threshold/
| parse @message "subject: *, seconds after warning threshold:*\"" as subject, elapsedtime
```

`subject` 指的是 pod 使用的服务账户。`elapsedtime` 表示读取最新令牌后的时间（以秒为单位）。对 API 服务器的请求在 `elapsedtime` 超过 90 天时被拒绝。您应该主动更新应用程序的 Kubernetes 客户端开发工具包，以使用前面列出的其中一个自动刷新令牌的版本。如果使用的服务账户令牌接近 90 天，并且您没有足够的时间在令牌到期之前更新客户端开发工具包版本，您可以终止现有 pods 并创建新的。这将导致重新获取服务账户令牌，从而为您提供额外 90 天的时间来更新客户端版本开发工具包。

如果 pod 是部署的一部分，那么在保持高可用性的同时终止 pods 的建议方法是使用以下命令执行部署。将 *my-deployment* 替换为您的部署的名称。

```
kubectl rollout restart deployment/my-deployment
```

集群附加组件

已更新以下集群附加组件以使用自动重新获取服务账户令牌的 Kubernetes 客户端开发工具包。我们建议确保列出的版本或更高版本已安装在 `1.21` 或更高版本的集群上。

- Amazon VPC CNI 和 CNI 指标帮助程序插件版本 `1.8.0` 和更高版本。要查看当前版本或进行更新，请参阅 管理 Amazon VPC CNI plugin for Kubernetes (p. 233) 和 指标帮助程序 (p. 283)。
- CoreDNS 版本 `1.8.4` 和更高版本。要查看当前版本或更新它，请参阅 管理 CoreDNS 附加组件 (p. 294)。
- Amazon Load Balancer Controller 版本 `2.0.0` 和更高版本。要查看当前版本或更新它，请参阅 安装 Amazon Load Balancer Controller 附加组件 (p. 287)。
- `kube-proxy` 版本 `1.21.2-eksbuild.2` 和更高版本。要查看当前版本或更新它，请参阅 管理 `kube-proxy` 附加组件 (p. 300)。
- Amazon for Fluent Bit 版本 `2.25.0` 或更高版本。要更新当前版本，请参阅 GitHub 上的发布。
- Fluentd 镜像版本 1.14.6-1.2 或更高版本以及适用于 Kubernetes 元数据的 Fluentd 筛选器插件版本 2.11.1 或更高版本。

服务账户的 IAM 角色

pod 的容器中的应用程序可以使用 Amazon 开发工具包或 Amazon CLI，以向使用 Amazon Identity and Access Management（IAM）权限的 Amazon Web Services 发出 API 请求。应用程序必须通过 Amazon 凭证签署 Amazon API 请求。服务账户的 IAM 角色可管理供应用程序使用的凭证，这与 Amazon EC2 实例配置文件为 Amazon EC2 实例提供凭证的方式相似。您可以将 IAM 角色与 Kubernetes 服务账户关联并配置容器组（pod）使用服务账户，而不是创建 Amazon 凭证并将其分配到容器或使用 Amazon EC2 实例的角色。

服务账户的 IAM 角色提供下列优势：

- 最低权限 – 您可以将 IAM 权限范围限定到服务账户，并且只有使用该服务账户的 pods 可以访问这些权限。此功能还消除了对 `kiam` 或 `kube2iam` 等第三方解决方案的需求。
- 凭证隔离 – 容器组（pod）的容器只能检索与该容器所使用服务账户关联的 IAM 角色的凭证。容器永远无法访问其他 pods 中其他容器所使用的凭证。在使用服务账户的 IAM 角色时，pod's 容器还具有分配给 Amazon EKS 节点 IAM 角色 (p. 427) 的权限，除非你阻止 pod 访问 Amazon EC2 实例元数据服务（IMDS）。有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。
- 可审核性 – 可通过 Amazon CloudTrail 进行访问和事件日志记录以帮助确保可追溯性审核。

通过完成下列步骤为服务账户启用 IAM 角色：

1. the section called “创建 IAM OIDC 提供商” (p. 396) – 对于每个集群，您只需完成一次此步骤。
2. the section called “配置 Amazon STS 端点” (p. 405) – 针对您希望应用程序拥有的每组唯一权限完成此步骤。
3. the section called “配置容器组（pod）” (p. 402) – 为需要访问 Amazon Web Services 的每个容器组（pod）完成此步骤。

IAM、Kubernetes 以及 OpenID Connect（OIDC）背景信息

2014 年，Amazon Identity and Access Management 使用 OpenID Connect（OIDC）增加了对联合身份验证的支持。此功能允许您通过支持的身份提供商对 Amazon API 调用进行身份验证，并获得有效的 OIDC JSON Web 令牌（JWT）。您可以将此令牌传递到 Amazon STS `AssumeRoleWithWebIdentity` API 操作并接收 IAM 临时角色凭证。您可以使用这些凭证与任意 Amazon Web Service 交互，包括 Amazon S3 和 DynamoDB。

Kubernetes 长期以来将服务账户用作其内部身份系统。Pods 可以使用自动装载的令牌（这是非 OIDC JWT，只有 Kubernetes API 服务器可以验证）进行 Kubernetes API 服务器的身份验证。这些旧服务账户令牌不会过期，轮换签名密钥是一个困难的过程。在 Kubernetes 版本 `1.12` 中，添加了对新 `ProjectedServiceAccountToken` 功能的支持。此功能是也包含服务账户身份的 OIDC JSON Web 令牌，并支持可配置的受众。

Amazon EKS 为包含 `ProjectedServiceAccountToken` JSON Web 令牌的签名密钥的每个集群托管公有 OIDC 发现端点，这样 IAM 等外部系统就可以验证和接收 Kubernetes 颁发的 OIDC 令牌。

为集群创建 IAM OIDC 提供商

您的集群具有与其关联的 OpenID Connect（OIDC）颁发者 URL。要将 Amazon Identity and Access Management（IAM）角色用于服务账户，您的集群必须存在 IAM OIDC 提供商。

先决条件

- 现有 版 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS (p. 4)。
- 您的设备或 Amazon CloudShell 上安装并配置了 `2.7.21` 版或更高版本，或 `1.25.46` 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 `aws configure` 进行快速配置。
- 您的设备或 Amazon CloudShell 上安装了 `kubectl` 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 `1.22`，则可以将 `kubectl` 版本 `1.21`、`1.22` 或 `1.23` 用于它。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl` (p. 4)。
- 包含集群配置的现有 `kubectl config` 文件。要创建 `kubectl config` 文件，请参阅 为 Amazon EKS 创建 `kubeconfig` (p. 365)。

您可以使用 `eksctl` 或 Amazon Web Services Management Console 为集群创建 OIDC 提供商。

eksctl

先决条件

您的设备或 Amazon CloudShell 上安装了 `0.110.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 安装或更新 `eksctl` (p. 10)。

使用 **`eksctl`** 为集群创建 IAM OIDC 身份提供商

1. 确定集群是否拥有现有 IAM OIDC 提供商。

   检索集群的 OIDC 提供商 ID 并将其存储在变量中。

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

2. 确定您的账户中是否已存在具有您的集群 ID 的 IAM OIDC 提供商。

```
aws iam list-open-id-connect-providers | grep $oidc_id
```

如果上一个命令返回了输出，则表示您的集群已经有提供商，您可以跳过下一步。如果没有返回输出，则您必须为集群创建 IAM OIDC 提供商。

3. 使用以下命令为您的集群创建 IAM OIDC 身份提供商。将 *my-cluster* 替换为您自己的值。

```
eksctl utils associate-iam-oidc-provider --cluster my-cluster --approve
```

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 为集群创建 IAM OIDC 身份提供商

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 在左侧窗格中，选择 Clusters（集群），然后在 Clusters（集群）页面上选择集群的名称。
3. 在 Overview（概述）选项卡上的 Details（详细信息）部分中，记下 OpenID Connect provider URL（OpenID Connect 提供商 URL）的值。
4. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
5. 请在左侧导航窗格中，选择 Access management（访问管理）下方的 Identity Providers（标识提供程序）。如果列出的 Provider（提供商）与集群的 URL 匹配，那么您的集群已经有了提供商。如果未列出与集群 URL 匹配的提供商，则必须创建一个提供商。
6. 要创建提供商，请选择 Add Provider（添加提供商）。
7. 对于 Provider type（提供商类型），请选择 OpenID Connect。
8. 对于 Provider URL（提供商 URL），输入集群的 OIDC 提供商 URL，然后选择 Get thumbprint（获取指纹）。
9. 对于 Audience（受众），输入 **sts.amazonaws.com**，然后选择 Add provider（添加提供商）。

下一步

the section called “配置角色和服务账户” (p. 397)

配置 Kubernetes 服务账户以代入 IAM 角色

本主题介绍如何配置 Kubernetes 服务账户以代入 Amazon Identity and Access Management（IAM）角色。然后，配置为使用服务账号的任何容器组（pod）都可以访问该角色有权访问的任何 Amazon Web Service。

先决条件

- 现有集群。如果您没有，可以按照 开始使用 Amazon EKS (p. 4) 指南之一创建一个。
- 集群的现有 IAM OpenID Connect（OIDC）提供商。要了解您是否已拥有一个提供商或如何创建一个提供商，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。
- 您的设备或 Amazon CloudShell 上安装并配置了 `2.7.21` 版或更高版本，或 `1.25.46` 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 `aws configure` 进行快速配置。
- 您的设备或 Amazon CloudShell 上安装了 `kubectl` 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 `1.22`，则可以将 `kubectl` 版本 `1.21`、`1.22` 或 `1.23` 用于它。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl` (p. 4)。

- 包含集群配置的现有 `kubectl config` 文件。要创建 `kubectl config` 文件，请参阅 为 Amazon EKS 创建 kubeconfig (p. 365)。

将 IAM 角色与 Kubernetes 服务账户关联。

1. 要将现有 IAM policy 关联到您的 IAM 角色，请跳至下一步 (p. 398)。

   创建一个 IAM policy。您可以创建自己的策略，也可以复制已授予您部分所需权限的 Amazon 托管策略，并根据您的特定要求对其进行自定义。有关更多信息，请参阅 IAM 用户指南 中的创建 IAM policy。

   a. 创建一个包含 pods 访问 Amazon Web Services 所需权限的文件。有关所有 Amazon Web Services 的所有操作的列表，请参阅服务授权参考。

      您可以运行以下命令创建一个示例策略文件，以实现对 Amazon S3 存储桶的只读访问权限。您可以选择将配置信息或引导脚本存储在此存储桶中，并且您 pod 中的容器可以从存储桶读取文件并将其加载到应用程序中。如果您要创建此示例策略，请将以下内容复制到您的设备。将 *my-pod-secrets-bucket* 替换为您的存储桶名称并运行该命令。

      ```
      cat >my-policy.json <<EOF
      {
          "Version": "2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Action": "s3:GetObject",
                  "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
              }
          ]
      }
      EOF
      ```

   b. 创建 IAM policy。

      ```
      aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
      ```

2. 创建 IAM 角色并将其与 Kubernetes 服务账户关联。您可以使用 `eksctl` 或 Amazon CLI。

   eksctl

   先决条件

   您的设备或 Amazon CloudShell 上安装了 `0.110.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 安装或更新 eksctl (p. 10)。

   将 *my-service-account* 替换为您希望 `eksctl` 创建并与 IAM 角色关联的 Kubernetes 服务账户的名称。将 *default* 替换为您希望 `eksctl` 在其中创建服务账户的命名空间。将 *my-cluster* 替换为您集群的名称。将 *my-role* 替换为您希望将服务账户关联到的角色名称。如果角色尚不存在，`eksctl` 会为您创建它。将 *111122223333* 替换为您的账户 ID，并将 *my-policy* 替换为现有策略的名称。

   ```
   eksctl create iamserviceaccount --name my-service-account --namespace default --
   cluster my-cluster --role-name "my-role" \
       --attach-policy-arn arn:aws:iam::111122223333:policy/my-policy --approve
   ```

Important

- 如果角色或服务账户已经存在，则之前的命令可能会失败。在这些情况下，`Eksctl` 提供不同的选项供您使用。有关更多信息，请运行 **`eksctl create iamserviceaccount --help`**。
- 在使用具有 pod 的服务账户 (p. 402)之前，您指定的或 `eksctl` 创建的服务账户必须绑定到现有的 Kubernetes `role`，或包括服务账户所需 Kubernetes 权限的 `clusterrole`。有关更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

Amazon CLI

1. 如果您有要代入 IAM 角色的现有 Kubernetes 服务账户，则您可以跳过此步骤。

   创建 Kubernetes 服务账户。将以下内容复制到您的设备。将 *`my-service-account`* 替换为所需的名称，如有必要，将 *`default`* 替换为其他命名空间。如更改 *`default`*，则命名空间必须已经存在。

   ```
   cat >my-service-account.yaml <<EOF
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: my-service-account
     namespace: default
   EOF
   kubectl apply -f my-service-account.yaml
   ```

   Important

   在使用具有 pod 的服务账户 (p. 402)之前，您创建的服务账户必须绑定到现有的 Kubernetes `role`，或包括服务账户所需 Kubernetes 权限的 `clusterrole`。有关更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

2. 使用以下命令将 Amazon Web Services 账户 ID 设置为环境变量。

   ```
   account_id=$(aws sts get-caller-identity --query "Account" --output text)
   ```

3. 使用以下命令将集群的 OIDC 身份提供商设置为环境变量。将 *`my-cluster`* 替换为您的集群名称。

   ```
   oidc_provider=$(aws eks describe-cluster --name my-cluster --query
    "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
   ```

4. 为服务账户的命名空间和名称设置变量。将 *`my-service-account`* 替换为要代入角色的 Kubernetes 服务账户。将 *`default`* 替换为服务账户的命名空间。

   ```
   export namespace=default
   export service_account=my-service-account
   ```

5. 运行以下命令为 IAM 角色创建信任策略文件。若要允许命名空间内的所有服务账户使用该角色，请将以下内容复制到您的设备。将 *`StringEquals`* 替换为 **`StringLike`**，并将 *`$service_account`* 替换为 **`*`**。您可在 `StringEquals` 和 `StringLike` 条件中添加多个条目，以允许多个服务账户或命名空间代入角色。要允许其他 Amazon Web Services 账户 中的角色（而不是您的集群所在账户中的角色）代入该角色户，请参阅 the section called “跨账户 IAM 权限” (p. 407) 获取更多信息。

   ```
   cat >trust-relationship.json <<EOF
   {
   ```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:
$namespace:$service_account"
        }
      }
    }
  ]
}
EOF
```

6. 创建角色。将 *my-role* 替换为您的 IAM 角色名称，并将 *my-role-description* 替换为您的角色描述。

```
aws iam create-role --role-name my-role --assume-role-policy-document
 file://trust-relationship.json --description "my-role-description"
```

7. 将 IAM policy 附加到您的角色。将 *my-role* 替换为您的 IAM 角色的名称，并将 *my-policy* 替换为您创建的现有策略的名称。

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:iam::
$account_id:policy/my-policy
```

8. 请使用要让服务账户代入的 IAM 角色的 Amazon 资源名称（ARN）注释您的服务账户。将 *my-role* 替换为您的现有 IAM 角色的名称。假设您在上一步中允许其他 Amazon Web Services 账户 中的角色（而不是您的集群所在账户中的角色）代入该角色户。那么，请确保指定其他账户中的 Amazon Web Services 账户 和角色。有关更多信息，请参阅the section called “跨账户 IAM 权限” (p. 407)。

```
kubectl annotate serviceaccount -n $namespace $service_account
 eks.amazonaws.com.cn/role-arn=arn:aws:iam::$account_id:role/my-role
```

3. 确认角色和服务账户配置正确。

   a. 确认 IAM 角色的信任策略配置正确。

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

   输出示例如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
```

```
                    "oidc.eks.region-code.amazonaws.com.cn/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-
service-account",
                    "oidc.eks.region-code.amazonaws.com.cn/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
}
```

b. 确认您在上一步中附加到角色的策略已附加到该角色。

```
aws iam list-attached-role-policies --role-name my-role --query
 AttachedPolicies[].PolicyArn --output text
```

输出示例如下。

```
arn:aws:iam::111122223333:policy/my-policy
```

c. 设置一个变量以存储要使用策略的 Amazon 资源名称（ARN）。将 *my-policy* 替换为要确认其权限的策略的名称。

```
export policy_arn=arn:aws:iam::111122223333:policy/my-policy
```

d. 查看该策略的默认版本。

```
aws iam get-policy --policy-arn $policy_arn
```

输出示例如下。

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws:iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        ...
    }
}
```

e. 查看策略内容以确保该策略包括您的 pod 所需的所有权限。如有必要，将以下命令中的 *1* 替换为上一个输出中返回的版本。

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

输出示例如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
```

```
}
```

如果您在上一步中创建了示例策略，则您的输出与示例相同。如果您创建了其他策略，则与*##*内容不同。

f. 确认使用角色注释 Kubernetes 服务账户。

```
kubectl describe serviceaccount my-service-account -n default
```

输出示例如下。

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com.cn/role-arn:
 arn:aws:iam::111122223333:role/my-role
Image pull secrets:  <none>
Mountable secrets:   my-service-account-token-qqjfl
Tokens:              my-service-account-token-qqjfl
...
```

4. （可选）the section called “配置 Amazon STS 端点” (p. 405)。Amazon 建议使用区域 Amazon STS 端点而不是全局端点。这可以减少延迟，提供内置冗余并提高会话令牌的有效性。

下一步

the section called “配置容器组（pod）” (p. 402)

配置容器组（pod）以使用 Kubernetes 服务账户

如果 pod 需要访问 Amazon Web Services，则您必须配置它以使用 Kubernetes 服务账户。服务账户必须关联到有权访问 Amazon Web Services 的 Amazon Identity and Access Management（IAM）角色。

先决条件

- 现有集群。如果还没有，可以使用 开始使用 Amazon EKS (p. 4) 指南之一创建一个。
- 集群的现有 IAM OpenID Connect（OIDC）提供商。要了解您是否已拥有一个提供商或如何创建一个提供商，请参阅 为集群创建 IAM OIDC 提供商 (p. 396)。
- 与 IAM 角色关联的现有 Kubernetes 服务账户。必须使用 IAM 角色的 Amazon 资源名称（ARN）注释服务账户。该角色必须具有关联的 IAM policy，其中包含您希望您的容器组（pod）必须具有的权限，以便使用 Amazon Web Services。有关如何创建和配置服务账户和角色的更多信息，请参阅 the section called “配置角色和服务账户” (p. 397)。
- 您的设备或 Amazon CloudShell 上安装并配置了 `2.7.21` 版或更高版本，或 `1.25.46` 版或更高版本的 Amazon CLI。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的安装、更新和卸载 Amazon CLI 以及使用 `aws configure` 进行快速配置。
- 您的设备或 Amazon CloudShell 上安装了 `kubectl` 命令行工具。该版本可以与集群的 Kubernetes 版本相同，或者最多早于或晚于该版本一个次要版本。例如，如果您的集群版本为 `1.22`，则可以将 `kubectl` 版本 `1.21`、`1.22` 或 `1.23` 用于它。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl` (p. 4)。
- 包含集群配置的现有 `kubectl config` 文件。要创建 `kubectl config` 文件，请参阅 为 Amazon EKS 创建 `kubeconfig` (p. 365)。

配置容器组（pod）以使用服务账户

1. 创建 Kubernetes 部署清单。使用与您的集群版本对应的以下命令。此命令会创建一个部署清单，您可以部署 pod 以确认该清单的配置。pod 使用现有的 Kubernetes 服务账户。必须正确配置服务账户。有关更多信息，请参阅the section called “配置角色和服务账户” (p. 397)。将*###*替换为您自己的值。

- 1.19 及更高版本的集群

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:1.21
EOF
```

- 1.18 集群 – 将 *1337* 替换为任何有效的组 ID。预设情况下，只有以 `root` 身份运行的容器才具有正确的文件系统权限来读取 Web 身份令牌文件。您可以通过使容器以 `root` 身份运行或通过为清单中的容器提供以下安全上下文，以提供这些权限。有关为 pods 设置安全上下文的含义的更多信息，请参阅 Kubernetes 文档中的为容器组（pod）或容器配置安全上下文。

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:1.21
      securityContext:
        fsGroup: 1337
EOF
```

2. 将清单部署到集群。

```
kubectl apply -f my-deployment.yaml
```

3. 确认您的 pod 具有所需的环境变量。

   a. 按上一步部署后，查看部署的 pods。

   ```
   kubectl get pods | grep my-app
   ```

   输出示例如下。

```
my-app-6f4dfff6cb-76cv9    1/1     Running    0          3m28s
```

b. 查看 pod 使用的 IAM 角色的 ARN。

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
```

输出示例如下。

```
AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/my-role
```

角色 ARN 必须与您用于注释现有服务账户的角色 ARN 匹配。有注释服务账户的更多信息，请参阅the section called “配置角色和服务账户” (p. 397)。

c. 确认 pod 具有 Web 身份令牌文件挂载。

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_WEB_IDENTITY_TOKEN_FILE:
```

输出示例如下。

```
AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com.cn/serviceaccount/
token
```

`kubelet` 代表 pod 请求并存储令牌。默认情况下，如果令牌早于其总生存时间的 80%，或者令牌大于 24 小时，则 `kubelet` 会刷新令牌。您可以使用 pod 规范中的设置修改任何账户（默认服务账户除外）的过期期限。有关更多信息，请参阅 Kubernetes 文档中的服务账户令牌卷预测。

集群上的 Amazon EKS 容器组（pod）身份 Webhook 监控使用具有以下注释的服务账户的 pods：

```
eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/my-role
```

Webhook 将之前的环境变量应用于这些容器组（pod）。您的集群无需使用 Webhook 来配置环境变量和令牌文件挂载。您可以手动配置 pods 以包含这些环境变量。Amazon 开发工具包支持的版本 (p. 408)首先在凭证链提供商中查找这些环境变量。角色凭证用于满足这些标准的 pods。

4. 确认您的 pods 可以使用您在附加到角色的 IAM policy 中分配的权限与 Amazon Web Services 进行交互。

   Note

   当 pod 使用来自与服务账户关联的 IAM 角色的 Amazon 凭证时，该 pod 容器中的 Amazon CLI 或其他开发工具包仅使用该角色提供的凭证。如果您不限制对提供给 Amazon EKS 节点 IAM 角色 (p. 427) 的凭证的访问，pod 仍然可以访问这些凭证。有关更多信息，请参阅限制对分配给工作节点的实例配置文件的访问。

   如果您的 pods 无法按预期与服务进行交互，请完成以下步骤以确认所有配置都正确。

   a. 确认您的容器组（pod）使用支持通过 OpenID Connect Web 身份令牌文件代入 IAM 角色的 Amazon 开发工具包版本。有关更多信息，请参阅the section called “支持的开发工具包” (p. 408)。

   b. 确认部署正在使用服务账户。

```
kubectl describe deployment my-app | grep "Service Account"
```

输出示例如下。

```
Service Account:  my-service-account
```

c. 如果您的 pods 仍然无法访问服务，请查看 the section called “配置角色和服务账户” (p. 397) 中描述的步骤 (p. 400)，以确认您的角色和服务账户的配置都正确。

配置服务账户的 Amazon Security Token Service 端点

如果您将 Kubernetes 服务账户与 服务账户的 IAM 角色 (p. 395) 结合使用，若您的集群和平台版本相同，或者晚于下表中所列的版本，则您可以配置服务账户使用的 Amazon Security Token Service 端点类型。如果您的 Kubernetes 或平台版本早于表中列出的版本，则您的服务账户只能使用全局端点。

Kubernetes 版本	平台版本	默认端点类型
1.23	eks.1	区域性
1.22	eks.1	区域性
1.21	eks.3	服务全球
1.20	eks.3	服务全球
1.19	eks.7	服务全球
1.18	eks.9	服务全球

Amazon 建议使用区域 Amazon STS 端点而不是全局端点。这可以减少延迟，提供内置冗余并提高会话令牌的有效性。Amazon Security Token Service 必须在 pod 运行的 Amazon Web Services 区域 中是活动的。此外，您的应用程序必须内置冗余功能，以便在该 Amazon Web Services 区域 的服务出现故障时选择其他 Amazon Web Services 区域。有关更多信息，请参阅 IAM 用户指南中的在 Amazon Web Services 区域中管理 Amazon STS。

先决条件

- 现有集群。如果还没有，可以使用 开始使用 Amazon EKS (p. 4) 指南之一创建一个。
- 集群的现有 IAM OIDC 提供商。有关更多信息，请参阅为集群创建 IAM OIDC 提供商 (p. 396)。
- 现有的 Kubernetes 服务账户配置为与适用于服务账户的 Amazon EKS IAM (p. 395) 功能结合使用。

要配置 Kubernetes 服务账户使用的端点类型

以下示例全部使用 Amazon VPC CNI 插件 (p. 243)所使用的 aws-node Kubernetes 服务账户。您可以将 *example values* 替换为您自己的服务账户、pods、命名空间和其他资源。

1. 选择一个使用要更改其端点的服务账户的容器组（pod）。确定容器组（pod）在其中运行的 Amazon Web Services 区域。将 *aws-node-6mfgv* 替换为您的 pod 名称，并将 *kube-system* 替换为您的 pod 的命名空间。

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
   ```

   输出示例如下。

   ```
   ip-192-168-79-166.cn-north-1/192.168.79.166
   ```

   在之前的输出中，pod 在 Amazon Web Services 区域 cn-north-1 中的节点上运行。

2. 确定容器组（pod）的服务账户使用的端点类型。

```
kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

输出示例如下。

```
AWS_STS_REGIONAL_ENDPOINTS: regional
```

如果当前端点是全局端点，则输出中将返回 `global`。如果没有返回输出，则默认端点类型正在使用中且尚未被覆盖。

3. 如果您的集群或平台版本与表中列出的版本相同或晚于该版本，则可以使用以下命令之一将服务账户使用的端点类型从默认类型更改为其他类型。将 *aws-node* 替换为服务账户的服务账户的名称，并将 *kube-system* 替换为服务账户的命名空间。

- 如果您的默认或当前端点类型为全局端点类型，并且您想将其更改为区域性端点：

```
kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
```

如果您使用 服务账户的 IAM 角色 (p. 395) 以在 pods 的容器中运行的应用程序中生成预签名 S3 URL，则区域端点的 URL 格式与以下示例类似：

```
https://bucket.s3.cn-north-1.amazonaws.com.cn/path?...&X-Amz-Credential=your-access-key-id/date/cn-north-1/s3/aws4_request&...
```

- 如果您的默认或当前端点类型为区域性端点类型，并且您想将其更改为全局性端点：

```
kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
```

如果您的应用程序明确向 Amazon STS 全局端点提出请求，且您不会覆盖在 Amazon EKS `1.22` 和更高版本集群中使用区域端点的默认行为，则请求将失败，并出现错误。有关更多信息，请参阅容器组（pod）容器收到以下错误：`An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region.` (p. 489)。

如果您使用 服务账户的 IAM 角色 (p. 395) 以在 pods 的容器中运行的应用程序中生成预签名 S3 URL，则全局端点的 URL 格式与以下示例类似：

```
https://bucket.s3.amazonaws.com.cn/path?...&X-Amz-Credential=your-access-key-id/date/cn-north-1/s3/aws4_request&...
```

如果您的自动化需要特定格式的预签名 URL，或者如果您的应用程序或使用预签名 URL 的下游依赖关系对目标 Amazon Web Services 区域 有期望，则进行必要的更改以使用适当的 Amazon STS 端点。

4. 删除并重新创建任何与服务账户关联的现有 pods，以应用凭证环境变量。变异 Webhook 不将其应用到已经在运行的 pods。您可以将 *pods*、*kube-system* 和 *-l k8s-app=aws-node* 替换为您设置注释的 pods 的信息。

```
kubectl delete pods -n kube-system -l k8s-app=aws-node
```

5. 确认 pods 已全部重新启动。

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

6. 查看其中一个 pods 的环境变量。验证 AWS_STS_REGIONAL_ENDPOINTS 值是您在上一步中将其设置的值。

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

输出示例如下。

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

跨账户 IAM 权限

您可以通过从其他账户的集群创建身份提供商或者使用链接的 AssumeRole 操作，配置跨账户 IAM 权限。在以下示例中，账户 A 拥有一个支持服务账户的 IAM 角色的 Amazon EKS 集群。在该集群上运行的 Pods 必须代入来自账户 B 的 IAM 权限。

Example 从其他账户的集群创建身份提供商

Example

在此示例中，账户 A 从其集群向账户 B 提供 OpenID Connect（OIDC）颁发者 URL。账户 B 遵循为集群创建 IAM OIDC 提供商 (p. 396)和配置 Kubernetes 服务账户以代入 IAM 角色 (p. 397)中的说明，并使用来自账户 A 集群的 OIDC 发布者 URL。然后，集群管理员注释账户 A 的集群中的服务账户以使用来自账户 B 的角色（*444455556666*）。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::444455556666:role/account-b-role
```

Example 使用链接 **AssumeRole** 操作

Example

在此示例中，账户 B 创建一个 IAM policy，其中包括授予账户 A 集群中 pods 的权限。账户 B（*444455556666*）将该策略附加到具有信任关系的 IAM 角色，允许对账户 A（*111122223333*）的 AssumeRole 权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

账户 A 创建具有信任策略的角色，该策略从使用集群的 OIDC 颁发者地址创建的身份提供商处获取凭证。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com.cn/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity"
    }
  ]
}
```

账户 A 将策略附加到具有以下权限的角色，以代入账户 B 创建的角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::444455556666:role/account-b-role"
        }
    ]
}
```

pods 的应用程序代码使用两个配置文件代入账户 B 的角色：`account_b_role` 和 `account_a_role`。`account_b_role` 配置文件使用 `account_a_role` 配置文件作为其源。对于 Amazon CLI，`~/.aws/config` 文件类似于以下内容。

```
[profile account_b_role]
source_profile = account_a_role
role_arn=arn:aws:iam::444455556666:role/account-b-role

[profile account_a_role]
web_identity_token_file = /var/run/secrets/eks.amazonaws.com/serviceaccount/token
role_arn=arn:aws:iam::111122223333:role/account-a-role
```

要为其他 Amazon 开发工具包指定链接的配置文件，请参考所使用开发工具包的文档。有关更多信息，请参阅用于在 Amazon 上进行构建的工具。

使用支持的 Amazon 开发工具包

pods 中的容器必须使用支持通过 OpenID Connect Web 身份令牌文件代入 IAM 角色的 Amazon 开发工具包版本。AmazonLinux 分发版本的程序包管理器中包含的开发工具包可能不是最新的，无法支持此功能。请确保为您的开发工具包使用以下版本或更高版本：

- Java（版本 2）– 2.10.11
- Java – 1.11.704
- Go – 1.23.13
- Python (Boto3) – 1.9.220
- Python (botocore) – 1.12.200
- Amazon CLI - 1.16.232
- Node – 3.27.0
- Ruby – 3.58.0
- C++ – 1.7.174

- .NET – 3.3.659.1
- PHP – 3.110.7

许多流行的 Kubernetes 附加组件，如 Cluster Autoscaler 和 安装 Amazon Load Balancer Controller 附加组件 (p. 287) 都支持服务账户的 IAM 角色。Amazon VPC CNI plugin for Kubernetes 也支持服务账户的 IAM 角色。

为了确保您使用的是受支持的开发工具包，请在构建容器时按照用于在 Amazon 上进行构建的工具中针对您的首选开发工具包的安装说明操作。

适用于 Amazon EKS 的身份和访问管理

Amazon Identity and Access Management (IAM) 是一项 Amazon Web Service，可以帮助管理员安全地控制对 Amazon 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）使用 Amazon EKS 资源。IAM 是一项无需额外费用即可使用的 Amazon Web Service。

Audience

如何使用 Amazon Identity and Access Management (IAM) 因您在 Amazon EKS 中执行的操作而异。

服务用户 – 如果您使用 Amazon EKS 服务来执行任务，则您的管理员会为您提供所需的凭证和权限。随着您使用更多 Amazon EKS 功能来完成工作，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 Amazon EKS 中的功能，请参阅 Amazon EKS 身份和访问问题排查 (p. 451)。

服务管理员 – 如果您在公司负责管理 Amazon EKS 资源，您可能对 Amazon EKS 具有完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon EKS 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon EKS 搭配使用的更多信息，请参阅 Amazon EKS 如何与 IAM 配合使用 (p. 412)。

IAM 管理员 – 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Amazon EKS 的访问的详细信息。要查看您可在 IAM 中使用的 Amazon EKS 基于身份的策略示例，请参阅 Amazon EKS 基于身份的策略示例 (p. 415)。

使用身份进行身份验证

身份验证是您使用身份凭证登录 Amazon 的方法。有关使用Amazon Web Services Management Console登录的更多信息，请参阅 IAM 用户指南中的以 IAM 用户或根用户身份登录Amazon Web Services Management Console。

您必须作为Amazon Web Services 账户根用户、IAM 用户或代入 IAM 角色以进行身份验证（登录到 Amazon）。您还可以使用公司的单一登录身份验证方法，甚至使用 Google 或 Facebook 登录。在这些情况下，您的管理员以前使用 IAM 角色设置了联合身份验证。在您使用来自其它公司的凭证访问 Amazon 时，您间接地代入了角色。

要直接登录到Amazon Web Services Management Console，请将密码与根用户电子邮件地址或 IAM 用户名一起使用。您可以使用根用户或 IAM 用户访问密钥以编程方式访问 Amazon。Amazon 提供了 SDK 和命令行工具，可使用您的凭证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。使用 Signature Version 4（用于对入站 API 请求进行验证的协议）完成此操作。有关验证请求的更多信息，请参阅《Amazon 一般参考》中的 Signature Version 4 签名流程。

无论使用何种身份验证方法，您可能还需要提供其它安全信息。例如，Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的在 Amazon 中使用多重身份验证 (MFA)。

Amazon Web Services 账户根用户

当您创建 Amazon Web Services 账户时，最初使用的是一个对账户中所有 Amazon Web Services和资源拥有完全访问权限的登录身份。此身份称为 Amazon Web Services 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《Amazon 一般参考》中的需要根用户凭证的任务。

IAM 用户和组

IAM 用户是Amazon Web Services 账户内对某个人员或应用程序具有特定权限的一个身份。IAM 用户可能具有长期凭证，例如用户名和密码或一组访问密钥。要了解如何生成访问密钥，请参阅 IAM 用户指南 中的管理 IAM 用户的访问密钥。为 IAM 用户生成访问密钥时，请确保查看并安全保存密钥对。您以后无法找回秘密访问密钥，而是必须生成新的访问密钥对。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是Amazon Web Services 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 Amazon Web Services Management Console 中暂时代入 IAM 角色。您可以调用 Amazon CLI 或 Amazon API 操作或使用自定义 URL 以代入角色。有关使用角色的方法的更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 临时 IAM 用户权限 – IAM 用户可以代入 IAM 角色，以暂时获得不同的权限以执行特定的任务。
- 联合用户访问 – 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、Web 身份提供商或 IAM Identity Center 身份存储的现有身份。这些身份称为联合身份。要向联合身份分配权限，您可以创建角色并为角色定义权限。当外部身份进行身份验证时，该身份将与角色相关联并被授予其定义的权限。如果您使用 IAM Identity Center，则需要配置权限集。IAM Identity Center 将权限集与 IAM 中的角色相关联，以控制您的身份在进行身份验证后可以访问的内容。有关身份联合验证的更多信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。有关 IAM Identity Center 的更多信息，请参阅《Amazon IAM Identity Center (successor to Amazon Single Sign-On) 用户指南》中的什么是 IAM Identity Center？。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 Amazon Web Services，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 – 某些 Amazon Web Services使用其它 Amazon Web Services中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service（Amazon S3）中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 主体权限 – 当您使用 IAM 用户或角色在 Amazon 中执行操作时，您将被视为主体。策略向主体授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否需要策略中的其他相关操作，请参阅服务授权参考中的 Amazon Elastic Kubernetes Service 的操作、资源和条件键。
  - 服务角色 – 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的创建向 Amazon Web Service 委派权限的角色。

- 服务相关角色 – 服务相关角色是与 Amazon Web Service关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。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 用户指南中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 Amazon 身份或资源，以控制 Amazon 中的访问。策略是 Amazon 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 Amazon 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南 中的 JSON 策略概览。

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

每个 IAM 实体（用户或角色）最初没有任何权限。原定设置情况下，用户什么都不能做，甚至不能更改他们自己的密码。要为用户授予执行某些操作的权限，管理员必须将权限策略附加到用户。或者，管理员可以将用户添加到具有预期权限的组中。当管理员为某个组授予访问权限时，该组内的全部用户都会获得这些访问权限。

IAM policy 定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管式策略是可以附加到Amazon Web Services 账户中的多个用户、组和角色的独立策略。托管式策略包括 Amazon 托管式策略和客户托管式策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南 中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合身份用户或 Amazon Web Services。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 Amazon 托管式策略。

访问控制列表 (ACL)

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Simple Storage Service（Amazon S3）、Amazon WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的访问控制列表 (ACL) 概览。

其它策略类型

Amazon 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体的基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 Amazon Organizations 中的最大权限。Amazon Organizations 服务可以分组和集中管理您的企业拥有的多个 Amazon Web Services 账户 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体的权限，包括每个 Amazon Web Services 账户 根用户。有关 Organizations 和 SCP 的更多信息，请参阅 Amazon Organizations 用户指南中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 Amazon 如何确定在涉及多种策略类型时是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon EKS 如何与 IAM 配合使用

在使用 IAM 管理对 Amazon EKS 的访问权限之前，您应该了解哪些 IAM 功能可用于 Amazon EKS。要大致了解 Amazon EKS 和其他Amazon服务如何与 IAM 一起使用，请参阅 IAM 用户指南中的与 IAM 一起使用的Amazon服务。

主题

- Amazon EKS 基于身份的策略 (p. 412)
- Amazon EKS 基于资源的策略 (p. 414)
- 基于 Amazon EKS 标签的授权 (p. 414)
- Amazon EKS IAM 角色 (p. 414)

Amazon EKS 基于身份的策略

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。Amazon EKS 支持特定的操作、资源和条件键。要了解您在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南中的 IAM JSON 策略元素参考。

操作

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 Amazon API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

Amazon EKS 中的策略操作在操作前面使用以下前缀：`eks:`。例如，要授予某人获取关于 Amazon EKS 集群的描述性信息的权限，您应将 `DescribeCluster` 操作纳入其策略中。策略语句必须包含 `Action` 或 `NotAction` 元素。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

```
"Action": ["eks:action1", "eks:action2"]
```

您也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，包括以下操作：

```
"Action": "eks:Describe*"
```

要查看 Amazon EKS 操作的列表，请参阅《服务授权参考》中的 Amazon Elastic Kubernetes Service 定义的操作。

资源

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon Resource Name (ARN) 指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

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

Amazon EKS 集群资源具有以下 ARN。

```
arn:aws:eks:region-code:account-id:cluster/cluster-name
```

有关 ARN 格式的更多信息，请参阅 Amazon 资源名称 (ARN) 和 Amazon 服务命名空间。

例如，要在语句中指定名为 *my-cluster* 的集群，请使用以下 ARN：

```
"Resource": "arn:aws:eks:region-code:111122223333:cluster/my-cluster"
```

要指定属于特定账户和 Amazon Web Services 区域 的所有集群，请使用通配符 (*)：

```
"Resource": "arn:aws:eks:region-code:111122223333:cluster/*"
```

无法对特定资源执行某些 Amazon EKS 操作，例如用于创建资源的操作。在这些情况下，您必须使用通配符 (*)。

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

要查看 Amazon EKS 资源类型及其 ARN 的列表，请参阅《服务授权参考》中的 Amazon Elastic Kubernetes Service 定义的资源。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 Amazon Elastic Kubernetes Service 定义的操作。

条件键

Amazon EKS 定义了自己的一组条件键，还支持使用一些全局条件键。要查看所有 Amazon 全局条件键，请参阅《IAM 用户指南》中的 Amazon 全局条件上下文键。

在将 OpenID Connect 提供商与集群关联时，您可以设置条件键。有关更多信息，请参阅示例 IAM policy (p. 364)。

所有 Amazon EC2 操作都支持 `aws:RequestedRegion` 和 `ec2:Region` 条件键。有关更多信息，请参阅示例：限制对特定 Amazon Web Services 区域的访问。

有关 Amazon EKS 条件键的列表，请参阅 《服务授权参考》中的 Amazon Elastic Kubernetes Service 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon Elastic Kubernetes Service 定义的操作。

示例

要查看 Amazon EKS 基于身份的策略的示例，请参阅 Amazon EKS 基于身份的策略示例 (p. 415)。

当您创建 Amazon EKS 集群时，将为 Amazon Identity and Access Management (IAM) 实体用户或角色，如创建集群的联合身份用户，自动授予 Amazon EKS 控制层面中基于集群角色的访问控制 (RBAC) 配置中的 `system:masters` 权限。此 IAM 实体不会显示在任何可见配置中，因此请确保跟踪最初创建集群的 IAM 实体。要授予其他 Amazon 用户或角色与集群进行交互的能力，必须编辑 Kubernetes 中的 `aws-auth ConfigMap`，创建 Kubernetes `rolebinding` 或 `clusterrolebinding`，名为 `aws-auth ConfigMap` 中指定的 `group`。

有关使用 ConfigMap 的更多信息，请参阅 让 IAM 用户和角色有权访问您的集群 (p. 355)。

Amazon EKS 基于资源的策略

Amazon EKS 不支持基于资源的策略。

基于 Amazon EKS 标签的授权

您可以将标签附加到 Amazon EKS 资源，或者在请求中将标签传递给 Amazon EKS。要基于标签控制访问，您需要使用 `aws:ResourceTag/`*`key-name`*、`aws:RequestTag/`*`key-name`* 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。有关标记 Amazon EKS 资源的更多信息，请参阅 为您的 Amazon EKS 资源添加标签 (p. 385)。想要了解您可以将条件键中的标签用于哪些操作，请参阅服务授权参考中的 Amazon EKS 定义的操作。

Amazon EKS IAM 角色

IAM 角色是 Amazon 账户中具有特定权限的实体。

将临时凭证用于 Amazon EKS

您可以使用临时凭证进行联合身份登录，担任 IAM 角色或担任跨账户角色。您可以通过调用 Amazon STS API 操作（如 AssumeRole 或 GetFederationToken）获得临时安全凭证。

Amazon EKS 支持使用临时凭证。

服务相关角色

服务相关角色允许 Amazon 服务访问其它服务中的资源以代表您完成操作。服务相关角色显示在您的 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

Amazon EKS 支持服务相关角色。有关创建或管理 Amazon EKS 服务相关角色的详细信息，请参阅 对 Amazon EKS 使用服务相关角色 (p. 418)。

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其它服务中的资源以代表您完成操作。服务角色显示在您的 IAM 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

Amazon EKS 支持服务角色。有关更多信息，请参阅 Amazon EKS 集群 IAM 角色 (p. 425) 和 Amazon EKS 节点 IAM 角色 (p. 427)。

在 Amazon EKS 中选择 IAM 角色

当您在 Amazon EKS 中创建集群资源时，您必须选择一个角色以允许 Amazon EKS 代表您访问一些其他 Amazon资源。如果您之前创建了一个服务角色，Amazon EKS 会为您提供一个角色列表供您选择。请务必选择一个附带 Amazon EKS 托管策略的角色。有关更多信息，请参阅 检查现有集群角色 (p. 425) 和 检查现有节点角色 (p. 427)。

Amazon EKS 基于身份的策略示例

预设情况下，IAM 用户和角色没有创建或修改 Amazon EKS 资源的权限。它们还无法使用 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 执行任务。IAM 管理员必须创建 IAM policy，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅 IAM 用户指南中的在 JSON 选项卡上创建策略。

创建 Amazon EKS 集群时，将为 Amazon Identity and Access Management (IAM) 实体用户或角色，如创建集群的联合身份用户，自动授予 Amazon EKS 控制层面中基于集群角色的访问控制 (RBAC) 配置中的 `system:masters` 权限。此 IAM 实体不会显示在任何可见配置中，因此请确保跟踪最初创建集群的 IAM 实体。要授予其他 Amazon 用户或角色与集群进行交互的能力，必须编辑 Kubernetes 中的 `aws-auth ConfigMap`，创建 Kubernetes `rolebinding` 或 `clusterrolebinding`，名为 `aws-auth ConfigMap` 中指定的 `group`。

有关使用 ConfigMap 的更多信息，请参阅 让 IAM 用户和角色有权访问您的集群 (p. 355)。

主题

- 策略最佳实践 (p. 415)
- 使用 Amazon EKS 控制台 (p. 416)
- 允许用户查看他们自己的权限 (p. 417)
- 创建 Kubernetes 集群 (p. 417)
- 更新 Kubernetes 集群 (p. 418)
- 列出或描述所有集群 (p. 418)

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Amazon EKS 资源。这些操作可能会使 Amazon Web Services 账户 产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- Amazon 托管策略及转向最低权限许可入门 - 要开始向用户和工作负载授予权限，请使用 Amazon 托管策略来为许多常见使用场景授予权限。您可以在 Amazon Web Services 账户 中找到这些策略。我们建议通过定义特定于您的使用场景的 Amazon 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管策略或工作职能的 Amazon 托管策略。

- 应用最低权限 – 在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定 Amazon Web Service（例如 Amazon CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM policy语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- 需要多重身份验证 (MFA) – 如果您的账户需要 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 Amazon EKS 控制台

要访问 Amazon EKS 控制台，IAM 实体（如用户或角色）必须具有一组最低的权限。这些权限允许实体列出和查看有关您的Amazon账户中的 Amazon EKS 资源的详细信息。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体，控制台将无法按预期正常运行。

为确保 IAM 实体仍可以使用 Amazon EKS 控制台，请用唯一名称创建一个策略，如 `AmazonEKSAdminPolicy`。将策略附加到实体。有关更多信息，请参阅 IAM 用户指南中的为用户添加权限。

> **Important**
>
> 以下示例策略将允许您查看控制台中 Configuration（配置）选项卡上的信息。要在 Amazon Web Services Management Console 中的 Overview（概述）和 Resources（资源）选项卡上查看信息，该实体还需要 Kubernetes 权限。有关更多信息，请参阅所需权限 (p. 458)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

对于只需要调用 Amazon CLI 或 Amazon API 的用户，无需为其提供最低控制台权限。相反，只允许访问与您尝试执行的 API 操作相匹配的操作。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上完成此操作或者以编程方式使用 Amazon CLI 或 Amazon API 所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

创建 Kubernetes 集群

此示例策略包含在 *cn-north-1* Amazon Web Services 区域 中创建名为 *my-cluster* 的 Amazon EKS 集群所需的最低权限。您可以将 Amazon Web Services 区域 替换为要在其中创建集群的 Amazon Web Services 区域。如果在 Amazon Web Services Management Console 中看到警告 The actions in your policy do not support resource-level permissions and require you to choose **All resources**（策略中的操作不支持资源级权限，必须选择 All resources），您可以放心忽略该警告。如果账户已经有 *AWSServiceRoleForAmazonEKS* 角色，您可以从策略中删除 iam:CreateServiceLinkedRole 操作。如果在账户中创建过 Amazon EKS 集群，则此角色已经存在，除非您已将其删除。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:cn-north-1:111122223333:cluster/my-cluster"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
eks.amazonaws.com/AWSServiceRoleForAmazonEKS",
```

```
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iam:AWSServiceName": "eks"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        }
    ]
}
```

更新 Kubernetes 集群

此示例策略包含在 cn-north-1 Amazon Web Services 区域 中更新名为 *my-cluster* 的集群所需的最低权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:UpdateClusterVersion",
            "Resource": "arn:aws:eks:cn-north-1:111122223333:cluster/my-cluster"
        }
    ]
}
```

列出或描述所有集群

此示例策略包含列出和描述账户中所有集群所需的最低权限。IAM 用户或角色必须能够列出和描述集群，才能使用 `update-kubeconfig` Amazon CLI 命令。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

对 Amazon EKS 使用服务相关角色

Amazon Elastic Kubernetes Service 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 Amazon EKS 直接相关。服务相关角色是由 Amazon EKS 预定义的，并包含服务代表您调用其他Amazon服务所需的所有权限。

主题

- 使用 Amazon EKS 集群的角色 (p. 419)

- 使用 Amazon EKS 节点组的角色 (p. 420)
- 使用 Amazon EKS Fargate 配置文件的角色 (p. 422)
- 使用角色将 Kubernetes 集群连接到 Amazon EKS (p. 423)

使用 Amazon EKS 集群的角色

Amazon Elastic Kubernetes Service 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 Amazon EKS 直接相关。服务相关角色是由 Amazon EKS 预定义的，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地设置 Amazon EKS，因为您不必手动添加必要的权限。Amazon EKS 定义其服务相关角色的权限，除非另外定义，否则只有 Amazon EKS 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。

只有在首先删除相关资源后，才能删除服务相关角色。这将保护您的 Amazon EKS 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 Role（服务相关角色）列中为 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。

Amazon EKS 的服务相关角色权限

Amazon EKS 使用名为 `AWSServiceRoleForAmazonEKS` 的服务相关角色 – 该角色允许 Amazon EKS 管理您账户中的集群。附加的策略允许角色管理以下资源：网络接口、安全组、日志和 VPC。

> **Note**
>
> `AWSServiceRoleForAmazonEKS` 服务相关角色不同于创建集群所需的角色。有关更多信息，请参阅Amazon EKS 集群 IAM 角色 (p. 425)。

`AWSServiceRoleForAmazonEKS` 服务相关角色信任以下服务代入该角色：

- `eks.amazonaws.com`

角色权限策略允许 Amazon EKS 对指定资源完成以下操作：

- `AmazonEKSServiceRolePolicy`

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 Amazon EKS 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、Amazon CLI 或 Amazon API 中创建集群时，Amazon EKS 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。当您创建集群时，Amazon EKS 将再次为您创建服务相关角色。

为 Amazon EKS 编辑服务相关角色

Amazon EKS 不允许您编辑 `AWSServiceRoleForAmazonEKS` 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。

删除适用于 Amazon EKS 的服务相关角色

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，然后才能手动删除它。

清除服务相关角色

必须先删除服务相关角色使用的所有资源，然后才能使用 IAM 删除该角色。

> Note
>
> 如果当您试图删除资源时 Amazon EKS 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。

删除 **`AWSServiceRoleForAmazonEKS`** 角色使用的 Amazon EKS 资源。

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择集群。
3. 如果您的集群具有任何节点组或 Fargate 配置文件，则必须先将其删除，然后才能删除该集群。有关更多信息，请参阅 删除托管节点组 (p. 105) 和 删除 Fargate 配置文件 (p. 134)。
4. 在 Clusters（集群）页面上，选择要删除的集群，然后选择 Delete（删除）。
5. 在删除确认窗口中键入集群的名称，然后选择 Delete（删除）。
6. 对您账户中的任何其他集群重复此过程。等待所有删除操作完成。

手动删除 服务相关角色

使用 IAM 控制台、Amazon CLI 或 Amazon API 删除 `AWSServiceRoleForAmazonEKS` 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。

Amazon EKS 服务相关角色支持的区域

Amazon EKS 支持在该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 EKS 服务端点和配额。

使用 Amazon EKS 节点组的角色

Amazon EKS 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 Amazon EKS 直接相关。服务相关角色是由 Amazon EKS 预定义的，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地设置 Amazon EKS，因为您不必手动添加必要的权限。Amazon EKS 定义其服务相关角色的权限，除非另外定义，否则只有 Amazon EKS 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。

只有在首先删除相关资源后，才能删除服务相关角色。这将保护您的 Amazon EKS 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 Role（服务相关角色）列中为 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。

Amazon EKS 的服务相关角色权限

Amazon EKS 使用名为 `AWSServiceRoleForAmazonEKSNodegroup` 的服务相关角色 – 该角色允许 Amazon EKS 管理您账户中的节点组。附加的策略允许角色管理以下资源：Auto Scaling 组、安全组、启动模板和 IAM 实例配置文件。

AWSServiceRoleForAmazonEKSNodegroup 服务相关角色信任以下服务代入该角色：

- eks-nodegroup.amazonaws.com

角色权限策略允许 Amazon EKS 对指定资源完成以下操作：

- AWSServiceRoleForAmazonEKSNodegroup

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 Amazon EKS 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、Amazon CLI 或 Amazon API 中创建节点组时，Amazon EKS 会为您创建服务相关角色。

> Important
>
> 如果您在其他使用此角色支持的功能的服务中完成某个操作，此服务相关角色可以出现在您的账户中。如果在 2017 年 1 月 1 日（从此时开始支持服务相关角色）之前已使用 Amazon EKS 服务，则 Amazon EKS 已在您的账户中创建了 AWSServiceRoleForAmazonEKSNodegroup 角色。要了解更多信息，请参阅我的 IAM 账户中出现新角色。

在 Amazon EKS (Amazon API) 中创建服务相关角色

无需手动创建服务相关角色。当您在 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 中创建托管节点组时，Amazon EKS 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。当您创建另一个托管节点组时，Amazon EKS 再次为您创建服务相关角色。

为 Amazon EKS 编辑服务相关角色

Amazon EKS 不允许您编辑 AWSServiceRoleForAmazonEKSNodegroup 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。

删除适用于 Amazon EKS 的服务相关角色

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，然后才能手动删除它。

清除服务相关角色

必须先删除服务相关角色使用的所有资源，然后才能使用 IAM 删除该角色。

> Note
>
> 如果当您试图删除资源时 Amazon EKS 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。

删除 **AWSServiceRoleForAmazonEKSNodegroup** 角色使用的 Amazon EKS 资源。

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择集群。
3. 选择 Compute（计算）选项卡。
4. 在 Node groups（节点组）部分中，选择要删除的节点组。

5. 在删除确认窗口中键入节点组的名称，然后选择 Delete（删除）。
6. 对集群中的任何其他节点组重复此过程。等待所有删除操作完成。

手动删除 服务相关角色

使用 IAM 控制台、Amazon CLI 或 Amazon API 删除 `AWSServiceRoleForAmazonEKSNodegroup` 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。

Amazon EKS 服务相关角色支持的区域

Amazon EKS 支持在该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 EKS 服务端点和配额。

使用 Amazon EKS Fargate 配置文件的角色

Amazon EKS 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 Amazon EKS 直接相关。服务相关角色是由 Amazon EKS 预定义的，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地设置 Amazon EKS，因为您不必手动添加必要的权限。Amazon EKS 定义其服务相关角色的权限，除非另外定义，否则只有 Amazon EKS 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。

只有在首先删除相关资源后，才能删除服务相关角色。这将保护您的 Amazon EKS 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 Role（服务相关角色）列中为 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。

Amazon EKS 的服务相关角色权限

Amazon EKS 使用名为 `AWSServiceRoleForAmazonEKSForFargate` 的服务相关角色 – 该角色允许 Amazon EKS Fargate 配置 Fargate Pod 所需的 VPC 网络。附加的策略允许角色创建和删除弹性网络接口，并描述弹性网络接口和资源。

`AWSServiceRoleForAmazonEKSForFargate` 服务相关角色信任以下服务代入该角色：

- `eks-fargate.amazonaws.com`

角色权限策略允许 Amazon EKS 对指定资源完成以下操作：

- `AmazonEKSForFargateServiceRolePolicy`

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 Amazon EKS 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、Amazon CLI 或 Amazon API 中创建 Fargate 配置文件时，Amazon EKS 将为您创建服务相关角色。

> **Important**
>
> 如果您在其他使用此角色支持的功能的服务中完成某个操作，此服务相关角色可以出现在您的账户中。如果在 2019 年 12 月 13 日（从此时开始支持服务相关角色）之前已使用 Amazon EKS 服务，

则 Amazon EKS 已在您的账户中创建了 AWSServiceRoleForAmazonEKSForFargate 角色。要了解更多信息，请参阅我的 IAM 账户中出现新角色。

在 Amazon EKS (Amazon API) 中创建服务相关角色

无需手动创建服务相关角色。当您在 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 中创建 Fargate 配置文件时，Amazon EKS 将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。当您创建另一个托管节点组时，Amazon EKS 再次为您创建服务相关角色。

为 Amazon EKS 编辑服务相关角色

Amazon EKS 不允许您编辑 `AWSServiceRoleForAmazonEKSForFargate` 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。

删除适用于 Amazon EKS 的服务相关角色

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，然后才能手动删除它。

清除服务相关角色

必须先删除服务相关角色使用的所有资源，然后才能使用 IAM 删除该角色。

> **Note**
>
> 如果当您试图删除资源时 Amazon EKS 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。

删除 **`AWSServiceRoleForAmazonEKSForFargate`** 角色使用的 Amazon EKS 资源。

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
2. 在左侧导航窗格中，选择集群。
3. 在 Clusters (集群) 页面上，选择您的集群。
4. 选择 Compute（计算）选项卡。
5. 如果 Fargate profiles（Fargate 配置文件）部分中有任何 Fargate 配置文件，则分别选择每个配置文件，然后选择 Delete（删除）。
6. 在删除确认窗口中键入配置文件的名称，然后选择 Delete（删除）。
7. 对集群中以及您账户中任何其他集群的任何其他 Fargate 配置文件重复此过程。

手动删除 服务相关角色

使用 IAM 控制台、Amazon CLI 或 Amazon API 删除 AWSServiceRoleForAmazonEKSForFargate 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。

Amazon EKS 服务相关角色支持的区域

Amazon EKS 支持在该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 EKS 服务端点和配额。

使用角色将 Kubernetes 集群连接到 Amazon EKS

Amazon EKS 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 Amazon EKS 直接相关。服务相关角色是由 Amazon EKS 预定义的，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地设置 Amazon EKS，因为您不必手动添加必要的权限。Amazon EKS 定义其服务相关角色的权限，除非另外定义，否则只有 Amazon EKS 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。

只有在首先删除相关资源后，才能删除服务相关角色。这将保护您的 Amazon EKS 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 Role（服务相关角色）列中为 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。

Amazon EKS 的服务相关角色权限

Amazon EKS 使用名为 `AWSServiceRoleForAmazonEKSConnector` 的服务相关角色：该角色允许 Amazon EKS 连接 Kubernetes 集群。附加的策略允许角色管理必要的资源，以连接到注册的 Kubernetes 集群。

`AWSServiceRoleForAmazonEKSConnector` 服务相关角色信任以下服务代入该角色：

- `eks-connector.amazonaws.com`

角色权限策略允许 Amazon EKS 对指定资源完成以下操作：

- `AmazonEKSConnectorServiceRolePolicy`

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 Amazon EKS 创建服务相关角色

无需手动创建服务相关角色来连接集群。当您在 Amazon Web Services Management Console、Amazon CLI、`eksctl` 或 Amazon API 中连接集群时，Amazon EKS 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。连接集群时，Amazon EKS 将再次为您创建服务相关角色。

为 Amazon EKS 编辑服务相关角色

Amazon EKS 不允许您编辑 `AWSServiceRoleForAmazonEKSConnector` 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。

删除适用于 Amazon EKS 的服务相关角色

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，然后才能手动删除它。

清除服务相关角色

必须先删除服务相关角色使用的所有资源，然后才能使用 IAM 删除该角色。

> **Note**
>
> 如果当您试图删除资源时 Amazon EKS 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。

删除 **`AWSServiceRoleForAmazonEKSConnector`** 角色使用的 Amazon EKS 资源。

1. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。

2. 在左侧导航窗格中，选择集群。
3. 在 Clusters (集群) 页面上，选择您的集群。
4. 选择 Deregister（取消注册）选项卡，然后选择 Ok（确定）选项卡。

手动删除 服务相关角色

使用 IAM 控制台、Amazon CLI 或 Amazon API 删除 AWSServiceRoleForAmazonEKSConnector 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。

Amazon EKS 服务相关角色支持的区域

Amazon EKS 支持在该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 EKS 服务端点和配额。

Amazon EKS 集群 IAM 角色

Amazon EKS 托管的 Kubernetes 集群会代表您调用其他 Amazon 服务，以管理您用于该服务的资源。必须先使用以下 IAM policy 创建 IAM 角色，然后才能创建 Amazon EKS 集群：

- `AmazonEKSClusterPolicy`

> **Note**
>
> 在 2020 年 4 月 16 日之前，AmazonEKSServicePolicy 也是必需的，建议的名称是 `eksServiceRole`。有了 `AWSServiceRoleForAmazonEKS` 服务相关角色后，在 2020 年 4 月 16 日或之后创建的集群将不再需要该策略。

检查现有集群角色

可使用以下程序检查并确定您的账户是否已有 Amazon EKS 集群角色。

在 IAM 控制台中检查 **`eksClusterRole`**

1. 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. 在角色列表中搜索 `eksClusterRole`。如果不存在包含 `eksClusterRole` 的角色，请参阅 创建 Amazon EKS 集群角色 (p. 426) 来创建角色。如果包含 `eksClusterRole` 的角色确实存在，则选择角色以查看附加的策略。
4. 请选择权限。
5. 确保将 AmazonEKSClusterPolicy 托管策略附加到此角色。如果附加该策略，则将正确配置 Amazon EKS 集群角色。
6. 选择 Trust relationships（信任关系），然后选择 Edit trust policy（编辑信任策略）。
7. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel（取消）。如果信任关系不匹配，请将策略复制到 Edit trust policy（编辑信任策略）窗口并选择 Update policy（更新策略）。

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

```
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

创建 Amazon EKS 集群角色

要创建 IAM 角色，您可以使用 Amazon Web Services Management Console 或 Amazon CLI。

Amazon Web Services Management Console

在 IAM 控制台中创建 Amazon EKS 集群角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择 Roles，然后选择 Create role。
3. 在 Trusted entity type（受信任的实体类型）下，选择 Amazon service（Amazon 服务）。
4. 从 Use cases for other Amazon Web Services（其他 Amazon Web Services 的使用案例）下拉列表中，选择 EKS。
5. 为您的使用案例选择 EKS - Cluster（EKS - 集群），然后选择 Next（下一步）。
6. 在 Add permissions（添加权限）选项卡上，选择 Next（下一步）。
7. 对于 Role name（角色名称），请为角色输入唯一名称，例如 **eksClusterRole**。
8. 对于 Description（说明），请输入描述性文本，例如 **Amazon EKS - Cluster role**。
9. 选择 Create role（创建角色）。

Amazon CLI

1. 将以下内容复制到名为 *cluster-trust-policy.json* 的文件中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 创建角色。您可以将 ***eksClusterRole*** 替换为您选择的任何名称。

```
aws iam create-role \
  --role-name eksClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. 将所需的 IAM policy 附加到角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name eksClusterRole
```

Amazon EKS 节点 IAM 角色

Amazon EKS 节点 `kubelet` 守护进程代表您调用 Amazon API。节点通过 IAM 实例配置文件和关联的策略获得这些 API 调用的权限。您必须先为节点创建 IAM 角色以在启动它们时使用，然后才能启动这些节点并在集群中注册它们。此要求适用于通过由 Amazon 提供的 Amazon EKS 优化版 AMI 启动的节点，也适用于您打算使用的任何其他节点 AMI。

> **Note**
>
> 您不能使用创建任何集群时使用的相同角色。

必须先使用以下 IAM policy 创建 IAM 角色，然后才能创建节点：

- `AmazonEKSWorkerNodePolicy`
- `AmazonEC2ContainerRegistryReadOnly`
- `AmazonEKS_CNI_Policy` 托管策略（如果使用 `IPv4` 系列创建集群）或您创建的 IPv6 策略 (p. 246)（如果使用 `IPv6` 系列创建集群）。但是，我们建议您将策略附加到专门用于 Amazon VPC CNI 附加组件的单独角色，而不是附加到此角色。有关为 Amazon VPC CNI 附加组件创建单独角色的详细信息，请参阅 the section called “为 IAM 账户配置插件” (p. 243)。

> **Note**
>
> Amazon EC2 节点组必须具有与 Fargate 配置文件不同的 IAM 角色。有关更多信息，请参阅Amazon EKS pod 执行 IAM 角色 (p. 430)。

检查现有节点角色

可以使用以下过程检查并查看您的账户是否已有 Amazon EKS 节点角色。

在 IAM 控制台中检查 **eksNodeRole**

1. 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. 在角色列表中搜索 `eksNodeRole`、`AmazonEKSNodeRole` 或 `NodeInstanceRole`。如果其中一个名称的角色不存在，请参阅 创建 Amazon EKS 节点 IAM 角色 (p. 428) 以创建该角色。如果包含 `eksNodeRole`、`AmazonEKSNodeRole` 或 `NodeInstanceRole` 的角色确实存在，请选择该角色以查看附加的策略。
4. 请选择权限。
5. 确保将 AmazonEKSWorkerNodePolicy 和 AmazonEC2ContainerRegistryReadOnly 托管策略附加到此角色。如果已经附加策略，则您的 Amazon EKS 节点已经正确配置。

   > **Note**
   >
   > 如果 AmazonEKS_CNI_Policy 策略已附加到角色，我们建议删除此策略并改为将其附加到映射到 `aws-node` Kubernetes 服务账户的 IAM 角色。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。
6. 选择 Trust relationships（信任关系），然后选择 Edit trust policy（编辑信任策略）。
7. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel（取消）。如果信任关系不匹配，请将策略复制到 Edit trust policy（编辑信任策略）窗口并选择 Update policy（更新策略）。

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

```
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

创建 Amazon EKS 节点 IAM 角色

您可以使用 Amazon Web Services Management Console 或 Amazon CLI 创建节点 IAM 角色。选择带有要用于创建角色的工具名称的选项卡。

Amazon Web Services Management Console

在 IAM 控制台中创建您的Amazon EKS 节点角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Roles（角色）。
3. 在 Roles（角色）页面上，选择 Create role（创建角色）。
4. 在 Select trusted entity（选择受信任的实体）页面上，请执行以下操作：
   a. 在 Trusted entity type（受信任的实体类型）部分中，选择 Amazon service（服务）。
   b. 在 Use case（使用案例）下，选择 EC2。
   c. 选择 Next (下一步)。
5. 在 Add permissions（添加权限）页面上，请执行以下操作：
   a. 在 Filter policies (筛选器策略) 框中，输入 **AmazonEKSWorkerNodePolicy**。
   b. 选中搜索结果中的 AmazonEKSWorkerNodePolicy 左侧的复选框。
   c. 请选择 Clear filters（清除筛选条件）。
   d. 在 Filter policies (筛选器策略) 框中，输入 **AmazonEC2ContainerRegistryReadOnly**。
   e. 选中搜索结果中的 AmazonEC2ContainerRegistryReadOnly 左侧的复选框。

      AmazonEKS_CNI_Policy 托管策略或您创建的 IPv6 策略 (p. 246)还必须附加到此角色或映射到 `aws-node` Kubernetes 服务账户的其他角色。我们建议将策略分配到与 Kubernetes 服务账户关联的角色，而不是将其分配到此角色。有关更多信息，请参阅配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。
   f. 选择 Next (下一步)。
6. 在 Name, review, and create（命名、查看和创建）页面中，请执行以下操作：
   a. 对于 Role name（角色名称），请为角色输入唯一名称，例如 **AmazonEKSNodeRole**。
   b. 对于 Description（说明），请将当前文本替换为描述性文本，例如 **Amazon EKS - Node role**。
   c. 在 Add tags (Optional)（添加标签(可选)）下，通过以键值对的形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 IAM 用户指南中的标记 IAM 实体。
   d. 选择 Create role（创建角色）。

Amazon CLI

1. 运行以下命令以创建 `node-role-trust-relationship.json` 文件。

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

2. 创建 IAM 角色。

```
aws iam create-role \
  --role-name AmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-relationship.json"
```

3. 将两个所需的 IAM 托管策略附加到 IAM 角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name AmazonEKSNodeRole
```

4. 根据创建集群时使用的 IP 系列，将以下 IAM policy 之一附加到 IAM 角色。该策略必须附加到此角色或与用于 Amazon VPC CNI plugin for Kubernetes 的 Kubernetes `aws-node` 服务账户关联的角色。我们建议将策略分配到与 Kubernetes 服务账户关联的角色。要将策略分配到与 Kubernetes 服务账户关联的角色，请参阅 配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。

   - IPv4

     ```
     aws iam attach-role-policy \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
       --role-name AmazonEKSNodeRole
     ```

   - IPv6

     1. 复制以下文本并将其保存到名为 *vpc-cni-ipv6-policy*.json 的文件。

        ```
        {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "ec2:AssignIpv6Addresses",
                        "ec2:DescribeInstances",
                        "ec2:DescribeTags",
                        "ec2:DescribeNetworkInterfaces",
                        "ec2:DescribeInstanceTypes"
                    ],
                    "Resource": "*"
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "ec2:CreateTags"
                    ],
                    "Resource": [
        ```

```
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. 创建 IAM policy。

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

3. 将 IAM policy 附加到 IAM 角色。请将 *111122223333* 替换为您的账户 ID。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

Amazon EKS pod 执行 IAM 角色

要在 Amazon Fargate 基础设施上运行 pods，需要有 Amazon EKS pod 执行角色。

当您的集群在 pods 基础设施上创建 Amazon Fargate 时，在 Fargate 基础设施上运行的组件必须代表您调用 Amazon API。这是为了他们可以执行诸如从 Amazon ECR 中拉取容器镜像或将日志路由到其他 Amazon 服务的操作。Amazon EKS pod 执行角色提供执行此操作的 IAM 权限。

创建 Fargate 配置文件时，必须使用配置文件为在 Fargate 基础设施上运行的 Amazon EKS 组件指定 pod 执行角色。此角色将被添加到集群的 Kubernetes 基于角色的访问控制 (RBAC, Role based access control) 以进行授权。这允许在 Fargate 基础设施上运行的 `kubelet` 注册到您的 Amazon EKS 集群，以便它可以作为节点显示在您的集群中。

> Note
>
> Fargate 配置文件的 IAM 角色必须与 Amazon EC2 节点组的 IAM 角色不同。
>
> Important
>
> 在 Fargate pod 中运行的容器不能承担与 pod 执行角色相关联的 IAM 权限。要授予 Fargate pod 中的容器访问其他 Amazon 服务的权限，您必须使用 服务账户的 IAM 角色 (p. 395)。

在创建 Fargate 配置文件之前，必须使用 `AmazonEKSFargatePodExecutionRolePolicy` 创建 IAM 角色。

检查正确配置的现有 pod 执行角色

您可以使用以下程序检查并查看您的账户是否已有正确配置的 Amazon EKS pod 执行角色。为了避免混淆代理安全问题，该角色必须根据 `SourceArn` 限制访问权限。您可以根据需要修改执行角色，以包括对其他集群上的 Fargate 配置文件的支持。

在 IAM 控制台中检查 Amazon EKS pod 执行角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Roles（角色）。
3. 在 Roles（角色）页面上，搜索 AmazonEKSFargatePodExecutionRole 的角色列表。如果该角色不存在，请参阅 创建 Amazon EKS pod 执行角色 (p. 431) 以创建该角色。如果该角色存在，请选择该角色。
4. 在 AmazonEKSFargatePodExecutionRole 页面上，请执行以下操作：

   a. 请选择权限。

b. 确保将 AmazonEKSFargatePodExecutionRolePolicy Amazon 托管策略附加到该角色。
c. 选择 Trust Relationships（信任关系）。
d. 选择 Edit trust policy（编辑信任策略）。

5. 在 Edit trust policy（编辑信任策略）页面中，验证信任关系是否包含以下策略，并且在集群上是否有 Fargate 配置文件的行。如果是这样，请选择 Cancel（取消）。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

如果策略匹配但没有指定集群上 Fargate 配置文件的行，则可以在 `ArnLike` 对象顶部添加以下行。请将 *region-code* 替换为您的集群所在的 Amazon Web Services 区域，将 *111122223333* 替换为账户 ID，并将 *my-cluster* 替换为集群的名称。

```
"aws:SourceArn": "arn:aws:eks:region-code:111122223333:fargateprofile/my-cluster/*",
```

如果策略不匹配，请将之前的完整策略复制到表单中，然后选择 Update policy（更新策略）。请将 *region-code* 替换为集群所在的 Amazon Web Services 区域。如果您希望在您的账户中在所有 Amazon Web Services 区域使用相同角色，则将 *region-code* 替换为 *。请将 *111122223333* 替换为账户 ID，并将 *my-cluster* 替换为您的集群名称。如果您希望在账户中对所有集群使用相同角色，请将 *my-cluster* 替换为 *。

创建 Amazon EKS pod 执行角色

如果尚无针对集群的 Amazon EKS pod 执行角色，您可以使用 Amazon Web Services Management Console 或 Amazon CLI 来创建它。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console 创建 Amazon Fargate 容器组（pod）执行角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Roles（角色）。
3. 在 Roles（角色）页面上，选择 Create role（创建角色）。
4. 在 Select trusted entity（选择受信任的实体）页面上，请执行以下操作：
   a. 在 Trusted entity type（受信任的实体类型）部分中，选择 Amazon service（Amazon 服务）。
   b. 从 Use cases for other Amazon Web Services（其他 Amazon Web Services 的使用案例）下拉列表中，选择 EKS。

c. 选择 EKS - Fargate pod（EKS - Fargate 容器组）。
d. 选择 Next（下一步）。

5. 在 Add permissions（添加权限）页面上，选择 Next（下一步）。
6. 在 Name, review, and create（命名、查看和创建）页面中，请执行以下操作：

a. 对于 Role name（角色名称），请为角色输入唯一名称，例如 **AmazonEKSFargatePodExecutionRole**。
b. 在 Add tags (Optional)（添加标签(可选)）下，通过以键值对的形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 IAM 用户指南中的标记 IAM 实体。
c. 选择 Create role（创建角色）。

7. 在 Roles（角色）页面上，搜索 AmazonEKSFargatePodExecutionRole 的角色列表。选择角色。
8. 在 AmazonEKSFargatePodExecutionRole 页面上，请执行以下操作：

a. 选择 Trust Relationships（信任关系）。
b. 选择 Edit trust policy（编辑信任策略）。

9. 在 Edit trust policy（编辑信任策略）页面上，执行以下操作：

a. 将以下内容复制并粘贴到 Edit trust policy（编辑信任策略）表单中。将 *region-code* 替换为您的集群所在的 Amazon Web Services 区域。如果您希望在您的账户中在所有 Amazon Web Services 区域使用相同角色，则将 *region-code* 替换为 *。请将 *111122223333* 替换为账户 ID，并将 *my-cluster* 替换为您的集群名称。如果您希望在账户中对所有集群使用相同角色，请将 *my-cluster* 替换为 *。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
          }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. 选择 Update policy（更新策略）。

Amazon CLI

使用 Amazon CLI 创建 Amazon Fargate 容器组（pod）执行角色

1. 复制并将以下内容粘贴到名为 *pod-execution-role-trust-policy*.json 的文件中。将 *region-code* 替换为您的集群所在的 Amazon Web Services 区域。如果您希望在您的账户中在所有 Amazon Web Services 区域使用相同角色，则将 *region-code* 替换为 *。请将 *111122223333* 替换为账户 ID，并将 *my-cluster* 替换为您的集群名称。如果您希望在账户中对所有集群使用相同角色，请将 *my-cluster* 替换为 *。

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

```
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 创建 pod 执行 IAM 角色。

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

3. 将所需的 Amazon EKS 托管 IAM policy 附加到角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Amazon EKS Connector IAM 角色

您可以连接 Kubernetes 集群以便在 Amazon Web Services Management Console 中查看。要连接 Kubernetes 集群，创建 IAM 角色。

检查现有 connector 角色

您可以使用以下过程检查并查看您的账户是否已有 Amazon EKS connector 角色。

在 IAM 控制台中检查 **AmazonEKSConnectorAgentRole**

1. 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. 在角色列表中搜索 AmazonEKSConnectorAgentRole。如果不存在包含 AmazonEKSConnectorAgentRole 的角色，请参阅 创建 Amazon EKS Connector 代理角色 (p. 434) 来创建角色。如果包含 AmazonEKSConnectorAgentRole 的角色确实存在，则选择角色以查看附加的策略。
4. 请选择权限。
5. 确保将 AmazonEKSClusterPolicy 托管策略附加到此角色。如果附加该策略，则将正确配置 Amazon EKS 集群角色。
6. 选择 Trust relationships（信任关系），然后选择 Edit trust policy（编辑信任策略）。
7. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel（取消）。如果信任关系不匹配，请将策略复制到 Edit trust policy（编辑信任策略）窗口并选择 Update policy（更新策略）。

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

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

创建 Amazon EKS Connector 代理角色

您可以使用 Amazon Web Services Management Console 或 Amazon CloudFormation 创建 connector 代理角色。选择带有要用于创建角色的工具名称的选项卡。

Amazon CLI

1. 创建一个名为 `eks-connector-agent-trust-policy.json` 的文件，其中包含要用于 IAM 角色的以下 JSON。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 创建一个名为 `eks-connector-agent-policy.json` 的文件，其中包含要用于 IAM 角色的以下 JSON。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SsmControlChannel",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel"
            ],
            "Resource": "arn:aws:eks:*:*:cluster/*"
        },
        {
            "Sid": "ssmDataplaneOperations",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenDataChannel",
                "ssmmessages:OpenControlChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

3. 使用您在之前列表项中创建的信任策略和策略创建 Amazon EKS Connector 代理角色。

```
aws iam create-role \
     --role-name AmazonEKSConnectorAgentRole \
     --assume-role-policy-document file://eks-connector-agent-trust-policy.json
```

4. 将该策略附加到 Amazon EKS Connector 代理角色。

```
aws iam put-role-policy \
     --role-name AmazonEKSConnectorAgentRole \
     --policy-name AmazonEKSConnectorAgentPolicy \
     --policy-document file://eks-connector-agent-policy.json
```

Amazon CloudFormation

要用 Amazon CloudFormation 创建 Amazon EKS Connector 代理角色

1. 将以下 Amazon CloudFormation 模板保存到本地系统中的文本文件。

   Note

   此模板还创建服务相关角色，否则调用 `registerCluster` API 时将创建。有关详细信息，请参阅 使用角色将 Kubernetes 集群连接到 Amazon EKS (p. 423)。

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Provisions necessary resources needed to register clusters in EKS'
Parameters: {}
Resources:
  EKSConnectorSLR:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: eks-connector.amazonaws.com

  EKSConnectorAgentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action: [ 'sts:AssumeRole' ]
            Principal:
              Service: 'ssm.amazonaws.com'

  EKSConnectorAgentPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: EKSConnectorAgentPolicy
      Roles:
        - {Ref: 'EKSConnectorAgentRole'}
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateControlChannel' ]
            Resource:
            - Fn::Sub: 'arn:${AWS::Partition}:eks:*:*:cluster/*'
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateDataChannel', 'ssmmessages:OpenDataChannel',
 'ssmmessages:OpenControlChannel' ]
            Resource: "*"
```

```
Outputs:
  EKSConnectorAgentRoleArn:
    Description: The agent role that EKS connector uses to communicate with Amazon
 Web Services.
    Value: !GetAtt EKSConnectorAgentRole.Arn
```

2. 打开 Amazon CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
3. 选择 Create stack（创建堆栈）（无论是新资源还是现有资源。
4. 对于指定模板，选择上传模板文件，然后选择选择文件。
5. 选择您之前创建的文件，然后选择下一步。
6. 对于 Stack name (堆栈名称)，输入角色的名称，如 `eksConnectorAgentRole`，然后选择 Next (下一步)。
7. 在 Configure stack options（配置堆栈选项）页面上，请选择 Next（下一步）。
8. 在 Review（审核）页面上审核您的信息，确认堆栈可创建 IAM 资源，然后选择 Create stack（创建堆栈）。

Amazon Elastic Kubernetes Service 的Amazon托管策略

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 托管策略的更多信息，请参阅 IAM 用户指南中的 Amazon 托管策略。

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，服务最有可能会更新 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策略。例如，ReadOnlyAccess Amazon 托管策略提供对所有 Amazon 服务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。

Amazon托管策略：AmazonEKS_CNI_Policy

您可以将 `AmazonEKS_CNI_Policy` 附加到 IAM 实体。在创建 Amazon EC2 节点组之前，此策略必须附加到节点 IAM 角色 (p. 427)，或 Amazon VPC CNI plugin for Kubernetes 专用的 IAM 角色。这样它可以代表您执行操作。我们建议您将策略附加到仅由插件使用的角色。有关更多信息，请参阅 Amazon EKS 中使用适用于 Kubernetes 的 Amazon VPC CNI 插件的 Pod 联网 (p. 233) 和 配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户 (p. 243)。

权限详细信息

此策略包含允许 Amazon EKS 完成以下任务的以下权限：

- **ec2** – 允许 Amazon VPC CNI 插件执行操作，例如为 pods 预置弹性网络接口和 IP 地址，以便为在 Amazon EKS 中运行的应用程序提供联网。

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

```
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignPrivateIpAddresses",
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes",
                "ec2:DetachNetworkInterface",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:UnassignPrivateIpAddresses"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

Amazon托管策略：AmazonEKSClusterPolicy

您可以将 `AmazonEKSClusterPolicy` 附加到您的 IAM 实体。在创建集群之前，您必须拥有附加了此策略的集群 IAM 角色 (p. 425)。由 Amazon EKS 托管的 Kubernetes 集群会代表您调用其他 Amazon 服务。它们这样做的目的是为了管理您与服务一起使用的资源。

此策略包含允许 Amazon EKS 完成以下任务的以下权限：

- **`autoscaling`** – 读取和更新 Auto Scaling 组的配置。Amazon EKS 不使用这些权限，但它们保留在策略中，以确保向后兼容。
- **`ec2`** – 使用与 Amazon EC2 节点关联的卷和网络资源执行工作。此为必需操作，以便 Kubernetes 控制面板可以将实例加入到集群，并动态预置和管理 Kubernetes 持久卷请求的 Amazon EBS 卷。
- **`elasticloadbalancing`** – 使用 Elastic Load Balancer 并将节点添加到其中作为目标。此为必需操作，以便 Kubernetes 控制面板能够动态预置 Kubernetes 服务请求的 Elastic Load Balancer。
- **`iam`** – 创建服务相关角色。此为必需操作，以便 Kubernetes 控制面板能够动态预置 Kubernetes 服务请求的 Elastic Load Balancer。
- **`kms`** – 从 Amazon KMS 中读取密钥。这对于 Kubernetes 控制面板是必需的，以支持 `etcd` 中存储的 Kubernetes 密钥的密钥加密。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:UpdateAutoScalingGroup",
                "ec2:AttachVolume",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateRoute",
```

```
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:DeleteRoute",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteVolume",
                "ec2:DescribeInstances",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVolumes",
                "ec2:DescribeVolumesModifications",
                "ec2:DescribeVpcs",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DetachVolume",
                "ec2:ModifyInstanceAttribute",
                "ec2:ModifyVolume",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeInternetGateways",
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
                "elasticloadbalancing:AttachLoadBalancerToSubnets",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateLoadBalancerListeners",
                "elasticloadbalancing:CreateLoadBalancerPolicy",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeleteLoadBalancerListeners",
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
                "elasticloadbalancing:DescribeLoadBalancerPolicies",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetGroupAttributes",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DetachLoadBalancerFromSubnets",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:ModifyLoadBalancerAttributes",
                "elasticloadbalancing:ModifyTargetGroup",
                "elasticloadbalancing:ModifyTargetGroupAttributes",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer",
                "elasticloadbalancing:SetLoadBalancerPoliciesOfListener",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "elasticloadbalancing.amazonaws.com"
                }
            }
```

```
        }
    ]
}
```

Amazon托管策略：AmazonEKSFargatePodExecutionRolePolicy

您可以将 `AmazonEKSFargatePodExecutionRolePolicy` 附加到您的 IAM 实体。在创建 Fargate 配置文件之前，您必须创建 Fargate pod 执行角色并将此策略附加到其上。有关更多信息，请参阅 创建 Fargate pod 执行角色 (p. 129) 和 Amazon Fargate 配置文件 (p. 131)。

此策略向该角色授予权限，以提供对必须在 Fargate 上运行 Amazon EKS pods 的其他 Amazon 服务资源的访问权限。

权限详细信息

此策略包含允许 Amazon EKS 完成以下任务的以下权限：

- **ecr** – 允许在 Fargate 上运行的容器组（pod）提取存储在 Amazon ECR 中的容器镜像。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon托管策略：AmazonEKSForFargateServiceRolePolicy

您不能将 `AmazonEKSForFargateServiceRolePolicy` 附加到您的 IAM 实体。将此策略附加到允许 Amazon EKS 代表您执行操作的服务相关角色。有关更多信息，请参阅 AWSServiceRoleforAmazonEKSForFargate。

此策略授予 Amazon EKS 运行 Fargate 任务所必要的权限。仅当您具有 Fargate 节点时，才会使用此策略。

权限详细信息

此策略包含允许 Amazon EKS 完成以下任务的以下权限。

- **ec2** – 创建和删除弹性网络接口，并描述弹性网络接口和资源。此为必需操作，以便 Amazon EKS Fargate 服务可以配置 Fargate 容器组（pod）所需的 VPC 联网。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
```

```
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon托管策略：AmazonEKSServicePolicy

您可以将 `AmazonEKSServicePolicy` 附加到您的 IAM 实体。在 2020 年 4 月 16 日之前创建的集群需要您创建 IAM 角色并向其附加此策略。在 2020 年 4 月 16 日或之后创建的集群无需您创建角色，也不需要您分配此策略。当您使用具有 `iam:CreateServiceLinkedRole` 权限的 IAM 委托人创建集群时，将会自动为您创建 AmazonServiceRoleforAmazonEKS (p. 419) 服务相关角色。服务相关角色附加了 Amazon托管策略：AmazonEKSServiceRolePolicy (p. 441)。

此策略允许 Amazon EKS 创建和管理运行 Amazon EKS 集群所需的资源。

权限详细信息

此策略包含允许 Amazon EKS 完成以下任务的以下权限。

- **`eks`** – 启动更新后，更新您的集群的 Kubernetes 版本。Amazon EKS 不使用此权限，但其仍保留在策略中，以确保向后兼容。
- **`ec2`** – 使用弹性网络接口和其他网络资源和标签。此为必需操作，以便 Amazon EKS 配置联网，促进节点与 Kubernetes 控制面板之间的通信。
- **`route53`** – 将 VPC 与托管区域关联。此为必需操作，以便 Amazon EKS 为 Kubernetes 集群 API 服务器启用私有端点联网。
- **`logs`** – 录入事件。此为必需操作，这样 Amazon EKS 就可以将 Kubernetes 控制面板日志发送到 CloudWatch。
- **`iam`** – 创建服务相关角色。此为必需操作，以便 Amazon EKS 可以代表您创建 `AWSServiceRoleForAmazonEKS` (p. 419) 服务相关角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:ModifyNetworkInterfaceAttribute",
                "iam:ListAttachedRolePolicies",
                "eks:UpdateClusterVersion"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "route53:AssociateVPCWithHostedZone",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/eks/*:*"
        },
        {
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:*:*:log-group:/aws/eks/*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

Amazon托管策略：AmazonEKSServiceRolePolicy

您不能将 `AmazonEKSServiceRolePolicy` 附加到您的 IAM 实体。将此策略附加到允许 Amazon EKS 代表您执行操作的服务相关角色。有关更多信息，请参阅Amazon EKS 的服务相关角色权限 (p. 419)。当您使用具有 `iam:CreateServiceLinkedRole` 权限的 IAM 委托人创建集群时，将会自动为您创建 AmazonServiceRoleforAmazonEKS (p. 419) 服务相关角色，而且此策略会附加到该角色。

此策略允许服务相关角色代表您调用Amazon服务。

权限详细信息

此策略包含允许 Amazon EKS 完成以下任务的以下权限。

- **ec2** – 创建和描述弹性网络接口和 Amazon EC2 实例、集群安全组 (p. 231)及创建集群所需的 VPC。

- **iam** – 列出附加到 IAM 角色的所有托管策略。此为必需操作，以便 Amazon EKS 能够列出和验证创建集群所需的所有托管策略和权限。
- 将 VPC 与托管区关联 – 此为必需操作，以便 Amazon EKS 为 Kubernetes 集群 API 服务器启用私有端点联网。
- 录入事件 – 此为必需操作，这样 Amazon EKS 就可以将 Kubernetes 控制面板日志发送到 CloudWatch。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DetachNetworkInterface",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeInstances",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:CreateNetworkInterfacePermission",
                "iam:ListAttachedRolePolicies",
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteSecurityGroup",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*",
            "Condition": {
                "ForAnyValue:StringLike": {
                    "ec2:ResourceTag/Name": "eks-cluster-sg*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*"
            ],
            "Condition": {
                "ForAnyValue:StringLike": {
                    "aws:TagKeys": [
                        "kubernetes.io/cluster/*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
```

```
                "ec2:DeleteTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:security-group/*"
            ],
            "Condition": {
                "ForAnyValue:StringLike": {
                    "aws:TagKeys": [
                        "kubernetes.io/cluster/*"
                    ],
                    "aws:RequestTag/Name": "eks-cluster-sg*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "route53:AssociateVPCWithHostedZone",
            "Resource": "arn:aws:route53:::hostedzone/*"
        },
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:*:*:log-group:/aws/eks/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/eks/*:*"
        },
        {
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:*:*:log-group:/aws/eks/*:*:*"
        }
    ]
}
```

Amazon托管策略：AmazonEKSVPCResourceController

您可以将 `AmazonEKSVPCResourceController` 策略附加得到 IAM 身份。如果您使用适用于 pods 的安全组 (p. 273)，您必须将此策略附加到您的 Amazon EKS 集群 IAM 角色 (p. 425)，以便其代表您执行操作。

此策略授予集群角色管理弹性网络接口和节点 IP 地址的权限。

权限详细信息

此策略包含允许 Amazon EKS 完成以下任务的以下权限：

- **ec2** – 管理弹性网络接口和 IP 地址，以支持 pod 安全组和 Windows 节点。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
```

```
                    "ec2:ResourceTag/eks:eni:owner": "eks-vpc-resource-controller"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DetachNetworkInterface",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:AttachNetworkInterface",
                "ec2:UnassignPrivateIpAddresses",
                "ec2:AssignPrivateIpAddresses"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon托管策略：AmazonEKSWorkerNodePolicy

您可以将 `AmazonEKSWorkerNodePolicy` 附加到 IAM 实体。您必须将此策略附加到您在创建允许 Amazon EKS 代表您执行操作的 Amazon EC2 节点时指定的节点 IAM 角色 (p. 427)。如果您使用 `eksctl` 创建节点组，则其会创建节点 IAM 角色并自动将此策略附加到角色。

此策略授予 Amazon EKS Amazon EC2 节点连接到 Amazon EKS 集群的权限。

权限详细信息

此策略包含允许 Amazon EKS 完成以下任务的以下权限：

- **ec2** – 读取实例卷和网络信息。此为必需操作，以便 Kubernetes 节点能够描述有关节点加入 Amazon EKS 集群所需的 Amazon EC2 资源的信息。
- **eks** – 可选地将集群描述为节点引导启动的一部分。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVolumes",
                "ec2:DescribeVolumesModifications",
                "ec2:DescribeVpcs",
                "eks:DescribeCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon托管策略：AWSServiceRoleForAmazonEKSNodegroup

您不能将 AmazonServiceRoleForAmazonEKSNodegroup 附加到您的 IAM 实体。将此策略附加到允许 Amazon EKS 代表您执行操作的服务相关角色。有关更多信息，请参阅Amazon EKS 的服务相关角色权限 (p. 420)。

此策略授予 AmazonServiceRoleForAmazonEKSNodegroup 角色权限，允许其在您的账户中创建和管理 Amazon EC2 节点组。

权限详细信息

此策略包含允许 Amazon EKS 完成以下任务的以下权限：

- **ec2** – 使用安全组、标签和启动模板。这对于 Amazon EKS 托管节点组启用远程访问配置是必需的。此外，Amazon EKS 托管节点组会代表您创建启动模板。这样做的目的是配置为每个托管节点组提供支持的 Amazon EC2 Auto Scaling 组。
- **iam** – 创建服务相关角色并传递角色。这是 Amazon EKS 托管节点组管理创建托管节点组时传递的角色的实例配置文件所必需的。此实例配置文件由作为托管节点组的一部分启动的 Amazon EC2 实例使用。Amazon EKS 需要为其他服务（如 Amazon EC2 Auto Scaling 组）创建服务相关角色。它们用于创建托管节点组
- **autoscaling** – 使用安全 Auto Scaling 组。这是 Amazon EKS 托管节点组管理支持每个托管节点组的 Amazon EC2 Auto Scaling 组所必需的。它还用于支持一些功能，例如，在节点组更新期间终止或回收节点时移出 pods。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SharedSecurityGroupRelatedPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:RevokeSecurityGroupIngress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:DescribeInstances",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:DeleteSecurityGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/eks": "*"
                }
            }
        },
        {
            "Sid": "EKSCreatedSecurityGroupRelatedPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:RevokeSecurityGroupIngress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:DescribeInstances",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:DeleteSecurityGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/eks:nodegroup-name": "*"
```

```
                }
            }
        },
        {
            "Sid": "LaunchTemplateRelatedPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/eks:nodegroup-name": "*"
                }
            }
        },
        {
            "Sid": "AutoscalingRelatedPermissions",
            "Effect": "Allow",
            "Action": [
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "autoscaling:CompleteLifecycleAction",
                "autoscaling:PutLifecycleHook",
                "autoscaling:PutNotificationConfiguration",
                "autoscaling:EnableMetricsCollection"
            ],
            "Resource": "arn:aws:autoscaling:*:*:*:autoScalingGroupName/eks-*"
        },
        {
            "Sid": "AllowAutoscalingToCreateSLR",
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "autoscaling.amazonaws.com"
                }
            },
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*"
        },
        {
            "Sid": "AllowASGCreationByEKS",
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:CreateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "eks",
                        "eks:cluster-name",
                        "eks:nodegroup-name"
                    ]
                }
            }
        },
        {
            "Sid": "AllowPassRoleToAutoscaling",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
```

```
            "StringEquals": {
                "iam:PassedToService": "autoscaling.amazonaws.com"
            }
        }
    },
    {
        "Sid": "AllowPassRoleToEC2",
        "Effect": "Allow",
        "Action": "iam:PassRole",
        "Resource": "*",
        "Condition": {
            "StringEqualsIfExists": {
                "iam:PassedToService": [
                    "ec2.amazonaws.com",
                    "ec2.amazonaws.com.cn"
                ]
            }
        }
    },
    {
        "Sid": "PermissionsToManageResourcesForNodegroups",
        "Effect": "Allow",
        "Action": [
            "iam:GetRole",
            "ec2:CreateLaunchTemplate",
            "ec2:DescribeInstances",
            "iam:GetInstanceProfile",
            "ec2:DescribeLaunchTemplates",
            "autoscaling:DescribeAutoScalingGroups",
            "ec2:CreateSecurityGroup",
            "ec2:DescribeLaunchTemplateVersions",
            "ec2:RunInstances",
            "ec2:DescribeSecurityGroups",
            "ec2:GetConsoleOutput",
            "ec2:DescribeRouteTables",
            "ec2:DescribeSubnets"
        ],
        "Resource": "*"
    },
    {
        "Sid": "PermissionsToCreateAndManageInstanceProfiles",
        "Effect": "Allow",
        "Action": [
            "iam:CreateInstanceProfile",
            "iam:DeleteInstanceProfile",
            "iam:RemoveRoleFromInstanceProfile",
            "iam:AddRoleToInstanceProfile"
        ],
        "Resource": "arn:aws:iam::*:instance-profile/eks-*"
    },
    {
        "Sid": "PermissionsToManageEKSAndKubernetesTags",
        "Effect": "Allow",
        "Action": [
            "ec2:CreateTags",
            "ec2:DeleteTags"
        ],
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringLike": {
                "aws:TagKeys": [
                    "eks",
                    "eks:cluster-name",
                    "eks:nodegroup-name",
                    "kubernetes.io/cluster/*"
                ]
```

```
                }
            }
        }
    ]
}
```

Amazon 托管策略：AmazonEBSCSIDriverPolicy

`AmazonEBSCSIDriverPolicy` 允许 Amazon EBS Container Storage Interface（CSI）驱动程序代表您创建、修改、附加、分离和删除卷。它还授予 EBS CSI 驱动程序创建和删除快照以及列出实例、卷和快照的权限。

`AmazonEBSCSIDriverPolicy` 包含以下权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateSnapshot",
        "ec2:AttachVolume",
        "ec2:DetachVolume",
        "ec2:ModifyVolume",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInstances",
        "ec2:DescribeSnapshots",
        "ec2:DescribeTags",
        "ec2:DescribeVolumes",
        "ec2:DescribeVolumesModifications"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:volume/*",
        "arn:aws:ec2:*:*:snapshot/*"
      ],
      "Condition": {
        "StringEquals": {
          "ec2:CreateAction": [
            "CreateVolume",
            "CreateSnapshot"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteTags"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:volume/*",
        "arn:aws:ec2:*:*:snapshot/*"
      ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "ec2:CreateVolume"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/ebs.csi.aws.com/cluster": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateVolume"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/CSIVolumeName": "*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateVolume"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/kubernetes.io/cluster/*": "owned"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteVolume"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteVolume"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "ec2:ResourceTag/CSIVolumeName": "*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteVolume"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
```

```
          "ec2:ResourceTag/kubernetes.io/cluster/*": "owned"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteSnapshot"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "ec2:ResourceTag/CSIVolumeSnapshotName": "*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteSnapshot"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "ec2:ResourceTag/ebs.csi.aws.com/cluster": "true"
        }
      }
    }
  ]
}
```

Amazon托管策略的 Amazon EKS 更新

查看有关 Amazon EKS（自从其开始跟踪更新更改以来）的Amazon托管策略的更新的详细信息。有关此页面更改的自动提示，请订阅 Amazon EKS 文档历史记录页面上的 RSS 源。

更改	说明	日期
引入的 AmazonEBSCSIDriverPolicy（p. 448）	Amazon 引入了 AmazonEBSCSIDriverPolicy。	2022 年 4 月 22 日
已添加权限到 AmazonEKSWorkerNodePolicy（p. 443）	增加了 `ec2:DescribeInstanceTypes` 以启用能够自动发现实例级别属性的 Amazon EKS 优化版 AMI。	2022 年 3 月 21 日
已将权限添加至 AWSServiceRoleForAmazonEKSNodegroup（p. 445）	添加了 `autoscaling:EnableMetricsCollection` 权限以允许 Amazon EKS 启用指标收集。	2021 年 12 月 13 日
已将权限添加至 AmazonEKSClusterPolicy（p. 437）	添加了 `ec2:DescribeAccountAttributes`、`ec2:DescribeAddresses` 和 `ec2:DescribeInternetGateways` 权限，以允许 Amazon EKS 为 Network Load Balancer 创建服务相关角色。	2021 年 6 月 17 日
Amazon EKS 已开始跟踪更改。	Amazon EKS 开始跟踪其Amazon托管策略的更改。	2021 年 6 月 17 日

Amazon EKS 身份和访问问题排查

要诊断并修复您在使用 Amazon EKS 和 IAM 时可能遇到的常见问题，请参阅 IAM 故障排除 (p. 487)。

Amazon Elastic Kubernetes Service 的合规性验证

作为多个 Amazon 合规性计划的一部分，第三方审核员将评估 Amazon Web Services的安全性与合规性，例如 SOC、PCI、FedRAMP 和 HIPAA。

要了解 Amazon EKS 或其他 Amazon Web Services 是否在特定合规性计划范围内，请参阅合规性计划范围内的 Amazon Web Services。有关常规信息，请参阅Amazon Web Services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、在 Amazon Artifact 中下载报告。

您使用 Amazon Web Services 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的法律法规。Amazon 提供以下资源来帮助满足合规性：

- 安全性与合规性快速入门指南安全性与合规性快速入门指南 – 这些部署指南讨论了架构注意事项，并提供了在Amazon上部署注重安全性和合规性的基准环境的步骤。
- 亚马逊云科技上的 HIPAA 安全性和合规性架构设计 – 此白皮书介绍了企业如何使用 Amazon Web Services 创建符合 HIPAA 要求的应用程序。

  Note

  并非所有 Amazon Web Services都符合 HIPAA 要求。有关更多信息，请参阅符合 HIPAA 要求的服务参考。
- 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 – 此 Amazon Config 服务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 Web Service 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规范。

Amazon EKS 中的恢复能力

Amazon全球基础设施围绕Amazon Web Services 区域和可用区构建。Amazon Web Services 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。

Amazon EKS 跨多个 Amazon 可用区运行和扩展 Kubernetes 控制面板以确保高可用性。Amazon EKS 可以根据负载自动缩放控制层面实例，检测并替换运行状况不佳的控制层面实例，并自动修补控制层面。启动版本更新后，Amazon EKS 会为您更新控制面板，从而在更新期间保持控制面板的高可用性。

此控制面板包含至少两个 API 服务器实例和三个 `etcd` 实例，这些实例在一个 Amazon Web Services 区域内的三个可用区之间运行。Amazon EKS：

- 主动监控控制层面实例上的负载，并自动扩展它们以确保高性能。
- 自动检测和替换运行状况不佳的控制面板实例，并根据需要跨 Amazon Web Services 区域内的可用区重新启动它们。
- 利用 Amazon Web Services 区域区域的架构以保持高可用性。因此，Amazon EKS 能够提供确保 API 服务器端点可用性的 SLA。

有关 Amazon Web Services 区域 和可用区的更多信息，请参阅 Amazon 全球基础设施。

Amazon EKS 中的基础设施安全性

作为一项托管服务，Amazon EKS 由Amazon Web Services：安全流程概述文件中所述的 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 EKS。客户端必须支持传输层安全性协议（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。

在创建 Amazon EKS 集群时，为要使用的集群指定 VPC 子网。Amazon EKS 需要至少位于两个可用区中的子网。我们建议使用同时带有公有子网和私有子网的 VPC，以便 Kubernetes 可以在公有子网中创建公有负载均衡器，进而将流量负载平衡到在私有子网中的节点上运行的容器组（pod）。

有关 VPC 注意事项的更多信息，请参阅Amazon EKS VPC 和子网要求和注意事项 (p. 226)。

如果使用 开始使用 Amazon EKS (p. 4) 演练中提供的 Amazon CloudFormation 模板创建 VPC 和节点组，则将使用推荐设置来配置控制层面和节点安全组。

有关安全组注意事项的更多信息，请参阅Amazon EKS 安全组要求和注意事项 (p. 231)。

在创建新集群时，Amazon EKS 将为您用于与集群进行通信的托管 Kubernetes API 服务器（使用 Kubernetes 管理工具，如 `kubectl`）创建端点。预设情况下，此 API 服务器端点对于 Internet 是公有的，对 API 服务器的访问将使用 Amazon Identity and Access Management（IAM）与本机 Kubernetes 基于角色的访问控制（RBAC）相结合的方式加以保护。

您可以启用对 Kubernetes API 服务器的私有访问，以便节点与 API 服务器之间的所有通信都在 VPC 内。您可以限制从 Internet 访问 API 服务器的 IP 地址，或完全禁用对 API 服务器的 Internet 访问。

有关修改集群终端节点访问的更多信息，请参阅修改集群终端节点访问 (p. 38)。

您可以使用 Project Calico (p. 306) 等工具来实施网络策略。Project Calico 是一个第三方开源项目。有关更多信息，请参阅 Project Calico 文档。

Amazon EKS 中的配置和漏洞分析

安全性是配置和维护 Kubernetes 集群和应用程序的重要注意事项。Center for Internet Security (CIS) Kubernetes 基准提供了关于 Amazon EKS 节点安全配置的指导。该基准：

- 适用于 Amazon EC2 节点（托管和自我管理），您在其中负责 Kubernetes 组件的安全配置。
- 提供了一种社区批准的标准方法，以确保您在使用 Amazon EKS 时已安全地配置了 Kubernetes 集群和节点。
- 由四个部分组成：控制层面日志记录配置、节点安全配置、策略和托管服务。
- 支持 Amazon EKS 中当前提供的所有 Kubernetes 版本，并且可以使用 kube-bench（一个用于在 Kubernetes 集群上使用 CIS 基准检查配置的标准开源工具）运行。

要了解更多信息，请参阅 CIS Amazon EKS 基准简介。

Amazon EKS 平台版本表示集群控制面板的功能，包括启用哪些 Kubernetes API 服务器标志和当前的 Kubernetes 补丁版本。使用最新平台版本部署新集群。有关详细信息，请参阅Amazon EKS 平台版本 (p. 67)。

您可以将 Amazon EKS 集群更新 (p. 28)到更新的 Kubernetes 版本。随着新 Kubernetes 版本在 Amazon EKS 中提供，我们建议您主动将集群更新为使用最新的可用版本。有关 EKS 中 Kubernetes 版本的更多信息，请参阅 Amazon EKS Kubernetes 版本 (p. 59)。

在 Amazon Linux 安全中心跟踪 Amazon Linux 2 的安全和隐私事件，或订阅关联的 RSS 源。安全和隐私事件包括受影响问题的概述、程序包以及更新实例以解决问题的说明。

您可以使用 Amazon Inspector 来检查节点的意外网络访问和这些 Amazon EC2 实例上的漏洞。

Amazon EKS 的安全最佳实践

Github 提供 Amazon EKS 安全最佳实践：https://aws.github.io/aws-eks-best-practices/security/docs/

Pod 安全策略

Kubernetes pod 安全策略准入控制器根据一组规则验证 pod 创建和更新请求。预设情况下，Amazon EKS 集群附带完全宽松的安全策略（没有任何限制）。有关更多信息，请参阅 Kubernetes 文档中的容器组（pod）安全策略。

Note

此功能在 Kubernetes 版本 `1.21` 中已被弃用，并计划在版本 `1.25` 中删除。要了解更多信息，请参阅 Kubernetes 博客上的 Deprecation: Past, Present, and Future（PodSecurityPolicy 弃用：过去、现在和未来），以及 Amazon 博客上的 Using Gatekeeper as a drop-in Pod Security Policy replacement in Amazon EKS [在 Amazon EKS 中使用 Gatekeeper 作为容器组（pod）安全策略的简易替代]。在版本 `1.23` 中，它被替换为容器组（pod）安全准入功能。要了解此功能的更多信息，请参阅 Amazon EKS 最佳实践指南中的 Pod Security Standards (PSS) and Pod Security Admission (PSA) [容器组（pod）安全标准（PSS）和容器组（pod）安全准入（PSA）]。

Amazon EKS 默认 pod 安全策略

带 Kubernetes 版本 `1.13` 和更高版本的 Amazon EKS 集群具有名为 `eks.privileged` 的默认 pod 安全策略。此策略对于系统中可以接受什么类型的 pod 没有限制，这相当于在禁用 `PodSecurityPolicy` 控制器的情况下运行 Kubernetes。

Note

创建此策略是为了与未启用 `PodSecurityPolicy` 控制器的集群保持向后兼容性。您可以针对集群、各个命名空间和服务账户创建更具有限制性的策略，然后删除默认策略以启用这些更具有限制性的策略。

您可以使用以下命令查看默认策略。

```
kubectl get psp eks.privileged
```

输出示例如下。

```
NAME              PRIV    CAPS    SELINUX    RUNASUSER    FSGROUP    SUPGROUP    READONLYROOTFS
  VOLUMES
```

```
eks.privileged   true   *      RunAsAny   RunAsAny    RunAsAny   RunAsAny   false
  *
```

有关更多详细信息，您可以使用以下命令描述此策略。

```
kubectl describe psp eks.privileged
```

输出示例如下。

```
Name:  eks.privileged

Settings:
  Allow Privileged:                       true
  Allow Privilege Escalation:             0xc0004ce5f8
  Default Add Capabilities:               <none>
  Required Drop Capabilities:             <none>
  Allowed Capabilities:                   *
  Allowed Volume Types:                   *
  Allow Host Network:                     true
  Allow Host Ports:                       0-65535
  Allow Host PID:                         true
  Allow Host IPC:                         true
  Read Only Root Filesystem:              false
  SELinux Context Strategy: RunAsAny
    User:                                 <none>
    Role:                                 <none>
    Type:                                 <none>
    Level:                                <none>
  Run As User Strategy: RunAsAny
    Ranges:                               <none>
  FSGroup Strategy: RunAsAny
    Ranges:                               <none>
  Supplemental Groups Strategy: RunAsAny
    Ranges:                               <none>
```

您可以在完整 YAML 文件中查看 `eks.privileged` pod 安全策略、其集群角色以及 安装或恢复默认的 pod 安全策略 (p. 455) 中绑定的集群角色。

删除默认的 Amazon EKS pod 安全策略

如果为 pods 创建更加限制的策略，创建后，可以删除默认的 Amazon EKS `eks.privileged` pod 安全策略，以启用您的自定义策略。

> **Important**
>
> 如果您使用的是版本 `1.7.0` 或更高版本的 CNI 插件，并将自定义容器组（pod）安全策略分配给用于 Daemonset 部署的 `aws-node`容器组（pod）的 `aws-node` Kubernetes 服务账户，则该策略的 `allowedCapabilities` 部分必须具有 `NET_ADMIN`，而且策略的 `spec` 必须具有 `hostNetwork: true` 和 `privileged: true`。

删除默认的 pod 安全策略

1. 创建一个名为 *`privileged-podsecuritypolicy.yaml`* 的文件，具有 安装或恢复默认的 pod 安全策略 (p. 455) 中示例文件的内容。
2. 使用以下命令删除 YAML。这将删除默认的 pod 安全策略、`ClusterRole` 以及与其相关联的 `ClusterRoleBinding`。

   ```
   kubectl delete -f privileged-podsecuritypolicy.yaml
   ```

安装或恢复默认的 pod 安全策略

如果要从 Kubernetes 的早期版本升级，或者您已修改或删除默认的 Amazon EKS `eks.privileged` pod 安全策略，则可按照以下步骤恢复该策略。

安装或恢复默认的 pod 安全策略

1. 创建以下内容的名为 *privileged-podsecuritypolicy.yaml* 的文件。

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: eks.privileged
  annotations:
    kubernetes.io/description: 'privileged allows full unrestricted access to
      pod features, as if the PodSecurityPolicy controller was not enabled.'
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  volumes:
  - '*'
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
  readOnlyRootFilesystem: false

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:podsecuritypolicy:privileged
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
rules:
- apiGroups:
  - policy
  resourceNames:
  - eks.privileged
  resources:
  - podsecuritypolicies
  verbs:
  - use

---
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: ClusterRoleBinding
metadata:
  name: eks:podsecuritypolicy:authenticated
  annotations:
    kubernetes.io/description: 'Allow all authenticated users to create privileged
 pods.'
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:podsecuritypolicy:privileged
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: system:authenticated
```

2. 使用以下命令应用 YAML。

```
kubectl apply -f privileged-podsecuritypolicy.yaml
```

将 Kubernetes 与 Amazon Secrets Manager 密钥结合使用

要将 Secrets Manager 中的密钥和 Parameter Store 中的参数显示为挂载在 Amazon EKS pods 中的文件，您可以使用 Kubernetes Secrets Store CSI Driver 的 Amazon 密钥和配置提供程序 (ASCP)。

使用 ASCP，您可以在 Secrets Manager 中存储并管理密钥，然后通过 Amazon EKS 上运行的工作负载检索。您可以使用 IAM 角色和策略限制集群中特定 Kubernetes pods 钥访问权限。ASCP 检索 pod 标识并交换 IAM 角色的身份。ASCP 担任 pod 的 IAM 角色，然后可以从授权该角色的 Secrets Manager 中检索密钥。

如果对密钥使用 Secrets Manager 自动轮换，还可以使用 Secrets Store CSI Driver 轮换协调程序功能，确保从 Secrets Manager 中检索最新的密码。

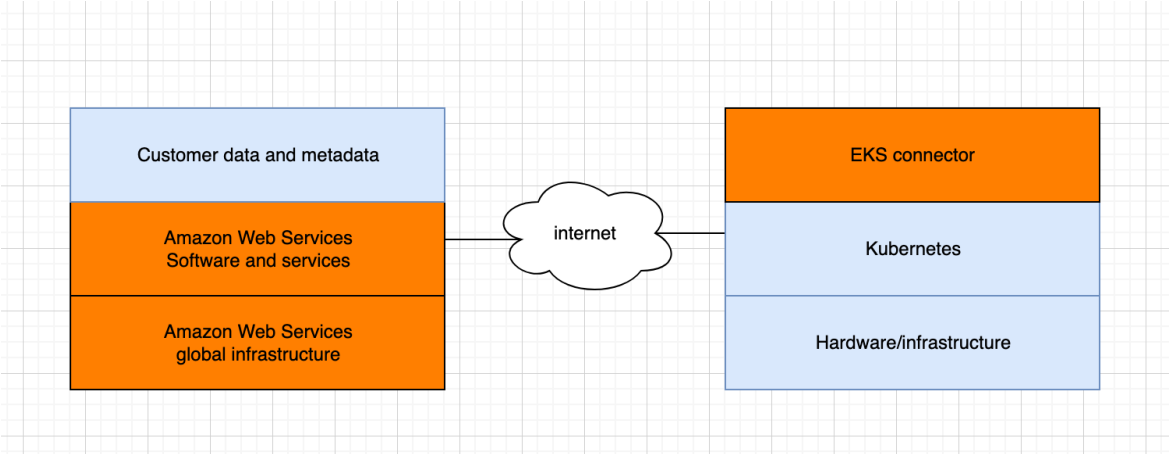
有关更多信息，请参阅 Amazon Secrets Manager 用户指南中的在 Amazon EKS 中使用 Secrets Manager 密钥。

Amazon EKS Connector 注意事项

Important

此功能在 Amazon Web Services 区域（中国）不可用。

Amazon EKS Connector 是在您的 Kubernetes 集群上运行的开源组件。此集群可以位于 Amazon 环境之外。这为安全责任创造了额外的注意事项。下面的图表对此配置进行说明。橙色代表 Amazon 责任，蓝色代表客户责任：



本主题介绍了连接的集群在 Amazon 外部时责任模型的差异。

Amazon 责任

- 维护、构建和交付 Amazon EKS Connector，这是在客户的 Kubernetes 集群上运行并与 Amazon 通信的开源组件。
- 维护连接的 Kubernetes 集群和 Amazon 服务之间的传输层和应用程序层通信安全。

客户责任

- Kubernetes 集群特定的安全性，具体来说如下：
  - Kubernetes 密钥必须得到正确的加密和保护。
  - 锁定对 `eks-connector` 命名空间的访问权限。
- 配置基于角色的访问控制（RBAC）权限以管理 Amazon 的用户访问权限。有关说明，请参阅向用户授予访问权限以查看集群。
- 安装和升级 Amazon EKS Connector。
- 维护支持已连接的 Kubernetes 集群的硬件、软件和基础设施。
- 保护他们的 Amazon 账户安全（例如，通过使用安全的根用户凭证）。

查看 Kubernetes 资源

您可以使用 Amazon Web Services Management Console 查看部署到您的集群的 Kubernetes 资源。您无法使用 Amazon CLI 或 `eksctl` (p. 10) 查看 Kubernetes 资源。要使用命令行工具查看 Kubernetes 资源，请使用 `kubectl` (p. 4)。

先决条件

要在 Amazon Web Services Management Console 中查看 Compute（计算）选项卡上的 Resources（资源）选项卡和 Nodes（节点）部分，您登录到 Amazon Web Services Management Console 的用户或者您在登录后切换到的角色必须具有特定的 IAM 和 Kubernetes 权限。有关更多信息，请参阅所需权限 (p. 458)。

使用 Amazon Web Services Management Console 查看 Kubernetes 资源

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 在 Clusters（集群）列表中，选择包含要查看的 Kubernetes 资源的集群。
3. 选择 Resources 选项卡。
4. 选择您要查看其资源的 Resource type（资源类型）组，例如 Workloads（工作负载）。您可以看到该组中的资源类型列表。
5. 选择资源类型，例如 Workloads（工作负载）组中的 Deployments（部署）。您可以看到资源类型的描述、Kubernetes 文档的链接以获取有关资源类型的更多信息，以及在集群上部署的该类型资源的列表。如果列表为空，则表示您的集群中没有部署此类型的资源。
6. 选择一种资源以查看关于该资源的更多信息。请尝试以下示例：

   - 依次选择 Workloads（工作负载）组、Deployments（部署）资源类型和 coredns 资源。当您选择资源时，您默认情况下处于 Structured view（结构化视图）中。对于某些资源类型，您会在 Structured view（结构化视图）中看到 Pods（容器组（pod））部分。本节列出了由工作负载管理的 pods。您可以选择列出的任何 pod 以查看有关 pod 的信息。并非所有资源类型都在结构化视图中显示信息。如果您在该资源的页面右上角选择 Raw view（原始视图），您可以看到来自 Kubernetes API 对该资源的完整 JSON 响应。
   - 选择 Cluster（集群）组，然后选择 Nodes（节点）资源类型。您可以看到集群中所有节点的列表。节点可以是任意 Amazon EKS 节点类型 (p. 84)。这与您在为集群选择 Compute（计算）选项时在 Nodes（节点）部分中看到的列表相同。从列表中选择节点资源。在结构化视图中，您还会看到 Pods（容器组（pod））部分。此部分将向您显示在节点上运行的所有 pods。

所需权限

要在 Amazon Web Services Management Console 中查看 Compute（计算）选项卡上的 Resources（资源）选项卡和 Nodes（节点）部分，您登录到 Amazon Web Services Management Console 的用户或者您在登录后切换到的角色必须具有特定的最低 IAM 和 Kubernetes 权限。完成以下步骤以将所需的权限分配给您的用户和角色。

1. 确保 `eks:AccessKubernetesApi` 以及查看 Kubernetes 资源的其他必要 IAM 权限，都分配给了您登录 Amazon Web Services Management Console 所使用的用户，或登录控制台后切换到的角色。有关如何编辑用户权限的更多信息，请参阅《IAM 用户指南》中的更改用户的权限（控制台）。有关如何编辑角色权限策略的信息，请参阅《IAM 用户指南》中的修改角色权限策略（控制台）。

   以下示例策略包括用户或角色查看账户中所有集群的 Kubernetes 资源所需的权限。请将 *111122223333* 替换为您的账户 ID。

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

```
        {
            "Effect": "Allow",
            "Action": [
                "eks:ListFargateProfiles",
                "eks:DescribeNodegroup",
                "eks:ListNodegroups",
                "eks:ListUpdates",
                "eks:AccessKubernetesApi",
                "eks:ListAddons",
                "eks:DescribeCluster",
                "eks:DescribeAddonVersions",
                "eks:ListClusters",
                "eks:ListIdentityProviderConfigs",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:GetParameter",
            "Resource": "arn:aws:ssm:*:111122223333:parameter/*"
        }
    ]
}
```

对于在已连接集群中的节点，Amazon EKS Connector IAM 角色 (p. 433) 应能够模拟集群中的 IAM 用户或角色。这使 Amazon EKS Connector (p. 495) 可以将 IAM 用户或角色映射到 Kubernetes 用户。

2. 创建绑定到 Kubernetes `role` 或 `clusterrole` 的 Kubernetes `rolebinding` 或 `clusterrolebinding`，该角色具有查看 Kubernetes 资源所需的权限。要了解有关 Kubernetes 角色和角色绑定的更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。您可以将以下清单之一应用于创建 `role` 和 `rolebinding` 或者具有必要 Kubernetes 权限的 `clusterrole` 和 `clusterrolebinding` 的集群。

   - 查看所有命名空间中的 Kubernetes 资源 - 文件中的组名称为 `eks-console-dashboard-full-access-group`。使用以下命令将清单应用于集群：

     ```
     kubectl apply -f https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/docs/eks-console-full-access.yaml
     ```

   - 查看特定命名空间中的 Kubernetes 资源 - 此文件中的命名空间为 `default`。文件中的组名为 `eks-console-dashboard-restricted-access-group`。使用以下命令将清单应用于集群：

     ```
     kubectl apply -f https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/docs/eks-console-restricted-access.yaml
     ```

   如果您需要更改 Kubernetes 组名称、命名空间、权限或文件中的任何其他配置，请先下载文件并对其进行编辑，然后再将其应用于集群：

   1. 使用下面的命令之一下载文件：

      ```
      curl -o eks-console-full-access.yaml https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/docs/eks-console-full-access.yaml
      ```

      ```
      curl -o eks-console-restricted-access.yaml https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/docs/eks-console-restricted-access.yaml
      ```

   2. 根据需要编辑文件。
   3. 使用以下命令之一将清单应用于集群：

```
kubectl apply -f eks-console-full-access.yaml
```

```
kubectl apply -f eks-console-restricted-access.yaml
```

3. 将 IAM 用户或角色映射到 aws-auth ConfigMap 中的 Kubernetes 用户或组。您可以使用 eksctl 之类的工具更新 ConfigMap，或者可以通过编辑它来进行手动更新。

   Important

   我们建议使用 eksctl 或者其他工具来编辑 ConfigMap。有关您可以使用的其他工具的信息，请参阅《Amazon EKS 最佳实践指南》中的使用工具对 aws-authConfigMap 进行更改。格式不正确的 aws-auth ConfigMap 可能会导致您失去对集群的访问权限。

   eksctl

   先决条件

   您的设备或 Amazon CloudShell 上安装了 0.110.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 安装或更新 eksctl (p. 10)。

   1. 查看 ConfigMap 中的当前映射。将 *my-cluster* 替换为您的集群名称。将 *region-code* 替换为集群所在的 Amazon Web Services 区域。

      ```
      eksctl get iamidentitymapping --cluster my-cluster --region=region-code
      ```

      输出示例如下。

      ```
      ARN
                      USERNAME                                GROUPS
            ACCOUNT
      arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
      NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
       system:bootstrappers,system:nodes
      ```

   2. 为角色添加映射。此示例假设您已在第一步中将 IAM 权限附加到名为 *my-console-viewer-role* 的角色。请将 *111122223333* 替换为您的账户 ID。

      ```
      eksctl create iamidentitymapping \
          --cluster my-cluster \
          --region=region-code \
          --arn arn:aws:iam::111122223333:role/my-console-viewer-role \
          --group eks-console-dashboard-full-access-group \
          --no-duplicate-arns
      ```

      Important

      角色 ARN 不能包含 role/my-team/developers/my-role 等路径。ARN 的格式必须为 arn:aws:iam::*111122223333*:role/*my-role*。在此示例中，my-team/developers/ 需要删除。

      输出示例如下。

      ```
      ...
      2022-05-09 14:51:20 [#]  adding identity "arn:aws:iam::111122223333:role/my-
      console-viewer-role" to auth ConfigMap
      ```

3. 为用户添加映射。此示例假设您已在第一步中将 IAM 权限附加到名为 *my-user* 的用户。请将 *111122223333* 替换为您的账户 ID。

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user \
    --group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

输出示例如下。

```
...
2022-05-09 14:53:48 [#]  adding identity "arn:aws:iam::111122223333:user/my-user"
 to auth ConfigMap
```

4. 再次查看 ConfigMap 中的映射。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

输出示例如下。

```
ARN
                USERNAME                                GROUPS
               ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:role/my-console-viewer-role
                                                        eks-console-dashboard-
full-access-group
arn:aws:iam::111122223333:user/my-user
                                                        eks-console-dashboard-
restricted-access-group
```

Edit ConfigMap manually

有关将用户或角色添加到 aws-auth ConfigMap 的更多信息，请参阅将 IAM 用户、角色或 Amazon账户添加到 ConfigMap (p. 356)。

1. 打开 ConfigMap 文件进行编辑。

```
kubectl edit -n kube-system configmap/aws-auth
```

2. 将映射添加到 aws-auth ConfigMap，但不要替换任何现有的映射。以下示例添加了在第一步中添加权限的 IAM 用户和角色与在上一步中创建的 Kubernetes 组之间的映射：

   - *my-console-viewer-role* 角色和 eks-console-dashboard-full-access-group。
   - *my-user* 用户和 eks-console-dashboard-restricted-access-group。

   这些示例假设您已在第一步中将 IAM 权限附加到名为 *my-console-viewer-role* 的角色和名为 *my-user* 的用户。请将 *111122223333* 替换为您的账户 ID。

```
apiVersion: v1
data:
mapRoles: |
  - groups:
    - eks-console-dashboard-full-access-group
```

```
    rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
    username: my-console-viewer-role
mapUsers: |
  - groups:
    - eks-console-dashboard-restricted-access-group
    userarn: arn:aws:iam::111122223333:user/my-user
    username: my-user
```

Important

角色 ARN 不能包含 `role/my-team/developers/my-console-viewer-role` 等路径。ARN 的格式必须为 `arn:aws:iam::111122223333:role/my-console-viewer-role`。在此示例中，`my-team/developers/` 需要删除。

3. 保存文件，然后退出编辑器。

Amazon EKS 中的可观察性

您可以使用许多可用的监控或日志记录工具在 Amazon EKS 中观察数据。配置 Fargate 日志路由器后，您的 Amazon EKS 日志数据可以流式传输到 Amazon 服务或合作伙伴工具来进行数据分析。这些 Amazon 服务在 Amazon Web Services Management Console 中提供。您可以使用 Amazon CloudWatch 监控 Amazon EKS 资源。本主题介绍了 Amazon EKS 中的可用功能和格式。Amazon Web Services Management Console 中提供有很多服务，它们提供数据来对您的 Amazon EKS 问题进行排查。有关这些服务的概述，请参阅 Amazon Web Services Management Console 文档。

您可以在 Amazon EKS 控制台的左侧导航窗格中通过在选择 Clusters（集群）后选择集群名称来查看集群运行状况和详细信息。要查看有关部署到集群的任何现有 Kubernetes 资源的详细信息，请参阅 查看 Kubernetes 资源 (p. 458)。

监控是保持 Amazon EKS 和您的 Amazon 解决方案的可靠性、可用性和性能的重要方面。我们建议您从您的 Amazon 解决方案的所有部分收集监控数据。这样，您可以更轻松地调试出现的多点故障。在开始监控 Amazon EKS 之前，请确保有一个监控计划解决以下问题。

- 您的目标是什么？ 如果集群大幅扩展，是否需要实时通知？
- 需要观察哪些资源？
- 您需要多久观察一次这些资源？ 贵公司是否想快速应对风险？
- 您想使用什么工具？ 如果您已经在启动过程中运行 Fargate，则您可以使用内置的日志路由器。
- 您想由谁负责执行监控任务？
- 当出现问题时，您希望向谁发送通知？

Amazon EKS 中的日志记录和监控

Amazon EKS 提供了用于日志记录和监控的内置工具。您可以选择从控制面板开始的日志记录级别。控制面板日志记录记录对集群的所有 API 调用，审计信息，以捕获哪些用户对集群执行了哪些操作，以及基于角色的日志。

Amazon EKS 控制层面日志记录将审计和诊断日志直接从 Amazon EKS 控制层面提供到您账户中的 CloudWatch Logs。这些日志可让您轻松地保护和运行您的集群。您可以选择您需要的确切的日志类型，日志将作为日志流发送到 CloudWatch 中每个 Amazon EKS 集群的组。有关更多信息，请参阅 Amazon EKS 控制面板日志记录 (p. 44)和 Amazon CloudWatch 日志记录。

Note

当您在 Amazon CloudWatch 中检查 Amazon EKS 身份验证器日志时，系统将显示一些文本类似于以下示例文本的条目。

```
level=info msg="mapping IAM role" groups="[]"
 role="arn:aws:iam::111122223333:role/XXXXXXXXXXXXXXXXXX-NodeManagerRole-XXXXXXXX"
 username="eks:node-manager"
```

预期为包含此文本的条目。`username` 是一个 Amazon EKS 内部服务角色，用于为托管节点组和 Fargate 执行特定操作。
对于低级、可自定义的日志记录，Kubernetes 日志记录可用。

Amazon EKS 与 Amazon CloudTrail 集成，后者是在 Amazon EKS 中提供用户、角色或Amazon服务所采取操作的记录的服务。CloudTrail 将 Amazon EKS 的所有 API 调用作为事件捕获。这些捕获包括来自 Amazon EKS 控制台的调用和对 Amazon EKS API 操作的代码调用。有关更多信息，请参阅使用 Amazon CloudTrail 记录 Amazon EKS API 调用 (p. 465)。

Kubernetes API 服务器公开了大量可用于监控和分析的指标。有关更多信息，请参阅Prometheus 的控制层面指标 (p. 381)。

要为自定义 Amazon CloudWatch 日志配置 Fluent Bit，请参阅 Amazon CloudWatch 用户指南中的将 Fluent Bit 设置为 DaemonSet 以将日志发送到 Amazon CloudWatch Logs。

Amazon EKS 日志记录和监控工具

亚马逊云科技为您提供了各种可用于监控 Amazon EKS 的工具。您可以配置一些工具来设置自动监控，但有些工具需要手动调用。我们建议您在环境和现有工具集允许的范围内尽量实现监控任务自动化。

日志记录

领域	工具	日志	设置
应用程序	Amazon CloudWatch Container Insights	它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。	设置过程 要为 ADOT 设置 Amazon CloudWatch Container Insights 日志记录，请使用此过程。
容器	Amazon Fargate	它为容器提供计算容量，包括资源使用量。	设置过程
控制层面	Prometheus	CloudWatch Logs 摄取、归档存储和数据扫描速率适用于已启用的控制层面日志。	设置过程
控制层面	Amazon CloudTrail	它记录用户、角色或服务进行的 API 调用。	设置过程
多个领域	Fargate 日志路由器	将日志流式传输到 Amazon 服务或合作伙伴工具。将 Amazon 用于 Fluent Bit。日志可以流式传输到其他 Amazon 服务或合作伙伴工具。	设置过程

监控

领域	工具	日志	设置
应用程序	Amazon CloudWatch Container Insights	Amazon CloudWatch Container Insights 从容器化应用程序和微服务中收集、	Amazon Web Services Management Console 的设置过程

领域	工具	日志	设置
		聚合和汇总指标与日志。	
应用程序	Amazon Distro for OpenTelemetry (ADOT)	它收集相关的指标、跟踪数据和元数据并将其发送到 Amazon 监控服务或合作伙伴。 它可以通过 Amazon CloudWatch Container Insights 设置。	设置过程
应用程序	Amazon DevOps Guru	它可以检测节点级别的运行性能和可用性。	启用 Amazon CloudWatch Container Insights
应用程序	Amazon X-Ray	它接收有关应用程序的跟踪数据。此跟踪数据包括传入请求和传出请求以及有关请求的元数据。对于 Amazon EKS，实施需要 OpenTelemetry 附加组件。	设置过程
Cluster	Amazon CloudWatch	它会记录集群资源。	不适用
容器	Amazon Fargate	它为容器提供计算容量，包括资源使用量。	设置过程

使用 Amazon CloudTrail 记录 Amazon EKS API 调用

Amazon EKS 与 Amazon CloudTrail 集成，后者是在 Amazon EKS 中提供用户、角色或 Amazon 服务所采取操作的记录的服务。CloudTrail 将 Amazon EKS 的所有 API 调用作为事件捕获。这包含来自 Amazon EKS 控制台的调用和对 Amazon EKS API 操作的代码调用。

如果您创建了跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶。这包括 Amazon EKS 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，您可以确定有关请求的几个详细信息。例如，您可以确定何时向 Amazon EKS 发出了请求，发出请求的 IP 地址以及何人发出的请求。

要了解有关 CloudTrail 的更多信息，请参阅《Amazon CloudTrail 用户指南》。

CloudTrail 中的 Amazon EKS 信息

在您创建 Amazon 账户时，还将在您的 Amazon 账户上启用 CloudTrail。当 Amazon EKS 中发生任何活动时，该活动将记录在 CloudTrail 事件中，并与其他 Amazon 服务事件一同保存在 Event history（事件历史记

录）中。您可以在 Amazon 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录Amazon账户中的事件（包括 Amazon EKS 的事件），请创建跟踪记录。通过跟踪记录，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有Amazon Web Services 区域。此跟踪记录来自 Amazon Web Services 区域 分区中所有 Amazon 的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，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行动。有关详细信息，请参阅以下资源：

- 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

所有 Amazon EKS 操作都由 CloudTrail 记录，并记录在 Amazon EKS API 参考中。例如，对 `CreateCluster`、`ListClusters` 和 `DeleteCluster` 部分的调用将在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。您可以使用身份信息确定以下信息。

- 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。

了解 Amazon EKS 日志文件条目

跟踪记录是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作的信息。其中包括操作的日期和时间以及所使用的请求参数等信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `CreateCluster` 操作。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/username",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "username"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "region-code",
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/6.4.0",
  "requestParameters": {
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-a670c2df",
        "subnet-4f8c5004"
      ]
```

```
    },
    "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-CAC1G1VH3ZKZ",
    "clusterName": "test"
  },
  "responseElements": {
    "cluster": {
      "clusterName": "test",
      "status": "CREATING",
      "createdAt": 1527535003.208,
      "certificateAuthority": {},
      "arn": "arn:aws:eks:region-code:111122223333:cluster/test",
      "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-CAC1G1VH3ZKZ",
      "version": "1.10",
      "resourcesVpcConfig": {
        "securityGroupIds": [],
        "vpcId": "vpc-21277358",
        "subnetIds": [
          "subnet-a670c2df",
          "subnet-4f8c5004"
        ]
      }
    }
  },
  "requestID": "a7a0735d-62ab-11e8-9f79-81ce5b2b7d37",
  "eventID": "eab22523-174a-499c-9dd6-91e7be3ff8e3",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

Amazon EKS 服务相关角色的日志条目

Amazon EKS 服务相关角色对Amazon资源进行 API 调用。将显示由 Amazon EKS 服务相关角色进行的调用的 CloudTrail 日志条目以及 `username: AWSServiceRoleForAmazonEKS` 和 `username: AWSServiceRoleForAmazonEKSNodegroup`。有关 Amazon EKS 和服务相关角色的更多信息，请参阅对 Amazon EKS 使用服务相关角色 (p. 418)。

以下示例显示了一个 CloudTrail 日志条目，该条目演示 `AWSServiceRoleForAmazonEKSNodegroup` 服务相关角色执行的 `DeleteInstanceProfile` 操作（在 `sessionContext` 中注明）。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3WHGPEZ7SJ2CW55C5:EKS",
        "arn": "arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForAmazonEKSNodegroup/
EKS",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA3WHGPEZ7SJ2CW55C5",
                "arn": "arn:aws:iam::111122223333:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAmazonEKSNodegroup"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-02-26T00:56:33Z"
            }
```

```
        },
        "invokedBy": "eks-nodegroup.amazonaws.com"
    },
    "eventTime": "2020-02-26T00:56:34Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "DeleteInstanceProfile",
    "awsRegion": "region-code",
    "sourceIPAddress": "eks-nodegroup.amazonaws.com",
    "userAgent": "eks-nodegroup.amazonaws.com",
    "requestParameters": {
        "instanceProfileName": "eks-11111111-2222-3333-4444-abcdef123456"
    },
    "responseElements": null,
    "requestID": "11111111-2222-3333-4444-abcdef123456",
    "eventID": "11111111-2222-3333-4444-abcdef123456",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

对 ADOT Operator 的 Amazon EKS 附加组件支持

Amazon EKS 支持使用 Amazon Web Services Management Console、Amazon CLI 和 Amazon EKS API 来安装和管理 Amazon Distro for OpenTelemetry (ADOT) Operator。这样就可以简化体验，用于检测在 Amazon EKS 上运行的应用程序，以便将指标和跟踪数据发送到 Amazon CloudWatch、Prometheus 和 X-Ray 等多个监控服务选项。本主题介绍如何开始将 ADOT 用于 Amazon EKS 集群上运行的应用程序。

- 先决条件 (p. 468)
- 创建 IAM 角色 (p. 469)
- 安装附加组件 (p. 470)
- 为您的首选监控选项部署 ADOT 收集器 (p. 472)
- 部署示例应用程序 (p. 474)以生成 OTLP 数据

安装 ADOT 之前，请确保您阅读并理解考虑因素 (p. 468)和先决条件 (p. 468)。

ADOT 考虑因素

在使用 ADOT 之前，请查看以下考虑因素和先决条件：

- ADOT Operator 尚不支持 arm64 架构。
- 连接的集群不能使用此附加组件。

Amazon Distro for OpenTelemetry (ADOT) 的先决条件

安装 Amazon Distro for OpenTelemetry (ADOT) 附加组件之前，您必须满足以下先决条件。

- 授予 Amazon EKS 附加组件安装 ADOT 的权限：

```
kubectl apply -f https://amazon-eks.s3.amazonaws.com/docs/addons-otel-permissions.yaml
```

- 您的 Amazon EKS 集群必须使用 Kubernetes 版本 `1.19` 或者更高版本。您可以使用以下命令验证版本。要更新集群，请参阅 更新 Amazon EKS 集群 Kubernetes 版本 (p. 28)。

```
kubectl version | grep "Server Version"
```

- 满足 TLS 证书要求 (p. 469) 以确保端到端加密。

TLS 证书要求

ADOT Operator 使用准入 Webhook 更改和验证收集器自定义资源（CR）请求。在 Kubernetes 中，Webhook 需要 API 服务器配置为信任的 TLS 证书。您可以通过多种方式生成所需的 TLS 证书。但是，默认方法是手动安装最新版本的 cert-manager。cert-manager 会生成自签名证书。

安装 cert-manager

安装 cert-manager

1. 使用以下命令安装 cert-manager。这会创建允许端到端加密的必要 cert-manager 对象。必须对将安装 ADOT 的每个集群执行此操作。

```
kubectl apply -f \
https://github.com/cert-manager/cert-manager/releases/download/v1.8.2/cert-manager.yaml
```

2. 请使用以下命令验证 cert-manager 是否已准备就绪。

```
kubectl get pod -w -n cert-manager
```

输出示例如下。

```
NAME READY  STATUS RESTARTS  AGE
cert-manager-5597cff495-mnb2p  1/1  Running  0 12d
cert-manager-cainjector-bd5f9c764-8jp5g  1/1  Running  0 12d
cert-manager-webhook-5f57f59fbc-h9st8  1/1  Running  0 12d
```

创建 IAM 角色

您安装 Amazon Distro for OpenTelemetry (ADOT) (p. 470)所在的每个集群必须让此角色来授予您的 Amazon 服务账户权限。按照以下步骤创建 IAM 角色并使用 IRSA (p. 395) 将其与 Amazon EKS 服务账户关联：

1. 为集群创建 IAM OIDC 提供商 (p. 396).
2. 创建您的服务账户和 IAM 角色。请注意此命令中包含以下标志：
   - 对于 `--name` 标志，指定您要创建的服务账户的名称。在本例中，它是 *`adot-collector`*。
   - 对于 `--namespace` 标志，指定您的服务帐户将驻留的命名空间；在此示例中，您将使用 *`default`* 命名空间。
   - 对于 `--cluster` 标志，指定集群的名称。
   - 使用 `--attach-policy-arn` 参数指定您用于角色的集成的托管 IAM policy。例如，如果您使用 ADOT Collector 向 CloudWatch 发送指标数据，则指定 `CloudWatchAgentServerPolicy` 托管策略。
   - `--override-existing-serviceaccounts` 标志适用于已在集群中创建的但没有 IAM 角色的服务账户。如果该服务账户没有 IAM 角色，您可以排除此标志。

     考虑到上述注释，输入以下命令：

```
eksctl create iamserviceaccount \
    --name adot-collector \
```

```
    --namespace default \
    --cluster my-cluster \
    --attach-policy-arn arn:aws:iam::aws:policy/AmazonPrometheusRemoteWriteAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
    --approve \
    --override-existing-serviceaccounts
```

在以下收集器配置中，已添加配置的 `serviceAccount: adot-collector` 字段以使用 IRSA。

管理 Amazon Distro for OpenTelemetry Operator

Amazon Distro for OpenTelemetry (ADOT) Operator 作为 Amazon EKS 附加组件提供。安装 ADOT Operator 后，您可以将 ADOT 收集器配置为指定部署类型和接收应用程序指标或跟踪数据的服务。本主题介绍如何管理 ADOT 附加组件。

- 安装 Amazon Distro for OpenTelemetry (ADOT) Operator (p. 470)
- 更新 Amazon Distro for OpenTelemetry (ADOT) Operator (p. 471)
- 删除 Amazon Distro for OpenTelemetry (ADOT) Operator (p. 472)

要安装 ADOT 收集器，请参阅 部署 Amazon Distro for OpenTelemetry 收集器 (p. 472)。

安装 Amazon Distro for OpenTelemetry (ADOT) Operator

安装 ADOT 附加组件包括 ADOT Operator，该组件反过来会部署 ADOT 收集器。ADOT Operator 是一个自定义控制器，它通过 CustomResourceDefinition（CRD） 引入了一种名为 `OpenTelemetryCollector` 的新对象类型。当 ADOT Operator 检测到存在 `OpenTelemetryCollector` 资源时，将会安装 ADOT 收集器。

先决条件

- 您已满足 ADOT 先决条件 (p. 468)。
- 已安装了 kubectl。
- 必要时，使用以下命令更新您的 `kubeconfig`。

```
aws eks update-kubeconfig --name my-cluster --region region-code
```

- 已安装了 eksctl。
- Amazon CLI 版本 2 已安装。
- 现有 Amazon EKS 集群。
- 现有 Amazon EKS 服务 IAM 角色。如果您没有角色，则可以按照 创建 IAM 角色 (p. 469) 中的步骤创建一个。

Amazon Web Services Management Console

使用以下步骤将 ADOT Amazon EKS 附加组件安装到 Amazon EKS 集群中：

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 在左侧窗格中，选择 Clusters（集群），然后在 Clusters（集群）页面上选择集群的名称。
3. 选择 Add-ons（附加组件）选项卡。
4. 单击 Add new（新增），然后从下拉列表中选择 Amazon Distro for OpenTelemetry。

5. 默认版本将在 Version（版本）下拉列表中选择。如果已在没有 IAM 角色的集群中创建服务账户，则单击 Override existing configuration for this add-on on the cluster（覆盖集群上此附加组件的现有配置）。
6. 单击 Add (添加)。

Amazon CLI

1. 使用命令将 ADOT Amazon EKS 附加组件安装到 Amazon EKS 集群中：

```
aws eks create-addon --addon-name adot --cluster-name my-cluster
```

在完成前，`status` 字段值将为 `CREATING`。

2. 使用命令验证 ADOT 是否已安装并运行：

```
aws eks describe-addon --addon-name adot --cluster-name my-cluster
```

当创建完成时，您将看到 `"status": "ACTIVE"`。

更新 Amazon Distro for OpenTelemetry (ADOT) Operator

Amazon EKS 不会自动更新您的集群上的 ADOT。您必须启动更新，然后 Amazon EKS 会为您更新 Amazon EKS 附加组件。

使用 Amazon CLI 更新 ADOT Amazon EKS 附加组件

1. 检查 ADOT 附加组件的当前版本。将 *my-cluster* 替换为您的集群名称。

```
aws eks describe-addon \
    --cluster-name my-cluster \
    --addon-name adot \
    --query "addon.addonVersion" \
    --output text
```

2. 确定集群版本支持的 ADOT 版本是否可用。

```
aws eks describe-addon-versions \
    --addon-name adot \
    --kubernetes-version 1.19 \
    --query "addons[].addonVersions[].[addonVersion, compatibilities[].defaultVersion]"
 \
    --output text
```

输出示例如下。

```
1.8.0
True
1.7.0
False
```

下面带有 `True` 的版本是使用新集群部署的默认版本。如有必要，可以返回和更新以后的版本。

3. 要更新 ADOT 版本，请使用以下命令并将 `addon-version` 值替换为所需版本。

```
aws eks update-addon \
    --cluster-name my-cluster \
```

```
    --addon-name adot \
    --addon-version 1.8.0 \
    --resolve-conflicts OVERWRITE
```

Note

`--resolve-conflicts OVERWRITE` 选项将通过覆盖您的设置来解决与 Amazon EKS 附加组件设置的任何冲突。确保您的设置不需要管理。如果没有此选项，必须手动解决冲突。有关更多信息，请参阅Amazon EKS 附加组件配置 (p. 342)。

删除 Amazon Distro for OpenTelemetry (ADOT) Operator

- 您必须将 ADOT 收集器资源与 ADOT 收集器分开删除。在此命令中，指定用于部署 ADOT 收集器的 YAML 文件：

```
kubectl delete -f collector-config.yaml
```

- 您可以通过 Amazon CLI 或 `eksctl` 删除 ADOT Operator。如果删除 ADOT Operator，则必须再次遵循安装说明 (p. 470)重新安装：

  CLI

```
aws eks delete-addon --addon-name adot --cluster-name my-cluster
```

  `eksctl`

```
eksctl delete addon --cluster my-cluster --name adot
```

部署 Amazon Distro for OpenTelemetry 收集器

一旦安装并运行 Amazon Distro for OpenTelemetry (ADOT) Operator (p. 470) 后，您就可以将 ADOT 收集器部署到 Amazon EKS 集群中。ADOT 收集器可以以四种模式之一进行部署：Deployment、Daemonset、StatefulSet 和 Sidecar，并且可以由单个服务进行可见性部署。有关这些部署模式的更多信息，请参阅 Github 上的 ADOT 收集器安装说明。

您可以通过将 YAML 文件应用于集群来部署 ADOT 收集器。您应该已经安装 ADOT Operator (p. 470)。您可以 创建 IAM 角色 (p. 469) 使用 IRSA (p. 395) 的 Amazon EKS 服务账户。通过这一操作，您的服务账户可向使用它的任意 pod 中的容器提供 Amazon 权限。

我们为以下服务（在部署模式中）提供了 ADOT 收集器示例：

- Prometheus (p. 472)
- Amazon CloudWatch (p. 473)
- X-Ray (p. 473)

在部署模式下为 Amazon Managed Service for Prometheus 部署 Amazon Distro for OpenTelemetry 收集器

以下过程使用以 `deployment` 作为 `mode` 值的示例 YAML 文件。这是默认模式，与独立应用程序类似地部署 ADOT 收集器。此配置接收来自示例应用程序和从集群上的 pods 中抓取的 Amazon Managed Service for Prometheus 指标中的 OTLP 指标。您可以将 `mode` 更改为 `Daemonset`、`StatefulSet` 和 `Sidecar`，具

体取决于您的部署策略。对于 Daemonset 部署，请参阅 ADOT 文档中的 Amazon Managed Prometheus 的高级收集器配置，或者单击此处下载 DaemonSet 示例。

1. 将 collector-config-amp.yaml 文件下载到您的计算机中。您还可以在 GitHub 上查看该文件。

```
curl -o collector-config-amp.yaml https://raw.githubusercontent.com/aws-observability/aws-otel-community/master/sample-configs/operator/collector-config-amp.yaml
```

2. 在 collector-config-amp.yaml 中，将以下内容替换为您自己的值：
   - mode: *deployment*
   - serviceAccount: *adot-collector*
   - endpoint: "*<YOUR_REMOTE_WRITE_ENDPOINT>*"
   - region: "*<YOUR_AWS_REGION>*"
   - name: *adot-collector*
3. 将 YAML 文件应用到集群以部署 ADOT 收集器：

```
kubectl apply -f collector-config-amp.yaml
```

为 CloudWatch 部署 Amazon Distro for OpenTelemetry 收集器

可以部署 Amazon Distro for OpenTelemetry (ADOT) 收集器以接收导出到 Amazon CloudWatch 的 OTLP 指标。此过程将下载一个示例 YAML 文件，您可以将其应用到集群。

1. 将 collector-config-cloudwatch.yaml 文件下载到您的计算机中。您还可以在 GitHub 上查看该文件。

```
curl -o collector-config-cloudwatch.yaml https://raw.githubusercontent.com/aws-observability/aws-otel-community/master/sample-configs/operator/collector-config-cloudwatch.yaml
```

2. 在 collector-config-cloudwatch.yaml 中，将以下内容替换为您自己的值：
   - mode: *deployment*
   - serviceAccount: *adot-collector*
   - value: *<YOUR_EKS_CLUSTER_NAME>*
   - region: "*<YOUR_AWS_REGION>*"
   - name: *adot-collector*
3. 将 YAML 文件应用到集群以部署 ADOT 收集器：

```
kubectl apply -f collector-config-cloudwatch.yaml
```

4. （可选）通过打开 Amazon CloudWatch 控制台验证指标数据是否被发送到 Amazon CloudWatch，并打开左侧的 Metrics（指标）菜单。选择 All metrics（所有指标），然后单击 custom namespaces（自定义命名空间）下的 AOCDockerDemo/AOCDockerDemoService 框。您可以通过选择任何分组来查看任何指标数据。

为 X-Ray 部署 Amazon Distro for OpenTelemetry (ADOT) 收集器

您可以部署 Amazon Distro for OpenTelemetry (ADOT) 收集器以通过应用 YAML 配置文件将跟踪发送到 X-Ray。此过程将下载一个示例 YAML 文件，您可以将其应用到集群。

1. 将 collector-config-xray.yaml 文件下载到您的计算机中。您还可以在 GitHub 上查看该文件。

```
curl -o collector-config-xray.yaml https://raw.githubusercontent.com/aws-observability/
aws-otel-community/master/sample-configs/operator/collector-config-xray.yaml
```

2. 在 collector-config-xray.yaml 中，将以下内容替换为您自己的值：
   - mode: *deployment*
   - serviceAccount: *adot-collector*
   - region: "*<YOUR_AWS_REGION>*"
3. 将 YAML 文件应用到集群以部署 ADOT 收集器：

```
kubectl apply -f collector-config-xray.yaml
```

4. （可选）通过打开 X-Ray 控制台验证跟踪数据是否正发送到 X-Ray，然后从左侧菜单中选择 Traces。您的跟踪数据应该存在。

部署示例应用程序以测试 Amazon Distro for OpenTelemetry Collector

示例应用程序将生成 OTLP 数据并将其发送到您通过 Amazon Distro for OpenTelemetry (ADOT) Collector 部署 (p. 472)配置的任何服务。如果您已经在集群内运行了可生成数据的应用程序，则此步骤是可选的。请参阅应用程序的文档，以确保将数据发送到正确的端点。

示例应用程序和流量生成器在很大程度上取自 ADOT 收集器存储库中的示例。一个 docker-compose.yaml 文件已使用 Kompose 工具转换为 Kubernetes 资源。

要应用流量生成器和示例应用程序，请执行以下步骤。

1. 将 traffic-generator.yaml 文件下载到您的计算机中。您还可以在 GitHub 上查看该文件。

```
curl -o traffic-generator.yaml https://raw.githubusercontent.com/aws-observability/aws-
otel-community/master/sample-configs/traffic-generator.yaml
```

2. 在 traffic-generator.yaml 中，确保 kind 值反映了您的部署模式：

```
kind: Deployment
```

   traffic-generator.yaml 对 Kubernetes 服务 sample-app:4567 进行 http 调用。这允许流量生成器在端口 4567 上与示例应用程序进行交互。sample-app 会解析为 sample-app pod 的 IP 地址。
3. 将 traffic-generator.yaml 应用到集群。

```
kubectl apply -f traffic-generator.yaml
```

4. 将 sample-app.yaml 文件下载到您的计算机中。您还可以在 GitHub 上查看该文件。

```
curl -o sample-app.yaml https://raw.githubusercontent.com/aws-observability/aws-otel-
community/master/sample-configs/sample-app.yaml
```

5. 在 sample-app.yaml 中，将以下内容替换为您自己的 Amazon Web Services 区域：

```
value: "<YOUR_AWS_REGION>"
```

   sample-app.yaml 定义了以下操作：

- 服务资源配置 `port: 4567` 以允许对流量生成器进行 HTTP 请求。
- 部署资源配置了一些环境变量：
  - `LISTEN_ADDRESS` 针对流程生成器中的 HTTP 请求配置为 `0.0.0.0:4567`。
  - `OTEL_EXPORTER_OTLP_ENDPOINT` 的值为 `http://my-collector-collector:4317`。`my-collector-collector` 是 Kubernetes 服务的名称，该服务允许示例应用程序在端口 `4317` 上与 ADOT 收集器进行交互。在 ADOT 收集器配置中，ADOT 收集器从端点接收指标和跟踪信息：`0.0.0.0:4317`。

6. 将 `sample-app.yaml` 应用到集群。

```
kubectl apply -f sample-app.yaml
```

已与 Amazon EKS 集成的Amazon服务

Amazon EKS 可与其他Amazon服务配合使用来提供应对您的业务挑战的更多解决方案。本主题介绍了使用 Amazon EKS 添加功能的服务或 Amazon EKS 用于执行任务的服务。

主题

- 利用 Amazon CloudFormation 创建 Amazon EKS 资源 (p. 476)
- Amazon EKS 和 Amazon 本地区域 (p. 476)
- Deep Learning Containers (p. 477)

利用 Amazon CloudFormation 创建 Amazon EKS 资源

Amazon EKS 与 Amazon CloudFormation 集成，后者是一项服务，可帮助您对Amazon资源进行建模和设置，这样您只需花较少的时间来创建和管理资源与基础设施。您创建一个描述您所需的所有Amazon资源（如 Amazon EKS 集群）的模板，而 Amazon CloudFormation 将负责为您设置和配置这些资源。

在您使用 Amazon CloudFormation 时，可重复使用您的模板来不断地重复设置您的 Amazon EKS 资源。仅描述您的资源一次，然后在多个Amazon账户和区域中反复配置相同的资源。

Amazon EKS 和 Amazon CloudFormation 模板

要为 Amazon EKS 和相关服务预置和配置资源，您必须了解 Amazon CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述要在 Amazon CloudFormation 堆栈中调配的资源。如果您不熟悉 JSON 或 YAML，可以在 Amazon CloudFormation Designer 的帮助下开始使用 Amazon CloudFormation 模板。有关更多信息，请参阅 Amazon CloudFormation 用户指南中的什么是 Amazon CloudFormation Designer？。

Amazon EKS 支持在 Amazon CloudFormation 中创建集群和节点组。有关更多信息（包括您的 Amazon EKS 资源的 JSON 和 YAML 模板示例），请参阅 Amazon CloudFormation 用户指南中的 Amazon EKS 资源类型参考。

了解有关 Amazon CloudFormation 的更多信息

要了解有关 Amazon CloudFormation 的更多信息，请参阅以下资源：

- Amazon CloudFormation
- Amazon CloudFormation 用户指南
- Amazon CloudFormation 命令行界面用户指南

Amazon EKS 和 Amazon 本地区域

Important

此功能在 Amazon Web Services 区域（中国）不可用。

Amazon Local Zone 是在地理上靠近用户的 Amazon Web Services 区域的扩展。Local Zones 有自己的 Internet 连接并支持 Amazon Direct Connect。在 Local Zones 中创建的资源可以通过低延迟通信服务于本地用户。有关更多信息，请参阅 Local Zones。

Amazon EKS 支持本地区域中的某些资源。这包括自行管理的 Amazon EC2 节点 (p. 106)、Amazon EBS 卷以及应用程序负载均衡器（ALB）。将本地区域用作 Amazon EKS 集群的一部分时，我们建议您考虑以下几个因素。

节点

您无法在本地区域中使用 Amazon EKS 创建托管的节点组或 Fargate 节点。但是，您可以使用 Amazon EC2 API、Amazon CloudFormation 或 `eksctl` 在本地区域中创建自行管理的 Amazon EC2 节点。有关更多信息，请参阅自行管理的节点 (p. 106)。

网络架构

- Amazon EKS 托管的 Kubernetes 控制面板始终在 Amazon Web Services 区域 中运行。Amazon EKS 托管的 Kubernetes 控制面板不能在本地区域中运行。由于本地区域在 VPC 中显示为子网，因此 Kubernetes 会将您的本地区域资源视为该子网的一部分。
- Amazon EKS Kubernetes 集群使用 Amazon EKS 托管的弹性网络接口，与您在 Amazon Web Services 区域 或本地区域中运行的 Amazon EC2 实例进行通信。要了解有关 Amazon EKS 联网架构的更多信息，请参阅 Amazon EKS 联网 (p. 226)。
- 与区域子网不同，Amazon EKS 不能将网络接口放置到您的 Local Zones 子网中。这意味着您在创建集群时一定不要指定 Local Zones 子网。

Deep Learning Containers

Amazon 深度学习容器是 Amazon EKS 和 Amazon Elastic Container Service (Amazon ECS) 上 TensorFlow 中训练和服务模型的一组 Docker 镜像。Deep Learning Containers 提供具有 TensorFlow、Nvidia CUDA（适用于 GPU 实例）和 Intel MKL（适用于 CPU 实例）库的优化环境，其在 Amazon ECR 中可用。

要开始使用 Amazon EKS 上的 Amazon Deep Learning Containers，请参阅 Amazon Deep Learning AMI 开发人员指南中的 Amazon EKS 上的 Amazon Deep Learning Containers。

Amazon EKS 故障排除

本章介绍使用 Amazon EKS 时可能遇到的一些常见错误以及相应的错误处理方式。

容量不足

如果您在尝试创建 Amazon EKS 集群时收到以下错误，则表示所指定的某个可用区容量不足，无法支持集群。

```
Cannot create cluster 'example-cluster' because region-1d, the targeted
Availability Zone, does not currently have sufficient capacity to support the
cluster. Retry and choose from these Availability Zones: region-1a, region-1b,
region-1c
```

在集群 VPC 中使用此错误消息所返回的可用区中托管的子网重新尝试创建集群。

节点未能加入集群

有几种常见原因会阻止节点加入集群：

- aws-auth-cm.yaml (p. 355) 文件没有用于节点的正确 IAM 角色 ARN。确保在 aws-auth-cm.yaml 文件中指定了节点 IAM 角色 ARN（非实例配置文件 ARN）。有关更多信息，请参阅启动自行管理的 Amazon Linux 节点 (p. 107)。
- 节点 Amazon CloudFormation 模板中的 ClusterName 与您希望节点加入的集群的名称不完全匹配。将不正确的值传递到此字段会导致节点的 /var/lib/kubelet/kubeconfig 文件配置不正确，并且节点将无法加入集群。
- 节点不会标记为由集群拥有。您的节点必须应用了以下标签，其中的 *my-cluster* 替换为集群的名称。

密钥	值
kubernetes.io/cluster/*my-cluster*	owned

- 节点可能无法使用公有 IP 地址访问集群。确保向部署在公有子网中的节点分配了公有 IP 地址。如果没有分配，您可以在节点启动后为其关联弹性 IP 地址。有关更多信息，请参阅将弹性 IP 地址与正在运行的实例或网络接口关联。如果公有子网未设置为自动将公有 IP 地址分配给部署到其中的实例，我们建议启用该设置。有关更多信息，请参阅修改子网的公有 IPv4 寻址属性。如果节点部署到私有子网，则该子网必须具有指向分配了公有 IP 地址的 NAT 网关的路由。
- 您的账户未启用节点部署所在 Amazon Web Services 区域的 STS 端点。要启用该区域，请参阅在 Amazon Web Services 区域中激活和停用 Amazon STS。
- Worker 节点没有私有 DNS 条目，从而导致 kubelet 日志中包含 node "" not found 错误。确保创建工作线程节点的 VPC 在 DHCP options set 中以 Options 的形式设置了 domain-name 和 domain-name-servers 的值。默认值为 domain-name:<region>.compute.internal 和 domain-name-servers:AmazonProvidedDNS。有关更多信息，请参阅 Amazon VPC 用户指南 中的 DHCP 选项集。

要确定导致 Worker 节点无法加入集群的常见原因并进行问题排查，您可以使用 AWSSupport-TroubleshootEKSWorkerNode 运行手册。有关更多信息，请参阅 Amazon Systems Manager Automation 运行手册参考中的 AWSSupport-TroubleshootEKSWorkerNode。

未经授权或访问被拒绝 (kubectl)

如果您在运行 kubectl 命令时收到以下错误之一，则说明您的 kubectl 未针对 Amazon EKS 正确配置，或您使用的 IAM 用户或角色凭证未映射到 Amazon EKS 集群中具有足够权限的 Kubernetes RBAC 用户。

- could not get token: AccessDenied: Access denied
- error: You must be logged in to the server (Unauthorized)
- error: the server doesn't have a resource type "svc"

这可能是因为集群是使用一组Amazon凭证（来自 IAM 用户或角色）创建的，而 kubectl 使用的是另一组凭证。

创建 Amazon EKS 集群后，创建集群的 IAM 实体（用户或角色）将作为管理员（具有 system:masters 权限）添加到 Kubernetes RBAC 授权表中。最初，仅该 IAM 用户可以使用 kubectl 调用 Kubernetes API 服务器。有关更多信息，请参阅让 IAM 用户和角色有权访问您的集群 (p. 355)。如果使用控制台创建集群，则必须确保在集群上运行 kubectl 命令时，相同的 IAM 用户凭证位于Amazon开发工具包凭证链中。

如果安装和配置 Amazon CLI，则可为用户配置 IAM 凭证。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的配置 Amazon CLI。

如果您已代入一个角色来创建 Amazon EKS 集群，则必须确保 kubectl 已配置为代入相同的角色。使用以下命令更新您的 kubeconfig 文件以使用 IAM 角色。有关更多信息，请参阅为 Amazon EKS 创建 kubeconfig (p. 365)。

```
aws eks update-kubeconfig \
    --region region-code \
    --name my-cluster \
    --role-arn arn:aws:iam::111122223333:role/role_name
```

要将 IAM 用户映射到 Kubernetes RBAC 用户，请参阅 让 IAM 用户和角色有权访问您的集群 (p. 355)。

aws-iam-authenticator 未找到

如果收到错误 "aws-iam-authenticator": executable file not found in $PATH，则表示未对 Amazon EKS 配置 kubectl。有关更多信息，请参阅安装 aws-iam-authenticator (p. 369)。

Note

如果安装了 Amazon CLI 版本 1.16.156 或更高版本，则不需要 aws-iam-authenticator。

hostname doesn't match

系统的 Python 版本必须为 2.7.9 或更高版本。否则，在 Amazon CLI 调用 Amazon EKS 时会收到 hostname doesn't match 错误。有关更多信息，请参阅“Python 请求常见问题”中的什么是“主机名不匹配”错误？。

getsockopt: no route to host

Docker 在 Amazon EKS 集群中的 172.17.0.0/16 CIDR 范围内运行。我们建议您的集群的 VPC 子网不重叠此范围。否则，您将收到以下错误：

```
Error: : error upgrading connection: error dialing backend: dial tcp
 172.17.<nn>.<nn>:10250: getsockopt: no route to host
```

托管节点组错误

如果您在 Amazon Web Services Management Console中收到“Instances failed to join the kubernetes cluster (实例无法加入 kubernetes 集群)”错误，请确保已启用集群的私有终端节点访问，或者您已正确配置 CIDR 块以用于公有终端节点访问。有关更多信息，请参阅Amazon EKS 集群端点访问控制 (p. 38)。

如果您的托管节点组遇到硬件运行状况问题，则 Amazon EKS 返回错误消息以帮助您诊断问题。这些运行状况检查无法检测到软件问题，因为它们基于 Amazon EC2 运行状况检查。以下列表展示了错误消息及其相关描述。

- AccessDenied：Amazon EKS 或您的一个或多个托管节点无法向 Kubernetes 集群 API 服务器进行身份验证或授权。有关解决该错误的更多信息，请参阅 修复托管节点组的 AccessDenied 错误 (p. 480)。
- AmiIdNotFound：找不到与您的启动模板关联的 AMI ID。确保 AMI 存在并与您的账户共享。
- AutoScalingGroupNotFound：找不到与托管节点组关联的 Auto Scaling 组。您可以重新创建具有相同设置的 Auto Scaling 组进行恢复。
- ClusterUnreachable：Amazon EKS 或您的一个或多个托管节点无法与 Kubernetes 集群 API 服务器通信。如果存在网络中断或 API 服务器处理请求时超时，则可能会发生这种情况。
- Ec2SecurityGroupNotFound：找不到集群的集群安全组。您必须重新创建集群。
- Ec2SecurityGroupDeletionFailure：无法删除托管节点组的远程访问安全组。从安全组中删除所有依赖关系。
- Ec2LaunchTemplateNotFound：找不到托管节点组的 Amazon EC2 启动模板。您必须重新创建节点组才能恢复。
- Ec2LaunchTemplateVersionMismatch：您托管节点组的 Amazon EC2 启动模板版本与 Amazon EKS 创建的版本不匹配。您可以恢复到 Amazon EKS 创建的版本以进行恢复。
- IamInstanceProfileNotFound：找不到您的托管节点组的 IAM 实例配置文件。您可以重新创建具有相同设置的实例配置文件进行恢复。
- IamNodeRoleNotFound：找不到您的托管节点组的 IAM 角色。您可以重新创建具有相同设置的 IAM 角色进行恢复。
- AsgInstanceLaunchFailures：您的 Auto Scaling 组在尝试启动实例时出现故障。
- NodeCreationFailure：您启动的实例无法注册到 Amazon EKS 集群。此故障的常见原因是节点 IAM 角色 (p. 427)权限不足，或节点缺少出站 Internet 访问权限。您的节点必须符合以下要求之一：
  - 能够使用公有 IP 地址访问互联网。与节点所在子网关联的安全组必须允许进行通信。有关更多信息，请参阅 the section called “子网要求和注意事项” (p. 227) 和 the section called “安全组要求” (p. 231)。
  - 节点和 VPC 必须满足 the section called “私有集群要求” (p. 56) 中的要求。
- InstanceLimitExceeded：您的Amazon账户无法启动指定实例类型的更多实例。您可以请求提高 Amazon EC2 实例限制以进行恢复。
- InsufficientFreeAddresses：与您托管节点组关联的一个或多个子网没有足够的 IP 地址供新节点使用。
- InternalFailure：这些错误一般由 Amazon EKS 服务器端问题导致。

修复托管节点组的 AccessDenied 错误

在托管节点组上执行操作时最常见的 AccessDenied 错误原因是缺少 eks:node-manager ClusterRole 或 ClusterRoleBinding。Amazon EKS 会在托管节点组启动的过程中在您的集群中设置这些资源，这些资源是管理节点组所必需的。

ClusterRole 可能随着时间的推移而改变，但它应该类似于以下示例：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
```

```
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
```

ClusterRoleBinding 可能随着时间的推移而改变，但它应该类似于以下示例：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

验证 eks:node-manager ClusterRole 是否存在。

```
kubectl describe clusterrole eks:node-manager
```

如果存在，则将输出与上一个 ClusterRole 示例进行比较。

验证 eks:node-manager ClusterRoleBinding 是否存在。

```
kubectl describe clusterrolebinding eks:node-manager
```

如果存在，则将输出与上一个 ClusterRoleBinding 示例进行比较。

如果您发现在请求托管节点组操作期间，由于 ClusterRole 或 ClusterRoleBinding 缺失或损坏造成了 AcessDenied 错误，则可还原它们。将以下内容保存到名为 *eks-node-manager-role.yaml* 的文件中。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
```

```
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

应用文件。

```
kubectl apply -f eks-node-manager-role.yaml
```

重试节点组操作，查看是否解决了您的问题。

CNI 日志收集工具

Amazon VPC CNI plugin for Kubernetes 具有自己的问题排查脚本，该脚本可在 `/opt/cni/bin/aws-cni-support.sh` 的节点上找到。您可以使用该脚本收集有关支持案例和常规故障排除的诊断日志。

使用以下命令可在您的节点上运行脚本：

```
sudo bash /opt/cni/bin/aws-cni-support.sh
```

> **Note**
>
> 如果脚本在该位置不存在，CNI 容器将无法运行。可以使用以下命令手动下载并运行脚本：
>
> ```
> curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/log-collector-script/linux/eks-log-collector.sh
> sudo bash eks-log-collector.sh
> ```

该脚本收集以下诊断信息。您已部署的 CNI 版本可以早于脚本版本。

```
      This is version 0.6.1. New versions can be found at https://github.com/awslabs/amazon-eks-ami

Trying to collect common operating system logs...
```

```
Trying to collect kernel logs...
Trying to collect mount points and volume information...
Trying to collect SELinux status...
Trying to collect iptables information...
Trying to collect installed packages...
Trying to collect active system services...
Trying to collect Docker daemon information...
Trying to collect kubelet information...
Trying to collect L-IPAMD information...
Trying to collect sysctls information...
Trying to collect networking information...
Trying to collect CNI configuration information...
Trying to collect running Docker containers and gather container data...
Trying to collect Docker daemon logs...
Trying to archive gathered information...

 Done... your bundled logs are located in /var/log/eks_i-0717c9d54b6cfaa19_2020-03-24_0103-
UTC_0.6.1.tar.gz
```

诊断信息收集并存储在：

```
/var/log/eks_i-0717c9d54b6cfaa19_2020-03-24_0103-UTC_0.6.1.tar.gz
```

容器运行时网络未准备就绪

您可能会收到类似于以下内容的 `Container runtime network not ready` 错误和授权错误：

```
4191 kubelet.go:2130] Container runtime network not ready: NetworkReady=false
 reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config
 uninitialized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
4191 kubelet_node_status.go:106] Unable to register node "ip-10-40-175-122.ec2.internal"
 with API server: Unauthorized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
```

这些错误最有可能与 Amazon IAM Authenticator (`aws-auth`) 配置映射未应用到集群有关。配置映射提供了 `system:bootstrappers` 和 `system:nodes` Kubernetes RBAC 权限，以便节点注册到集群。要将配置映射应用于集群，请参阅 将 aws-authConfigMap 应用到集群 (p. 361)。

若角色 ARN包含 `/` 以外的路径，如下面的示例所示，则身份验证器无法识别该 ARN：

```
arn:aws:iam::111122223333:role/development/apps/prod-iam-role-NodeInstanceRole-621LVEXAMPLE
```

当配置映射中指定的角色 ARN 包含非 `/` 的路径时，必须删除该路径。应将前述 ARN 指定为以下内容：

```
arn:aws:iam::111122223333:role/prod-iam-role-NodeInstanceRole-621LVEXAMPLE
```

TLS 握手超时

当节点无法建立到公有 API 服务器端点的连接时，您可能会遇到类似如下的错误。

```
server.go:233] failed to run Kubelet: could not init cloud provider "aws": error finding
 instance i-1111f2222f333e44c: "error listing Amazon instances: \"RequestError: send
 request failed\\ncaused by: Post  net/http: TLS handshake timeout\""
```

kubelet 进程将持续重新生成并测试 API 服务器终端节点。在控制层面中执行集群滚动更新（例如配置更改或版本更新）的任何过程中，也可能临时发生此错误。

要解决此问题，请检查路由表和安全组，以确保来自节点的流量可以到达公有端点。

InvalidClientTokenId

如果您将 IAM 角色用于部署于中国 Amazon Web Services 区域 的集群的 pod 或 DaemonSet 的服务账户，但尚未在规范中设置 AWS_DEFAULT_REGION 环境变量，则 pod 或 DaemonSet 可能会收到以下错误：

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation: The
 security token included in the request is invalid
```

要解决此问题，您需要将 AWS_DEFAULT_REGION 环境变量添加到您的 pod 或 DaemonSet 规范中，如以下示例 pod 规范中所示。

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: AWS_DEFAULT_REGION
      value: "region-code"
```

VPC 准入 Webhook 证书过期

如果用于签署 VPC 准入 Webhook 的证书过期，则新的 Windows pod 部署的状态将保持在 ContainerCreating。

要解决数据面板有旧版 Windows 支持的问题，请参阅 the section called “更新 VPC 准入 Webhook 证书” (p. 55)。如果您的集群和平台版本高于 Windows 支持先决条件 (p. 48)部分中列出的版本，则建议删除数据面板上的旧版 Windows 支持，然后为控制面板启用新版支持。执行此操作后，无需管理 webhook 证书。有关更多信息，请参阅为 Amazon EKS 集群启用 Windows 支持 (p. 48)。

在更新控制面板前，节点组必须匹配 Kubernetes 版本

集群中的托管节点和 Fargate 节点的次要版本必须与控制面板的当前版本相同，然后才能将控制面板更新为新的 Kubernetes 版本。EKS update-cluster-version API 会拒绝请求，直到您将所有 EKS 托管节点更新为当前集群版本。EKS 提供 API 来更新托管节点。有关更新托管节点组 Kubernetes 版本的信息，请参阅 更新托管节点组 (p. 95)。要更新 Fargate 节点的版本，请删除节点所表示的 pod，然后在更新控制面板后重新部署 pod。有关更多信息，请参阅更新 Amazon EKS 集群 Kubernetes 版本 (p. 28)。

启动多个节点时，出现 Too Many Requests 错误

如果同时启动多个节点，您可能会在表示 Too Many Requests 的 Amazon EC2 用户数据执行日志中看到错误消息。发生这种情况的原因是控制层面被 describeCluster 调用过载。过载导致节流，节点无法运行引导脚本，节点无法完全加入集群。

确保 --apiserver-endpoint、--b64-cluster-ca 和 --dns-cluster-ip 参数正在传递给 Worker 节点引导脚本。加入这些参数后，引导脚本无需调用 describeCluster，这有助于防止控制层面过

载。有关更多信息，请参阅提供用户数据以将实际参数传递给随 Amazon EKS 优化版 AMI 一起提供的 bootstrap.sh 文件 (p. 103)。

对于 Kubernetes API 服务器请求的 HTTP 401 未经授权错误响应

如果 1.21 或更高版本集群上的 pod 的服务账户令牌已过期，您将会看到这些错误。

Amazon EKS 1.21 版或更高版本集群的 Kubernetes API 服务器拒绝令牌超过 90 天的请求。在之前的 Kubernetes 版本中，令牌没有过期时间。这意味着依赖这些令牌的客户端必须在一小时内刷新它们。为防止 Kubernetes API 服务器因令牌无效而拒绝您的请求，您的工作负载使用的 Kubernetes 客户端 SDK 版本必须与以下版本相同或高于以下版本：

- Go 版本 0.15.7 和更高版本
- Python 版本 12.0.0 和更高版本
- Java 版本 9.0.0 和更高版本
- JavaScript 版本 0.10.3 和更高版本
- Ruby master 分支
- Haskell 版本 0.3.0.0
- C# 版本 7.0.5 和更高版本

您可以识别您的集群中使用过时令牌的所有现有 pods。有关更多信息，请参阅 Kubernetes 服务账户 (p. 395)。

Amazon EKS 平台版本比当前平台版本落后两个版本以上

当 Amazon EKS 无法自动更新集群的平台版本 (p. 67)时，可能会发生这种情况。造成这种情况的原因有很多，而一些常见的原因如下。如果这些问题中的任何一个适用于您的集群，该集群可能仍然可以正常工作，但 Amazon EKS 不会更新其平台版本。

问题

集群 IAM 角色 (p. 425)被删除 – 该角色是在创建集群时指定的。可使用以下命令查看所指定的角色。将 *my-cluster* 替换为您集群的名称。

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut -d /
 -f 2
```

输出示例如下。

```
eksClusterRole
```

解决方案

创建具有相同名称的新集群 IAM 角色 (p. 425)。

问题

集群创建期间指定的子网被删除 – 用于集群的子网是在集群创建期间指定的。可使用以下命令查看所指定的子网。将 *my-cluster* 替换为您集群的名称。

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

输出示例如下。

```
[
"subnet-EXAMPLE1",
"subnet-EXAMPLE2"
]
```

解决方案

确认您的账户中是否存在这些子网 ID。

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --query
 "Subnets[*].SubnetId"
```

输出示例如下。

```
[
"subnet-EXAMPLE3",
"subnet-EXAMPLE4"
]
```

如果输出中返回的子网 ID 与创建集群时指定的子网 ID 不匹配，若您希望 Amazon EKS 更新集群，则可能需要创建一个新集群。这是因为，如果您在创建集群时指定了两个以上的子网，Amazon EKS 会随机选择您指定的子网，以便在其中创建新的弹性网络接口。这些网络接口使控制面板能够与您的节点进行通信。如果 Amazon EKS 选择的子网不存在，则不会更新集群。您在创建集群时指定了一些子网，但您无法控制 Amazon EKS 会选择哪个子网来创建新的网络接口。创建集群后，您无法创建新的子网以供集群使用。

当您对集群进行 Kubernetes 版本更新时，更新可能会因为同样的原因而失败。

问题

集群创建期间指定的安全组被删除 – 如果您在集群创建期间指定了安全组，则可以使用以下命令查看其 ID。将 *my-cluster* 替换为您集群的名称。

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.securityGroupIds
```

输出示例如下。

```
[
    "sg-EXAMPLE1"
]
```

如果返回 `[]`，则在创建集群时未指定安全组，且缺少安全组也不是问题所在。如果返回安全组，请确认您的账户中存在这些安全组。

解决方案

确认您的账户中是否存在这些安全组。

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$vpc_id" --query
 "SecurityGroups[*].GroupId"
```

输出示例如下。

```
[
"sg-EXAMPLE2"
]
```

如果输出中返回的安全组 ID 与创建集群时指定的安全组 ID 不匹配，若您希望 Amazon EKS 更新集群，则需要创建一个新集群。如果创建集群时指定的安全组 ID 不存在，Amazon EKS 将不会更新集群。创建集群后，您无法指定不同的安全组。

当您对集群进行 Kubernetes 版本更新时，更新可能会因为同样的原因而失败。

Amazon EKS 不更新集群平台版本的其他原因

- 在创建集群时指定的每个子网中，可用 IP 地址达不到至少 6 个（而我们建议 16 个）。如果子网中没有足够的可用 IP 地址，则需要释放子网中的 IP 地址，或者创建一个新集群，且该集群使用的子网具有足够的可用 IP 地址。
- 您在创建集群时启用了密钥加密 (p. 42)，但您指定的 Amazon KMS 密钥已被删除。如果您希望 Amazon EKS 更新集群，则需要创建一个新集群

IAM 故障排除

本主题介绍将 Amazon EKS 与 IAM 结合使用时可能遇到的一些常见错误以及相应的错误处理方式。

AccessDeniedException

如果您在调用 Amazon API 操作时收到 `AccessDeniedException`，则表明您使用的 Amazon Identity and Access Management (IAM) 用户或角色凭证没有发起该调用所需的权限。

```
An error occurred (AccessDeniedException) when calling the DescribeCluster operation:
User: arn:aws:iam::111122223333:user/user_name is not authorized to perform:
eks:DescribeCluster on resource: arn:aws:eks:region:111122223333:cluster/my-cluster
```

在前述示例消息中，用户没有权限调用 Amazon EKS `DescribeCluster` API 操作。要为用户提供 Amazon EKS 管理员权限，请参阅 Amazon EKS 基于身份的策略示例 (p. 415)。

有关 IAM 的一般信息，请参阅 IAM 用户指南中的使用策略控制访问。

无法在 Compute（计算）选项卡上看到 Nodes（节点），或在 Resources（资源）选项卡上看到任何内容，您将在 Amazon Web Services Management Console 中收到错误

您可能会看到一条内容为“`Your current user or role does not have access to Kubernetes objects on this EKS cluster`”的控制台错误消息。请确保您登录到 Amazon Web Services Management Console 所使用的 IAM 用户（或您在登录后切换到的角色）具有必要的权限。有关更多信息，请参阅所需权限 (p. 458)。

aws-auth `ConfigMap` 不授予对集群的访问权限

Amazon IAM 身份验证器不允许在 `ConfigMap` 中使用角色 ARN 中的路径。因此，在指定 `rolearn` 之前，请删除路径。例如，将 `arn:aws:iam::111122223333:role/team/developers/eks-admin` 更改为 `arn:aws:iam::111122223333:role/eks-admin`。

我无权执行 iam:PassRole

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 Amazon EKS。

有些 Amazon Web Services允许您将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 Amazon EKS 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 Amazon 管理员。管理员是向您提供登录凭证的人。

我想要查看我的访问密钥

在创建 IAM 用户访问密钥后，您可以随时查看您的访问密钥 ID。但是，您无法再查看您的秘密访问密钥。如果您丢失了私有密钥，则必须创建一个新的访问密钥对。

访问密钥包含两部分：访问密钥 ID（例如 `AKIAIOSFODNN7EXAMPLE`）和秘密访问密钥（例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

> **Important**
>
> 请不要向第三方提供访问密钥，即便是为了帮助找到您的规范用户 ID 也不行。如果您这样做，可能会向某人提供对您的账户的永久访问权限。

当您创建访问密钥对时，系统会提示您将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果丢失了您的秘密访问密钥，您必须为 IAM 用户添加新的访问密钥。您最多可拥有两个访问密钥。如果您已有两个密钥，则必须删除一个密钥对，然后再创建新的密钥。要查看说明，请参阅 IAM 用户指南中的管理访问密钥。

我是管理员并希望允许其他人访问 Amazon EKS

要允许其他人访问 Amazon EKS，您必须为需要访问权限的人员或应用程序创建一个 IAM 实体（用户或角色）。它们将使用该实体的凭证访问 Amazon。然后，您必须将策略附加到实体，以便在 Amazon EKS 中为他们（它们）授予正确的权限。

要立即开始使用，请参阅 IAM 用户指南中的创建您的第一个 IAM 委派用户和组。

我希望允许我的Amazon账户以外的人访问我的 Amazon EKS 资源

您可以创建一个角色，以便其它账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon EKS 是否支持这些功能，请参阅 Amazon EKS 如何与 IAM 配合使用 (p. 412)。
- 要了解如何为您拥有的 Amazon Web Services 账户 中的资源提供访问权限，请参阅 IAM 用户指南中的为您拥有的另一个 Amazon Web Services 账户 中的 IAM 用户提供访问权限。
- 要了解如何为第三方Amazon Web Services 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的Amazon Web Services 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的为经过外部身份验证的用户（联合身份验证）提供访问权限。

- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。

容器组（pod）容器收到以下错误：An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region.

如果您的应用程序明确向 Amazon STS 全局端点 (`https://sts.amazonaws`) 提出请求并且您的 Kubernetes 服务账户已配置为使用区域端点，您的容器将收到此错误。您可以使用以下选项之一解决问题：

- 更新应用程序代码以删除对 Amazon STS 全局端点的显式调用。
- 更新应用程序代码以明确调用区域端点，例如 `https://sts.cn-north-1.amazonaws.com.cn`。您的应用程序应内置冗余功能，以便在该 Amazon Web Services 区域 的服务出现故障时选择其他 Amazon Web Services 区域。有关更多信息，请参阅 IAM 用户指南中的在 Amazon Web Services 区域中管理 Amazon STS。
- 将服务账户配置为使用全局端点。低于 `1.22` 的所有版本默认情况下使用全局端点，但 `1.22` 版本和更高版本集群默认使用区域端点。如果您的代码对 `1.22` 或更高版本集群上的全局端点进行显式调用，如果您使用的是默认端点类型，则会收到错误消息。有关更多信息，请参阅配置服务账户的 Amazon Security Token Service 端点 (p. 405)。

排查 Amazon EKS Connector 中的问题

本主题介绍您在使用 Amazon EKS Connector 时可能遇到的一些常见错误，包括有关如何解决这些错误的说明、变通方法和常见问题。

常见问题

本节介绍如何排查使用 Amazon EKS Connector 时可能遇到的一些功能问题。它提供了这些问题的解决方案和变通方法。

控制台错误：集群卡在待处理状态

如果注册集群之后，集群在 Amazon EKS 控制台中卡在 `Pending` 状态，可能是因为 Amazon EKS Connector 尚未成功将集群连接到 Amazon。对于已注册的集群，`Pending` 状态表示尚未成功建立连接。要解决此问题，请确保您已将清单应用到目标 Kubernetes 集群。如果您确实将其应用到集群，但集群仍处于“待处理”状态，则很可能是 Amazon EKS Connector 在您的集群中运行状况不佳。要排查此问题，请参阅本主题中名为 Amazon EKS Connector pods 处于崩溃循环 (p. 490) 的章节。

控制台错误：用户“system:serviceaccount:eks-connector:eks-connector”无法在集群范围内模拟 API 组“”中的资源“用户”

Amazon EKS Connector 使用 Kubernetes 用户模拟代表用户从 Amazon Web Services Management Console 进行操作。对于从 Amazon eks-connector 服务账户访问 Kubernetes API 的每个 IAM 身份，必须授予其使用 IAM ARN 作为用户名模拟相应 Kubernetes 用户的权限。在以下示例中，IAM ARN 映射到 Kubernetes 用户。

- 来自 Amazon 账户 *111122223333* 的 IAM 用户 *john* 映射到 Kubernetes 用户。

```
arn:aws:iam::111122223333:user/john
```

- 来自 Amazon 账户 *111122223333* 的 IAM 角色 *admin* 映射到 Kubernetes 用户。

```
arn:aws:iam::111122223333:role/admin
```

  结果是 IAM 角色 ARN，而不是 STS 会话 ARN。

有关如何配置 `ClusterRole` 和 `ClusterRoleBinding` 以授予 `eks-connector` 服务账户模拟映射用户权限的说明，请参阅 向用户授予查看集群上的 Kubernetes 资源的访问权限 (p. 498)。确保在模板中将 %IAM_ARN% 替换为 Amazon Web Services Management Console 用户的 IAM ARN。

控制台错误：... 已禁止：用户 ... 无法在集群范围内列出“API 组中的 ...”资源

考虑以下问题。Amazon EKS Connector 已成功模拟目标 Kubernetes 集群中请求的 Amazon Web Services Management Console 用户。但模拟用户没有 Kubernetes API 操作的 RBAC 权限。

要解决此问题，作为集群管理员，您必须向各个 Kubernetes 用户授予适当级别的 RBAC 权限。有关更多信息以及示例，请参阅 向用户授予查看集群上的 Kubernetes 资源的访问权限 (p. 498)。

控制台错误：Amazon EKS 无法与您的 Kubernetes 集群 API 服务器进行通信。集群必须处于 ACTIVE（活动）状态才能成功连接。过几分钟再试。

如果 Amazon EKS 服务无法与目标集群中的 Amazon EKS Connector 进行通信，可能是由于以下原因之一：

- 目标集群中的 Amazon EKS Connector 运行状况不佳。
- 目标集群与 Amazon Web Services 区域之间的连接不佳或连接中断。

要解决此问题，请查看 检查 Amazon EKS Connector 的日志 (p. 492)。如果您没有看到 Amazon EKS Connector 的错误，请在几分钟后重试连接。如果经常遇到目标集群的高延迟或间歇性连接，请考虑将集群重新注册到离您更近的 Amazon Web Services 区域 区域。

Amazon EKS Connector pods 处于崩溃循环

可导致 EKS Connector pod 进入 CrashLoopBackOff 状态的原因有很多。此问题可能涉及 connector-init 容器。在以下示例中，是 connector-init 容器进入 CrashLoopBackOff 状态。

```
kubectl get pods -n eks-connector
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:CrashLoopBackOff   1          7s
```

要排查此问题，检查 Amazon EKS Connector 的日志 (p. 492)。

无法启动 eks-connector: InvalidActivation

初次启动 Amazon EKS Connector 时，它会向亚马逊云科技注册 `activationId` 和 `activationCode`。注册可能会失败，进而可能导致 `connector-init` 容器崩溃。

```
F1116 20:30:47.261469       1 init.go:43] failed to initiate eks-connector:
 InvalidActivation:
```

要排查此问题，请考虑以下原因和建议的修复方法：

- 注册失败可能是因为 activationId 和 activationCode 不在清单文件中。如果是这种情况，请确保它们是从 RegisterCluster API 操作返回的正确值，并且 activationCode 位于清单文件中。activationCode 已添加到 Kubernetes 密钥中，因此必须为 base64 编码。有关更多信息，请参阅步骤 1：注册集群 (p. 496)。

- 注册失败可能是因为激活已过期。这是因为，出于安全原因，您必须在注册集群后的 3 天内激活 EKS Connector。要解决此问题，请确保在到期日期和时间之前将 EKS Connector 清单应用到目标 Kubernetes 集群。要确认激活到期日期，请调用 DescribeCluster API 操作。在以下示例响应中，到期日期和时间记录为 2021-11-12T22:28:51.101000-08:00。

```
aws eks describe-cluster --name my-cluster
{
    "cluster": {
        "name": "my-cluster",
        "arn": "arn:aws:eks:us-east-1:123456789012:cluster/my-cluster",
        "createdAt": "2021-11-09T22:28:51.449000-08:00",
        "status": "FAILED",
        "tags": {
        },
        "connectorConfig": {
            "activationId": "00000000-0000-0000-0000-000000000000",
            "activationExpiry": "2021-11-12T22:28:51.101000-08:00",
            "provider": "OTHER",
            "roleArn": "arn:aws:iam::123456789012:role/my-connector-role"
        }
    }
}
```

  如果 `activationExpiry` 已过，则注销集群然后重新注册。执行此操作会生成新的激活信息。

集群 Worker 节点缺少出站连接

为正常工作，EKS Connector 需要到多个 Amazon 终端节点的出站连接。如果没有到目标 Amazon Web Services 区域 的出站连接，则无法连接私有集群。要解决此问题，您必须添加必要的出站连接。有关连接器要求的信息，请参阅 Amazon EKS Connector 注意事项 (p. 495)。

Amazon EKS Connector pods 处于 ImagePullBackOff 状态

如果您运行 getpods 命令且 pods 处于 ImagePullBackOff 状态，则其无法正常工作。如果 EKS Connector pod 处于 `ImagePullBackOff` 状态，则其无法正常工作。请参见以下示例：

```
kubectl get pods -n eks-connector
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:ImagePullBackOff   0          4s
```

默认 Amazon EKS Connector 清单文件引用来自 Amazon ECR Public Registry 的镜像。目标 Kubernetes 集群可能无法从 Amazon ECR Public 拉取镜像。解决 Amazon ECR Public 镜像拉取问题，或考虑镜像您选择的私有容器注册表中的镜像。

常见问题

问：Amazon EKS Connector 背后的底层技术是如何工作的？

答：Amazon EKS Connector 基于 System Manager (SSM) Agent。EKS Connector 作为 StatefulSet 在您的 Kubernetes 集群上运行。它建立连接并代理集群的 API 服务器与亚马逊云科技之间的通信。这样做是为了在 Amazon EKS 控制台中显示集群数据，直到您将集群与 Amazon 断开连接。SSM Agent 是一个开源项目。有关此项目的更多信息，请参阅 GitHub 项目页面。

问：我想要连接一个本地部署的 Kubernetes 集群。我需要打开防火墙端口来连接它吗？

答：不，您不需要打开任何防火墙端口。Kubernetes 集群只需要到 Amazon Web Services 区域 的出站连接。Amazon 服务永远不会访问本地部署网络中的资源。Amazon EKS Connector 在您的集群上运行并启

动与 Amazon 的连接。集群注册完成后，Amazon 仅在您从 Amazon EKS 控制台启动操作（需要集群中 Kubernetes API 服务器的信息）后向 Amazon EKS Connector 发出命令。

问：Amazon EKS Connector 将哪些数据从我的集群发送到 Amazon？

答：Amazon EKS Connector 会发送在 Amazon 上注册集群所必需的技术信息。它还发送客户请求的 Amazon EKS 控制台功能的集群和工作负载元数据。Amazon EKS Connector 仅当您从 Amazon EKS 控制台或 Amazon EKS API 启动的操作需要将数据发送到 Amazon 时才会收集或发送此数据。默认情况下，Amazon 不会存储除 Kubernetes 版本号以外的任何数据。它仅在您授权的情况下才会存储数据。

问：我可以连接 Amazon Web Services 区域 以外的集群吗？

答：可以，您可以将任何位置的集群连接到 Amazon EKS。此外，您的 EKS 服务可以位于任何 Amazon 公共商业 Amazon Web Services 区域。这适用于从集群到目标 Amazon Web Services 区域的有效网络连接。我们建议您选择距集群位置最近的 Amazon Web Services 区域 以优化 UI 性能。例如，如果您的集群在东京运行，请将集群连接到东京的 Amazon Web Services 区域（即 `ap-northeast-1` Amazon Web Services 区域）以实现低延迟。您可以将任何位置的集群连接到任何公共商业 Amazon Web Services 区域（中国或 GovCloud Amazon Web Services 区域 除外）中的 Amazon EKS。

基本问题排查

本节介绍问题不清楚时诊断问题的步骤。

检查 Amazon EKS Connector 状态

使用以下命令检查 Amazon EKS Connector 状态。

```
kubectl get pods -n eks-connector
```

检查 Amazon EKS Connector 的日志

Amazon EKS Connector Pod 包含三个容器。要检索所有这些容器的完整日志以便进行查看，请运行以下命令：

- `connector-init`

```
kubectl logs eks-connector-0 --container connector-init -n eks-connector
kubectl logs eks-connector-1 --container connector-init -n eks-connector
```

- `connector-proxy`

```
kubectl logs eks-connector-0 --container connector-proxy -n eks-connector
kubectl logs eks-connector-1 --container connector-proxy -n eks-connector
```

- `connector-agent`

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector -- cat /var/
log/amazon/ssm/amazon-ssm-agent.log
kubectl exec eks-connector-1 --container connector-agent -n eks-connector -- cat /var/
log/amazon/ssm/amazon-ssm-agent.log
```

获取有效的集群名称

Amazon EKS 集群通过一个 Amazon 账户和 Amazon Web Services 区域内的 `clusterName` 唯一标识。如果您在 Amazon EKS 中有多个连接的集群，则可以确认当前 Kubernetes 集群注册了哪个 Amazon EKS 集群。为此，请输入以下内容以找出当前集群的 `clusterName`。

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log \
  | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+" \
  | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
kubectl exec eks-connector-1 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log \
  | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+" \
  | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
```

其他命令

以下命令可用于检索排查问题所需的信息。

- 使用以下命令可收集 Amazon EKS Connector 中 pods 使用的镜像。

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}" \
  | tr -s '[[:space:]]' '\n'
```

- 使用以下命令可收集运行 Amazon EKS Connector 的 Worker 节点名称。

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" \
  | tr -s '[[:space:]]' '\n'
```

- 运行以下命令可获取您的 Kubernetes 客户端和服务器版本。

```
kubectl version
```

- 运行以下命令可获取有关 Worker 节点的信息。

```
kubectl get nodes -o wide --show-labels
```

排查 ADOT Amazon EKS 附加组件中的问题

本主题介绍您在使用 ADOT Amazon EKS 附加组件时可能遇到的一些常见错误，包括有关如何解决这些错误的说明、变通方法和常见问题。

常见问题

本节介绍您在使用 ADOT Amazon EKS 附加组件和排查步骤问题时可能遇到的功能问题。

错误： `"code": "AccessDenied", "message": "roles.rbac.authorization.k8s.io \"opentelemetry-operator-leader-election-role\" is forbidden: User \"eks:addon-manager\" cannot patch resource \"roles\" in API group \"rbac.authorization.k8s.io\" in the namespace \"opentelemetry-operator-system\"`

您没有权限安装 ADOT for Amazon EKS 附加组件。请参阅安装 Amazon Distro for OpenTelemetry (ADOT) Operator (p. 470)。如果您已删除附加组件并且现在正在重新安装，请确保您已应用权限 (p. 468)。

创建附加组件错误： `"status": "CREATE_FAILED"`

- 可能存在冲突。您可以通过添加标志来覆盖冲突：`--resolve-conflicts=OVERWRITE` 并再次运行 create 命令。

- 不受支持的架构，例如 ARM64。查看日志以确定是否属于这种情况。

删除附加组件错误： `"status": "DELETE_FAILED"`

您可以通过将 `--preserve` 标志添加到 delete 命令中使 ADOT Operator 不再受 Amazon EKS 附加组件管理。

Amazon EKS Connector

Important

此功能在 Amazon Web Services 区域（中国）不可用。

您可以使用 Amazon EKS Connector 注册并将任何符合要求的 Kubernetes 集群连接至 Amazon，并在 Amazon EKS 控制台中进行显示。连接集群后，您可以在 Amazon EKS 控制台中查看集群的状态、配置和工作负载。您可以使用此功能在 Amazon EKS 控制台中查看已连接的群集，但您无法对其进行管理。Amazon EKS Connector 能够将以下类型的 Kubernetes 集群连接到 Amazon EKS。Amazon EKS Connector 也是 Github 上的开源项目。有关其他技术内容，包括常见问题和故障排除，请参阅 排查 Amazon EKS Connector 中的问题 (p. 489)

- 本地 Kubernetes 集群
- 在 Amazon EC2 上运行的自托管式集群
- 来自其他云提供商的管理集群

Amazon EKS Connector 注意事项

在使用 Amazon EKS Connector 之前，请先了解以下内容：

- 您必须拥有 Kubernetes 集群的管理权限才能将集群连接到 Amazon EKS。
- 连接前，Kubernetes 集群必须有 Linux 64 位 (x86) Worker 节点。不支持 ARM Worker 节点。
- 您的 Kubernetes 集群中必须包含拥有对 `ssm.` 和 `ssmmessages.` Systems Manager 端点出站访问的 Worker 节点。有关更多信息，请参阅 Amazon 一般参考中的 Systems Manager 端点。
- 每个区域原定设置最多可连接 10 个集群。您可以通过服务配额控制台请求增加配额。请参阅请求增加配额了解更多信息。
- 对于外部 Kubernetes 集群，仅支持 Amazon EKS `RegisterCluster`、`ListClusters`、`DescribeCluster` 和 `DeregisterCluster` API。
- 您必须拥有以下权限才能注册集群：
  - eks:RegisterCluster
  - ssm:CreateActivation
  - ssm:DeleteActivation
  - iam:PassRole
- 您必须拥有以下权限才能注销集群：
  - eks:DeregisterCluster
  - ssm:DeleteActivation
  - ssm:DeregisterManagedInstance

Amazon EKS Connector 所需的 IAM 角色

使用 Amazon EKS Connector 需要以下两个 IAM 角色：

- 您注册集群时需要创建 Amazon EKS Connector 服务相关角色。
- 必须手动创建 Amazon EKS Connector 代理 IAM 角色。有关详细信息，请参阅 Amazon EKS Connector IAM 角色 (p. 433)。

要为另一个用户启用集群和工作负载视图权限，必须将 `eks-connector` 和 Amazon EKS Connector 集群角色应用到集群。按 向用户授予查看集群上的 Kubernetes 资源的访问权限 (p. 498) 中的步骤操作。

连接集群

步骤 1：注册集群

您可以使用 Amazon CLI 和 Amazon Web Services Management Console 将外部 Kubernetes 集群连接到 Amazon EKS。此过程包括两个步骤：向 Amazon EKS 注册集群和应用 YAML 清单文件以启用连接。要允许其他用户查看集群，请按照 向用户授予查看集群上的 Kubernetes 资源的访问权限 (p. 498) 中的说明操作。

您必须拥有以下权限才能注册集群：

- `eks:RegisterCluster`
- `ssm:CreateActivation`
- `ssm:DeleteActivation`
- `iam:PassRole`

eksctl

先决条件

- 必须安装 `eksctl` 版本 `0.68` 或更高版本。要安装或升级，请参阅《`eksctl` 入门》。
- 创建 Amazon EKS Connector 代理 IAM 角色。有关更多信息，请参阅 Connector IAM 角色。

要使用 **`eksctl`** 注册集群

1. 提供名称、提供程序和区域来注册集群。

```
eksctl register cluster --name my-cluster --provider my-provider --region region-code
```

   这会在本地驱动器上创建两个文件：`my-first-registered-cluster.yaml` 和 `eks-connector-binding.yaml` 文件。必须在 3 天内将这两个文件应用到外部集群，否则注册过期。
2. 在集群的本机环境中应用 `eks-connector-binding.yaml` 文件：

```
kubectl apply -f eks-connector-binding.yaml
```

Amazon CLI

先决条件

- 必须安装 Amazon CLI。要进行安装或升级，请参阅安装 Amazon CLI。
- 确保创建了 Amazon EKS Connector 代理角色。

要使用 Amazon CLI 注册集群

1. 对于 Connector 配置，请指定您的 Amazon EKS Connector 代理 IAM 角色。有关更多信息，请参阅Amazon EKS Connector 所需的 IAM 角色 (p. 495)。

```
aws eks register-cluster \
     --name my-first-registered-cluster \
     --connector-config roleArn=arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole,provider="OTHER" \
     --region AWS_REGION
```

输出示例如下。

```
{
    "cluster": {
        "name": "my-first-registered-cluster",
        "arn": "arn:aws:eks:region:111122223333:cluster/my-first-registered-
cluster",
        "createdAt": 1627669203.531,
        "ConnectorConfig": {
            "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxxx",
            "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxxx",
            "activationExpiry": 1627672543.0,
            "provider": "OTHER",
            "roleArn": "arn:aws:iam::111122223333:role/AmazonEKSConnectorAgentRole"
        },
        "status": "CREATING"
    }
}
```

您将在后续步骤中使用 region、activationId 和 activationCode 值。

2. 下载 Amazon EKS Connector YAML 文件。

```
curl -o eks-connector.yaml https://amazon-eks.s3.us-west-2.amazonaws.com/eks-
connector/manifests/eks-connector/latest/eks-connector.yaml
```

3. 编辑 Amazon EKS Connector YAML 文件以将 %AWS_REGION%、%EKS_ACTIVATION_ID%、%EKS_ACTIVATION_CODE% 的所有引用替换为注册命令输出结果中的 region、activationId 和 activationCode。

   以下示例命令可以替换这些值。

```
sed -i "s~%AWS_REGION%~$AWS_REGION~g; s~%EKS_ACTIVATION_ID%~$EKS_ACTIVATION_ID~g;
 s~%EKS_ACTIVATION_CODE%~$(echo -n $EKS_ACTIVATION_CODE | base64)~g" eks-
connector.yaml
```

   Important

   确保激活码采用 base64 格式。

Amazon Web Services Management Console

先决条件

- 确保创建了 Amazon EKS Connector 代理角色。

要向控制台注册 Kubernetes 集群。

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择 Add cluster（添加集群），然后选择 Register（注册）以打开配置页面。
3. 在 Configure cluster (配置集群) 部分，填写以下字段：

- Name（名称）– 集群的唯一名称。
- Provider（提供程序）– 选择以显示 Kubernetes 集群提供程序的下拉列表。如果您不了解具体的提供程序，请选择 Other（其他）。
- EKS Connector 角色：选择用于连接集群的角色。

4. 选择 Register cluster（注册集群）。
5. 此时将显示集群概览页面。选择 Download YAML file（下载 YAML 文件），以将清单文件下载到本地驱动程序。

   Important

   - 这是您下载此文件的唯一机会。不要离开此页面，否则将无法访问此链接，届时您必须注销集群并从头开始此步骤。
   - 对于已注册集群，只能使用一次清单文件。如果从 Kubernetes 集群中删除资源，则必须重新注册集群并获取新的清单文件。

   继续执行下一步，以将清单文件应用于 Kubernetes 集群。

步骤 2：应用清单文件

通过将 Amazon EKS Connector 清单文件应用于 Kubernetes 集群来完成连接。要执行此操作，必须对前面描述的注册方法使用 Amazon CLI 或 `eksctl`。如果没有在 3 天内应用清单，Amazon EKS Connector 注册将过期。如果集群连接过期，则必须在再次连接集群之前注销集群。

1. 在集群的本机环境中，您可以运行以下命令应用更新的清单文件：

```
kubectl apply -f eks-connector.yaml
```

2. 将绑定 Amazon EKS Connector 清单和角色的 YAML 文件应用于 Kubernetes 集群后，确认该集群现在已连接。

```
aws eks describe-cluster \
     --name "my-first-registered-cluster" \
     --region AWS_REGION
```

   该输出应包含 `status=ACTIVE`。
3. 现在，您可以向集群添加标签（可选）。有关更多信息，请参阅标记 Amazon EKS 资源。

要向其他 IAM 用户授予对 Amazon EKS 控制台的访问权限以查看已连接的集群中的 Kubernetes 资源，请参阅 向用户授予查看集群上的 Kubernetes 资源的访问权限 (p. 498)。

向用户授予查看集群上的 Kubernetes 资源的访问权限

授权其他 IAM 用户访问 Amazon EKS 控制台，查看有关在已连接集群上运行的 Kubernetes 资源的信息。

先决条件

您访问 Amazon Web Services Management Console 所用的 IAM 用户或角色必须满足以下要求。

- 它拥有 `eks:AccessKubernetesApi` 权限。
- Amazon EKS Connector Service 账户应能够模拟集群中的 IAM 或角色。这让 eks-connector 将 IAM 用户或角色映射到 Kubernetes 用户。

创建并应用 Amazon EKS Connector 集群角色

1. 下载 `eks-connector` 集群角色模板。

```
curl -o eks-connector-clusterrole.yaml https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-clusterrole.yaml
```

2. 编辑集群角色模板 YAML 文件。将 `%IAM_ARN%` 参考替换为您的 IAM 用户或角色的 Amazon Resource Name (ARN)。
3. 将 Amazon EKS Connector 集群角色 YAML 应用于您的 Kubernetes 集群。

```
kubectl apply -f eks-connector-clusterrole.yaml
```

要使 IAM 用户或角色在 Amazon EKS 控制台上查看 Kubernetes 资源，用户或角色必须与 Kubernetes `role` 或 `clusterrole` 关联，并拥有读取这些资源的必要权限。有关更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

要配置 IAM 用户以访问连接的集群

1. 您可以下载示例清单，以创建 `clusterrole` 和 `clusterrolebinding` 或 `role` 和 `rolebinding`：
   - 查看所有命名空间中的 Kubernetes 资源 - `eks-connector-console-dashboard-full-access-clusterrole` 集群角色提供可在控制台中可视化的所有命名空间和资源的访问权限。您可以更改 `role`、`clusterrole` 及其对应绑定的名称，然后再将其应用于集群。使用以下命令下载示文件。

```
curl -o eks-connector-console-dashboard-full-access-group.yaml https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-full-access-group.yaml
```

   - 查看特定命名空间中的 Kubernetes 资源 – 此文件中的命名空间是 `default`，因此，如果要指定不同的命名空间，请编辑该文件，然后将其应用到集群。使用以下命令下载示例文件。

```
curl -o eks-connector-console-dashboard-restricted-access-group.yaml https://s3.cn-north-1.amazonaws.com.cn/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-restricted-access-group.yaml
```

2. 编辑完全访问权限或受限访问权限 YAML 文件以将 `%IAM_ARN%` 参考替换为您的 IAM 用户或角色的 Amazon Resource Name (ARN)。
3. 将完全访问权限或受限访问权限 YAML 文件应用于 Kubernetes 集群。将 YAML 文件的值替换为您自己的值。

```
kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
```

要查看已连接集群中的 Kubernetes 资源，请参阅 查看 Kubernetes 资源 (p. 458)。Resources（资源）选项卡上某些资源类型的数据不适用于已连接的集群。

注销集群

使用完已连接集群后，可以将其注销。注销后，该集群将不再在 Amazon EKS 控制台中显示。

您必须拥有以下权限才能调用 deregisterCluster API：

- eks:DeregisterCluster
- ssm:DeleteActivation
- ssm:DeregisterManagedInstance

eksctl

先决条件

- 必须安装 `eksctl` 版本 `0.68` 或更高版本。要安装或升级，请参阅《`eksctl` 入门》。
- 确保创建了 Amazon EKS Connector 代理角色。

要使用 **`eksctl`** 取消注册集群

- 对于 Connector 配置，请指定您的 Amazon EKS Connector 代理 IAM 角色。有关更多信息，请参阅Amazon EKS Connector 所需的 IAM 角色 (p. 495)。

```
eksctl deregister cluster --name my-cluster
```

Amazon CLI

先决条件

- 必须安装 Amazon CLI。要进行安装或升级，请参阅安装 Amazon CLI。
- 确保创建了 Amazon EKS Connector 代理角色。

注销已连接的集群。

```
aws eks deregister-cluster \
     --name my-cluster \
     --region region-code
```

Amazon Web Services Management Console

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择 Clusters (集群)。
3. 请在 Clusters（集群）页面上，选择连接的集群，然后选择 Deregister（注销）。
4. 确认您要注销该层。

要清除 Kubernetes 集群上的资源。

1. 从 Kubernetes 集群中删除 Amazon EKS Connector YAML 文件。

```
kubectl delete -f eks-connector.yaml
```

2. 如果您为其他 IAM 用户创建了 `clusterrole` 或 `clusterrolebinding` 来访问集群，请确保从 Kubernetes 集群中删除它们。

相关项目

这些开源项目扩展了在 Amazon 上或在其外部运行的 Kubernetes 集群的功能，包括由 Amazon EKS 托管的集群。

管理工具

Amazon EKS 和 Kubernetes 集群的相关管理工具。

eksctl

`eksctl` 是一个用于在 Amazon EKS 上创建集群的简单 CLI 工具。

- 项目 URL
- 项目文档
- Amazon开源博客：eksctl：使用一个命令的 Amazon EKS 集群

Amazon Controllers for Kubernetes

使用 Amazon Controllers for Kubernetes，您可以直接从 Kubernetes 集群创建和管理 Amazon 资源。

- 项目 URL
- Amazon 开源博客：Kubernetes 的 Amazon 服务运算符现已推出

Flux CD

Flux 是一个工具，通过它，您可以使用 Git 来管理集群配置。它使用集群中的运算符来触发 Kubernetes 内部的部署。有关运算符的详细信息，请参阅 GitHub 上的 Awesome Operators in the Wild。

- 项目 URL
- 项目文档

CDK for Kubernetes

借助 CDK for Kubernetes (cdk8s)，您可以使用熟悉的编程语言定义 Kubernetes 应用程序和组件。cdk8s 应用程序合成为标准 Kubernetes 清单，可应用于任何 Kubernetes 集群。

- 项目 URL
- 项目文档
- Amazon 容器博客：推出 cdk8s+：面向 Kubernetes 对象的意图驱动 API

联网

Amazon EKS 和 Kubernetes 集群的相关联网项目。

Amazon VPC CNI plugin for Kubernetes

Amazon EKS 支持通过 Amazon VPC CNI plugin for Kubernetes 进行本机 VPC 联网。此插件会将 VPC 中的 IP 地址分配给每个 pod。

- 项目 URL
- 项目文档

Amazon Load Balancer Controller for Kubernetes

Amazon Load Balancer Controller 有助于管理适用于 Kubernetes 集群的 Amazon 弹性负载均衡器。它通过预置 Amazon Application Load Balancer 来满足 Kubernetes 入口资源。它通过预置 Amazon 网络负载均衡器来满足 Kubernetes 服务资源。

- 项目 URL
- 项目文档

ExternalDNS

ExternalDNS 将公开的 Kubernetes 服务和入口与 DNS 提供程序进行同步，包括 Amazon Route 53 和 Amazon 服务发现。

- 项目 URL
- 项目文档

App Mesh 控制器

适用于 Kubernetes 的 App Mesh 控制器有助于为您的集群管理 App Mesh。使用该控制器，您可以使用集群中的自定义资源管理服务网格。该控制器管理向 pods 注入网络代理 Sidecar，以启用网格。

- 项目 URL
- 项目文档
- Amazon 博客：App Mesh 和 Amazon EKS 入门

安全性

Amazon EKS 和 Kubernetes 集群的相关安全项目。

Amazon IAM 身份验证器

一种使用 Amazon IAM 凭证向 Kubernetes 集群进行身份验证的工具（如果您未使用 Amazon CLI 版本 `1.16.156` 或更高版本）。有关更多信息，请参阅安装 `aws-iam-authenticator` (p. 369)。

- 项目 URL
- 项目文档
- Amazon 开源博客：将 Amazon IAM 身份验证器部署到 Kop

Machine Learning

Amazon EKS 和 Kubernetes 集群的相关 Machine Learning 项目。

Kubeflow

Kubernetes 的机器学习工具包。

- 项目 URL
- 项目文档
- Amazon 开源博客：Amazon EKS 上的 Kubeflow

Auto Scaling

Amazon EKS 和 Kubernetes 集群的相关 Auto Scaling 项目。

Cluster Autoscaler

Cluster Autoscaler 是一个根据 CPU 和内存压力自动调整 Kubernetes 集群大小的工具。

- 项目 URL
- 项目文档
- Amazon EKS 研讨会：https://eksworkshop.com/scaling/deploy_ca/

Escalator

Escalator 是适用于 Kubernetes 的已优化批处理或作业的水平自动缩放器。

- 项目 URL
- 项目文档

监控

Amazon EKS 和 Kubernetes 集群的相关监控项目。

Prometheus

Prometheus 是一个开源系统监控和警报工具包。

- 项目 URL
- 项目文档
- Amazon EKS 研讨会：https://eksworkshop.com/intermediate/240_monitoring/

持续集成/持续部署

Amazon EKS 和 Kubernetes 集群的相关 CI/CD 项目。

Jenkins X

Amazon EKS 和 Kubernetes 集群上现代云应用程序的 CI/CD 解决方案。

- 项目 URL
- 项目文档

Amazon EKS 新功能和路线图

您可以通过滚动到Amazon新功能页面上的“新功能”馈送，了解 Amazon EKS 新功能。您也可以在 GitHub 上查看路线图，通过它了解即将推出的功能和优先工作，从而规划未来使用 Amazon EKS 的方式。您可以直接向我们提供有关路线图优先工作的反馈。

Amazon EKS 文件历史记录

下表描述了 Amazon EKS 用户指南的主要更新和新功能。我们还经常更新文档来处理您发送给我们的反馈意见。

update-history-change	update-history-description	update-history-date
基于 Fargate vCPU 的配额	Fargate 将从基于容器组（pod）的配额过渡到基于 vCPU 的配额。	2022 年 9 月 8 日
成本监控	Amazon EKS 现在支持 Kubecost，使您能够监控按 Kubernetes 资源（包括 pods、节点、命名空间和标签）细分的成本。	2022 年 8 月 24 日
针对账单为资源添加标签	为所有集群增加了 `aws:eks:cluster-name` 生成的成本分配标签支持。	2022 年 8 月 16 日
Fargate 配置文件通配符	在命名空间、标签键和标签值的选择器标准中添加了对 Fargate 配置文件通配符的支持。	2022 年 8 月 16 日
Kubernetes 版本 `1.23`	增加了新集群和版本升级的 Kubernetes 版本 `1.23` 支持。	2022 年 8 月 11 日
查看 Amazon Web Services Management Console 中的 Kubernetes 资源	现在，您可以查看使用 Amazon Web Services Management Console 部署到集群的 Kubernetes 资源的相关信息。	2022 年 5 月 3 日
Observation（观察）页面和 ADOT 附加组件支持	添加了 Observability（可观察性）页面和 Amazon Distro for OpenTelemetry (ADOT)。	2022 年 4 月 21 日
Kubernetes 版本 `1.22`	增加了新集群和版本升级的 Kubernetes 版本 `1.22` 支持。	2022 年 4 月 22 日
增加了 Fargate pod 修补详细信息	升级 Fargate pods 时，Amazon EKS 首先尝试根据您的 pod 中断预算驱逐 pods。您可以创建事件规则来在删除 pods 之前对失败的驱逐做出反应。	2022 年 4 月 1 日
完整版本：对 Amazon EBS CSI 驱动程序的附加组件支持	您现在可以使用 Amazon Web Services Management Console、Amazon CLI 和 API来管理 Amazon EBS CSI 驱动程序。	2022 年 3 月 31 日
Amazon Outposts 内容更新	在 Amazon Outposts 上部署 Amazon EKS 集群的说明。	2022 年 3 月 22 日
Windows`containerd` 支持	现在，您可以选择 Windows 节点的 `containerd` 运行时。	2022 年 3 月 14 日

向安全文档中增加了 Amazon EKS Connector 考虑因素	描述与连接集群相关的责任共担模型。	2022 年 2 月 25 日
将 IPv6 地址分配到您的 pods 和服务	您现在可以创建 1.21 或更高版本的集群，将 IPv6 地址分配到您的 pods 和服务。	2022 年 1 月 6 日
预览版本：对 Amazon EBS CSI 驱动程序的附加组件支持	您现在可以使用 Amazon Web Services Management Console、Amazon CLI 和 API 进行预览，以管理 Amazon EBS CSI 驱动程序。	2021 年 12 月 9 日
Karpenter Autoscaler 支持	现在您可以使用 Karpenter 开源项目自动缩放节点。	2021 年 11 月 29 日
Fargate 日志记录中的 Fluent Bit Kubernetes 筛选支持	现在您可以将 Fluent Bit Kubernetes 筛选用于 Fargate 日志记录。	2021 年 11 月 10 日
控制面板提供 Windows 支持	控制面板现在提供 Windows 支持。无需在数据层面中启用。	2021 年 11 月 9 日
添加 Bottlerocket 作为托管节点组的 AMI 类型	以前，Bottlerocket 仅作为自行管理节点选项提供。现在可以将其配置为托管节点组，从而减少满足节点合规性要求所需的工作量。	2021 年 10 月 28 日
DL1 驱动程序支持	自定义 Amazon Linux AMI 现在支持 Amazon Linux 2 的深度学习工作负载。这样支持通用本地部署或云基准配置。	2021 年 10 月 25 日
VT1 视频支持	自定义 Amazon Linux AMI 现在支持部分配的 VT1。此支持功能可在您的 Amazon EKS 集群上传播 Xilinx U30 设备。	2021 年 9 月 13 日
Amazon EKS Connector 现已发布	您可以使用 Amazon EKS Connector 注册并将任何符合要求的 Kubernetes 集群连接至 Amazon，并在 Amazon EKS 控制台中进行显示。	2021 年 9 月 8 日
Amazon EKS Anywhere 现已发布	Amazon EKS Anywhere 是 Amazon EKS 的部署选项，使您能够在本地轻松创建和操作 Kubernetes 集群。	2021 年 9 月 8 日
Amazon FSx for NetApp ONTAP CSI 驱动程序	添加了总结 Amazon FSx for NetApp ONTAP CSI 驱动程序的主题，并提供了其他参考的链接。	2021 年 9 月 2 日
现在，托管节点组会自动计算 Amazon EKS 建议的节点最大 pods 数量	对于您未使用启动模板部署，或使用未在其中指定 AMI ID 启动模板部署的节点，托管节点组现在会自动为其计算 Amazon EKS 最大 pods 数量。	2021 年 8 月 30 日

在不移除 Amazon EKS 附加组件软件的情况下移除附加组件设置的 Amazon EKS 管理	现在，您可以移除 Amazon EKS 附加组件，而无需从集群中移除附件组件软件。	2021 年 8 月 20 日
使用 Multus 创建多宿主 pods	您现在可以使用 Multus 向一个 pod 添加多个网络接口。	2021 年 8 月 2 日
向您的 Linux Amazon EC2 节点添加更多 IP 地址	您现在可以向您的 Linux Amazon EC2 节点添加大量 IP 地址。这意味着您可以在每个节点上运行更高密度的 pods。	2021 年 7 月 27 日
Kubernetes 版本 `1.21`	增加了 Kubernetes 版本 `1.21` 支持。	2021 年 7 月 19 日
`Containerd` 运行时间引导	Amazon EKS 优化版加速型 Amazon Linux Amazon 机器映像（AMI）现在包含一个引导标记，您可以用它在 Amazon EKS 优化版和 Bottlerocket AMI 中启用 `containerd` 运行时间。此标记可用于所有受支持的 Kubernetes 版本的 AMI。	2021 年 7 月 19 日
已添加托管策略主题	所有 Amazon EKS IAM 托管策略以及自 2021 年 6 月 17 日以来对其做出的更改的列表。	2021 年 6 月 17 日
将 pods 的安全组与 Fargate 结合使用	除了将 pods 的安全组与 Amazon EC2 节点一起使用之外，您现在还可以将其与 Fargate 一起使用。	2021 年 6 月 1 日
已添加 CoreDNS 和 `kube-proxy` Amazon EKS 附加组件	Amazon EKS 现在可以帮助您管理集群的 CoreDNS 和 `kube-proxy` 附加组件。	2021 年 5 月 19 日
Kubernetes 版本 `1.20`	增加了新集群和版本升级的 Kubernetes 版本 `1.20` 支持。	2021 年 5 月 18 日
发布了 Amazon Load Balancer Controller`2.2.0`	现在，您可以使用 Amazon Load Balancer Controller，以利用实例或 IP 目标创建 Elastic Load Balancer。	2021 年 5 月 14 日
托管节点组的节点污点	Amazon EKS 现在支持向托管节点组添加节点污点。	2021 年 5 月 11 日
现有集群的密钥加密	Amazon EKS 现在支持向现有集群添加密钥加密。	2021 年 2 月 26 日
Kubernetes 版本 `1.19`	增加了新集群和版本升级的 Kubernetes 版本 `1.19` 支持。	2021 年 2 月 16 日
Amazon EKS 现在支持 OpenID Connect (OIDC) 身份提供商作为对 `1.16` 版或更高版本集群的用户进行身份验证的方法。	OIDC 身份提供商可与 Amazon Identity and Access Management (IAM) 一起使用或作为其替代方法。	2021 年 2 月 12 日

在 Amazon Web Services Management Console 中查看节点和工作负载资源	您现在可以在 Amazon Web Services Management Console 中查看有关托管、自行管理和 Fargate 节点以及您部署的 Kubernetes 工作负载的详细信息。	2020 年 12 月 1 日
在托管节点组中部署竞价型实例类型	您现在可以将多种 Spot 或按需实例类型部署到托管节点组。	2020 年 12 月 1 日
Amazon EKS 现在可以管理集群的特定附加组件	您可以自行管理附加组件，也可以让 Amazon EKS 通过 Amazon EKS API 控制附加组件的启动和版本，以用于运行 Kubernetes 版 `1.18` 和 `eks.3` 版或更高版本平台的集群。	2020 年 12 月 1 日
跨多个入口共享 ALB	现在，您可以共享跨多个 Kubernetes 入口共享 Amazon 应用程序负载均衡器（ALB）。过去，您必须为每个入口部署单独的 ALB。	2020 年 10 月 23 日
NLB IP 目标支持	您现在可以部署具有 IP 目标的网络负载均衡器。这意味着您可以使用 NLB 将网络流量负载均衡到 Fargate pods 以及直接均衡到运行于 Amazon EC2 节点上的 pods。	2020 年 10 月 23 日
Kubernetes 版本 `1.18`	增加了新集群和版本升级的 Kubernetes 版本 `1.18` 支持。	2020 年 10 月 13 日
为 Kubernetes 服务 IP 地址分配指定自定义 CIDR 块。	现在可以指定 Kubernetes 从中分配服务 IP 地址的自定义 CIDR 块。	2020 年 9 月 29 日
将安全组分配到单个 pods	现在，您可以将不同的安全组与在许多 Amazon EC2 实例类型上运行的某些单独的 pods 关联起来。	2020 年 9 月 9 日
在节点上 Bottlerocket 部署	您现在可以部署运行 Bottlerocket 的节点。	2020 年 8 月 31 日
启动 Arm 节点的功能通常可用	现在，您可以在托管节点组和自行管理节点组中启动 Arm 节点。	2020 年 8 月 17 日
托管节点组启动模板和自定义 AMI	您现在可使用 Amazon EC2 启动模板部署托管节点组。如果您选择，则启动模板可指定自定义 AMI。	2020 年 8 月 17 日
对 Amazon Fargate 的 EFS 支持	您现在可以将 Amazon EFS 与 Amazon Fargate 结合使用	2020 年 8 月 17 日

Amazon EKS 平台版本更新	这是一个新的平台版本（包括安全修复和增强功能）。这包括在将网络负载均衡器用于 Kubernetes 版本 1.15 或更高版本时 UDP 对 LoadBalancer 类型服务的支持。有关更多信息，请参阅 GitHub 上的允许 UDP 用于 Amazon 网络负载均衡器。	2020 年 8 月 12 日
Amazon EKS Amazon Web Services 区域 扩展 (p. 507)	Amazon EKS 现已在非洲（开普敦）（af-south-1）和欧洲（米兰）（eu-south-1）Amazon Web Services 区域推出。	2020 年 8 月 6 日
Fargate 使用情况指标	Amazon Fargate 提供 CloudWatch 使用情况指标，通过该指标可以了解您对 Fargate 按需资源的账户使用情况。	2020 年 8 月 3 日
Kubernetes 版本 1.17 (p. 507)	增加了新集群和版本升级的 Kubernetes 版本 1.17 支持。	2020 年 7 月 10 日
使用适用于 Kubernetes 的 App Mesh 控制器，在 Kubernetes 中创建和管理 App Mesh 资源	您可以在 Kubernetes 中创建和管理 App Mesh 资源。控制器还会自动在您部署的 pods 中注入 Envoy 代理和 Init 容器。	2020 年 6 月 18 日
Amazon EKS 现在支持 Amazon EC2 Inf1 节点	您可以向集群添加 Amazon EC2 Inf1 节点。	2020 年 6 月 4 日
Amazon EKS Amazon Web Services 区域 扩展 (p. 507)	Amazon EKS 现已在 Amazon GovCloud（美国东部）（us-gov-east-1）和 Amazon GovCloud（美国西部）（us-gov-west-1）Amazon Web Services 区域推出。	2020 年 5 月 13 日
Amazon EKS 不再支持 Kubernetes1.12	Amazon EKS 不再支持 Kubernetes 版本 1.12。将所有 1.12 版集群更新到版本 1.13 或更高版本以避免服务中断。	2020 年 5 月 12 日
Kubernetes 版本 1.16 (p. 507)	增加了新集群和版本升级的 Kubernetes 版本 1.16 支持。	2020 年 4 月 30 日
添加了 AWSServiceRoleForAmazonEKS 服务相关角色	添加了 AWSServiceRoleForAmazonEKS 服务相关角色。	2020 年 4 月 16 日
Kubernetes 版本 1.15 (p. 507)	增加了新集群和版本升级的 Kubernetes 版本 1.15 支持。	2020 年 3 月 10 日
Amazon EKS Amazon Web Services 区域 扩展 (p. 507)	Amazon EKS 现已在北京（cn-north-1）和宁夏（cn-northwest-1）Amazon Web Services 区域 推出。	2020 年 2 月 26 日

FSx for Lustre CSI 驱动程序	增加了在 Kubernetes `1.14` Amazon EKS 集群上安装 FSx for Lustre CSI 驱动程序的主题。	2019 年 12 月 23 日
限制对集群的公有访问端点的网络访问	此次更新后，您可以利用 Amazon EKS，限制可与 Kubernetes API 服务器的公有访问端点通信的 CIDR 范围。	2019 年 12 月 20 日
在 VPC 外部解析集群的私有访问端点地址	通过此项更新，您可以利用 Amazon EKS，在 VPC 外部解析 Kubernetes API 服务器的私有访问端点。	2019 年 12 月 13 日
（测试版）Amazon EC2 A1 Amazon EC2 实例节点	启动向 Amazon EKS 集群注册的 Amazon EC2 A1 Amazon EC2 实例节点。	2019 年 12 月 4 日
在 Amazon Outpost 上创建集群	Amazon EKS 现在支持在 Amazon Outposts 上创建集群。	2019 年 12 月 3 日
Amazon EKS 上的 Amazon Fargate	Amazon EKS Kubernetes 集群现在支持在 Fargate 上运行 pods。	2019 年 12 月 3 日
Amazon EKS Amazon Web Services 区域 扩展 (p. 507)	Amazon EKS 现已在加拿大（中部）(`ca-central-1`) Amazon Web Services 区域推出。	2019 年 11 月 21 日
托管节点组	Amazon EKS 托管节点组为 Amazon EKS Kubernetes 集群自动对节点（Amazon EC2 实例）进行预置和生命周期管理。	2019 年 11 月 18 日
Amazon EKS 平台版本更新	新平台版本，以解决 `CVE-2019-11253`。	2019 年 11 月 6 日
Amazon EKS 不再支持 Kubernetes`1.11`	Amazon EKS 不再支持 Kubernetes 版本 `1.11`。请将所有 `1.11` 版集群更新到版本 `1.12` 或更高版本以避免服务中断。	2019 年 11 月 4 日
Amazon EKS Amazon Web Services 区域 扩展 (p. 507)	Amazon EKS 现已在南美洲（圣保罗）(`sa-east-1`) Amazon Web Services 区域推出。	2019 年 10 月 16 日
Windows 支持	现在，运行 Kubernetes 版本 `1.14` 的 Amazon EKS 集群支持 Windows 工作负载。	2019 年 10 月 7 日
Autoscaling	添加了一章，用于介绍 Amazon EKS 集群支持的不同类型的 Kubernetes 弹性伸缩。	2019 年 9 月 30 日
Kubernetes 控制面板更新	更新了在 Amazon EKS 集群上安装 Kubernetes 控制面板以使用 Beta `2.0` 版的主题。	2019 年 9 月 28 日

Amazon EFS CSI 驱动程序	增加了在 Kubernetes `1.14` Amazon EKS 集群上安装 Amazon EFS CSI 驱动程序的主题。	2019 年 9 月 19 日
Amazon EKS 优化版 AMI ID 的 Amazon EC2 Systems Manager 参数	添加了使用 Amazon EC2 Systems Manager 参数检索 Amazon EKS 优化版 AMI ID 的主题。使用该参数，您无需查找 AMI ID。	2019 年 9 月 18 日
Amazon EKS 资源标记	您可以管理 Amazon EKS 集群的标记。	2019 年 9 月 16 日
Amazon EBS CSI 驱动程序	增加了在 Kubernetes `1.14` Amazon EKS 集群上安装 Amazon EBS CSI 驱动程序的主题。	2019 年 9 月 9 日
针对 `CVE-2019-9512` 和 `CVE-2019-9514` 进行修补的新的 Amazon EKS 优化版 AMI	Amazon EKS 更新了 Amazon EKS 优化版 AMI，用于解决 `CVE-2019-9512` 和 `CVE-2019-9514`。	2019 年 9 月 6 日
宣布在 Amazon EKS 中弃用 Kubernetes`1.11`	Amazon EKS 已于 2019 年 11 月 4 日停止支持 Kubernetes `1.11` 版。	2019 年 9 月 4 日
Kubernetes 版本 `1.14` (p. 507)	增加了新集群和版本升级的 Kubernetes 版本 `1.14` 支持。	2019 年 9 月 3 日
服务账户的 IAM 角色	通过 Amazon EKS 集群上的服务账户的 IAM 角色，您可以将 IAM 角色与 Kubernetes 服务账户关联。借助此功能，您不再需要向节点 IAM 角色提供扩展的权限。这样，该节点上的 pods 即可调用 Amazon API。	2019 年 9 月 3 日
Amazon EKS Amazon Web Services 区域 扩展 (p. 507)	Amazon EKS 现已在中东（巴林）(`me-south-1`) Amazon Web Services 区域推出。	2019 年 8 月 29 日
Amazon EKS 平台版本更新	新平台版本，以解决 `CVE-2019-9512` 和 `CVE-2019-9514`。	2019 年 8 月 28 日
Amazon EKS 平台版本更新	新平台版本，以解决 `CVE-2019-11247` 和 `CVE-2019-11249`。	2019 年 8 月 5 日
Amazon EKS 区域扩展 (p. 507)	Amazon EKS 现已在亚太地区（香港）(`ap-east-1`) Amazon Web Services 区域推出。	2019 年 7 月 31 日
Amazon EKS 不再支持 Kubernetes`1.10`	Amazon EKS 不再支持 Kubernetes 版本 `1.10`。将所有 `1.10` 版集群更新到版本 `1.11` 或更高版本以避免服务中断。	2019 年 7 月 30 日

增加了有关 ALB 入口控制器的主题	适用于 Kubernetes 的 Amazon ALB 入口控制器是一类控制器，在创建入口资源时，此控制器将触发 ALB 的创建。	2019 年 7 月 11 日
新的 Amazon EKS 优化版 AMI	从 AMI 删除不必要的 `kubectl` 二进制文件。	2019 年 7 月 3 日
Kubernetes 版本 `1.13` (p. 507)	增加了新集群和版本升级的 Kubernetes 版本 `1.13` 支持。	2019 年 6 月 18 日
针对 `AWS-2019-005` 进行修补的新的 Amazon EKS 优化版 AMI	Amazon EKS 已更新 Amazon EKS 优化版 AMI，解决了 `AWS-2019-005` 中描述的漏洞。	2019 年 6 月 17 日
宣布在 Amazon EKS 中停止对 Kubernetes`1.10` 的支持	2019 年 7 月 22 日，Amazon EKS 已停止支持 Kubernetes 版本 `1.10`。	2019 年 5 月 21 日
Amazon EKS 平台版本更新	Kubernetes `1.11` 和 `1.10` 集群的新平台版本支持在 `kubelet` 证书中使用自定义 DNS 名称并提高 `etcd` 性能。	2019 年 5 月 21 日
Amazon CLI`get-token` (p. 507) 命令	`aws eks get-token` 命令已添加到 Amazon CLI。您不再需要安装适用于 Kubernetes 的 Amazon IAM Authenticator 来创建用于集群 API 服务器通信的客户端安全令牌。将您的 Amazon CLI 安装升级到最新版本以使用此新功能。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。	2019 年 5 月 10 日
`eksctl` 入门	此入门指南描述如何使用 `eksctl` 安转开始使用 Amazon EKS 所需的所有资源。这是一种用于在 Amazon EKS 上创建和管理 Kubernetes 集群的简单命令行实用程序。	2019 年 5 月 10 日
Amazon EKS 平台版本更新	Kubernetes `1.12` 集群的新平台版本支持在 `kubelet` 证书中使用自定义 DNS 名称并提高 `etcd` 性能。这将修复导致节点 `kubelet` 守护程序每隔几秒便会请求新证书的漏洞。	2019 年 5 月 8 日
Prometheus 教程	增加了将 Prometheus 部署到 Amazon EKS 集群的主题。	2019 年 4 月 5 日
Amazon EKS 控制面板日志记录	通过此更新，您可以直接从 Amazon EKS 控制窗格中获取审计和诊断日志。您可以在账户中使用这些 CloudWatch 日志作为参考来保护和运行集群。	2019 年 4 月 4 日

Kubernetes 版本 1.12 (p. 507)	增加了新集群和版本升级的 Kubernetes 版本 1.12 支持。	2019 年 3 月 28 日
增加了 App Mesh 入门指南	增加了开始使用 App Mesh 和 Kubernetes 的文档。	2019 年 3 月 27 日
Amazon EKS API 服务器端点私有访问	增加了如何禁用 Amazon EKS 集群的 Kubernetes API 服务器端点的公有访问的文档。	2019 年 3 月 19 日
增加了安装 Kubernetes Metrics Server 的主题	Kubernetes Metrics Server 是集群中资源使用情况数据的聚合器。	2019 年 3 月 18 日
增加了相关开源项目的列表	这些开源项目扩展了在 Amazon 上运行的 Kubernetes 集群的功能，包括 Amazon EKS 管理的集群。	2019 年 3 月 15 日
增加了本地安装 Helm 的主题	Kubernetes 的 helm 序包管理器帮助您在集 Kubernetes 群上安装和管理应用程序。本主题介绍了如何在本地安装和运行 helm 和 tiller 二进制文件。这样，您可以在本地系统中使用 Helm CLI 安装和管理图表。	2019 年 3 月 11 日
Amazon EKS 平台版本更新	新平台版本将 Amazon EKS Kubernetes 1.11 集群更新到补丁级别 1.11.8 以解决 CVE-2019-1002100。	2019 年 3 月 8 日
提高了集群限制	Amazon EKS 将您可以在 Amazon Web Services 区域中创建的集群数从 3 增加到 50。	2019 年 2 月 13 日
Amazon EKS Amazon Web Services 区域 扩展 (p. 507)	Amazon EKS 现已在欧洲（伦敦）（eu-west-2）、欧洲（巴黎）（eu-west-3）和亚太地区（孟买）（ap-south-1）Amazon Web Services 区域 推出。	2019 年 2 月 13 日
针对 ALAS-2019-1156 进行修补的新的 Amazon EKS 优化版 AMI	Amazon EKS 已更新 Amazon EKS 优化版 AMI，解决了 ALAS-2019-1156 中描述的漏洞。	2019 年 2 月 11 日
针对 ALAS2-2019-1141 进行修补的新的 Amazon EKS 优化版 AMI	Amazon EKS 更新了 Amazon EKS 优化版 AMI，以处理 ALAS2-2019-1141 中引用的 CVE。	2019 年 1 月 9 日
Amazon EKS Amazon Web Services 区域 扩展 (p. 507)	Amazon EKS 现已在亚太地区（首尔）(ap-northeast-2) Amazon Web Services 区域推出。	2019 年 1 月 9 日

Amazon EKS 区域扩展 (p. 507)	Amazon EKS 现已在以下其他 Amazon Web Services 区域推出：欧洲（法兰克福）（eu-central-1）、亚太地区（东京）（ap-northeast-1）、亚太地区（新加坡）（ap-southeast-1）和亚太地区（悉尼）（ap-southeast-2）。	2018 年 12 月 19 日
Amazon EKS 集群更新	增加了 Amazon EKS 集群 Kubernetes 版本更新和节点替换文档。	2018 年 12 月 12 日
Amazon EKS Amazon Web Services 区域 扩展 (p. 507)	Amazon EKS 现已在欧洲（斯德哥尔摩）(eu-north-1) Amazon Web Services 区域推出。	2018 年 12 月 11 日
Amazon EKS 平台版本更新	将 Kubernetes 更新到补丁级别 1.10.11 以解决 CVE-2018-1002105 的新平台版本。	2018 年 12 月 4 日
为 ALB 接收控制器添加了 1.0.0 版本支持	ALB 接收控制器发行版本 1.0.0，提供来自 Amazon 的正式支持。	2018 年 11 月 20 日
添加了对 CNI 网络配置的支持	Amazon VPC CNI plugin for Kubernetes 版本 1.2.1 现在支持辅助 pod 网络接口的自定义网络配置。	2018 年 10 月 16 日
添加了对 MutatingAdmissionWebhook 和 ValidatingAdmissionWebhook 的支持	Amazon EKS 平台版本 1.10-eks.2 现在支持 MutatingAdmissionWebhook 和 ValidatingAdmissionWebhook 准入控制器。	2018 年 10 月 10 日
添加了合作伙伴 AMI 信息	Canonical 还与 Amazon EKS 合作创建了可在您的集群中使用的节点 AMI。	2018 年 10 月 3 日
添加了有关 Amazon CLIupdate-kubeconfig 命令的说明	Amazon EKS 向 Amazon CLI 添加了 update-kubeconfig，以简化用于访问集群的 kubeconfig 文件的创建过程。	2018 年 9 月 21 日
新的 Amazon EKS 优化版 AMI	Amazon EKS 更新了 Amazon EKS 优化版 AMI（使用和不使用 GPU 支持），以提供各种安全修复和 AMI 优化。	2018 年 9 月 13 日
Amazon EKS Amazon Web Services 区域 扩展 (p. 507)	Amazon EKS 现已在欧洲（爱尔兰）（eu-west-1）区域推出。	2018 年 9 月 5 日
Amazon EKS 平台版本更新	支持 Kubernetes 聚合层和 Horizontal Pod Autoscaler（HPA）的新平台版本。	2018 年 8 月 31 日

新增了 Amazon EKS 优化版 AMI 和 GPU 支持	Amazon EKS 更新了 Amazon EKS 优化版 AMI，以使用新的 Amazon CloudFormation 节点模板和引导脚本。此外，还提供了新的具有 GPU 支持的 Amazon EKS 优化版 AMI。	2018 年 8 月 22 日
针对 ALAS2-2018-1058 进行修补的新的 Amazon EKS 优化版 AMI	Amazon EKS 更新了 Amazon EKS 优化版 AMI，以处理 ALAS2-2018-1058 中引用的 CVE。	2018 年 8 月 14 日
Amazon EKS 优化版 AMI 生成脚本	Amazon EKS 已公开用于生成 Amazon EKS 优化版 AMI 的生成脚本的源代码。GitHub 上现在提供这些生成脚本。	2018 年 7 月 10 日
Amazon EKS 初始版本 (p. 507)	服务的初始文档发布	2018 年 5 月 6 日